Markov chain Monte Carlo (MCMC)

- MCMC focus the posterior density evaluations to the part of the parameter space where the most of the posterior mass is

Markov chain Monte Carlo (MCMC)

- MCMC focus the posterior density evaluations to the part of the parameter space where the most of the posterior mass is
- This lecture introduces two simplest MCMC algorithms Gibbs and Metropolis
  - these help to understand the basic idea
  - in your assignment you implement Metropolis algorithm

Markov chain Monte Carlo (MCMC)

- MCMC focus the posterior density evaluations to the part of the parameter space where the most of the posterior mass is
- This lecture introduces two simplest MCMC algorithms Gibbs and Metropolis
  - these help to understand the basic idea
  - in your assignment you implement Metropolis algorithm
- Next week introduces more elaborate and complex MCMC algorithm
  - after that in your assignments and projects, you will use ready made efficient implementation

Markov chain Monte Carlo (MCMC)

- MCMC focus the posterior density evaluations to the part of the parameter space where the most of the posterior mass is
- This lecture introduces two simplest MCMC algorithms Gibbs and Metropolis
  - these help to understand the basic idea
  - in your assignment you implement Metropolis algorithm
- Next week introduces more elaborate and complex MCMC algorithm
  - after that in your assignments and projects, you will use ready made efficient implementation
- This lecture introduces also necessary diagnostics to check whether MCMC results are useful

Chapter 11

- 11.1 Gibbs sampler
- 11.2 Metropolis and Metropolis-Hastings
- 11.3 Using Gibbs and Metropolis as building blocks
- 11.4 Inference and assessing convergence (important)
  - potential scale reduction $\widehat{R}$ (R-hat)
- 11.5 Effective number of simulation draws (important)
  - effective sample size (ESS / S_{eff})
- 11.6 Example: hierarchical normal model (quick glance)

Chapter 11 demos

- demo11_1: Gibbs sampling
- demo11_2: Metropolis sampling
- demo11_3: Convergence of Markov chain
- demo11_4: split-$\widehat{R}$ and effective sample size (ESS or S_{eff})
- demo11_5: Diagnostics with `posterior` and `bayesplot` packages

It's all about expectations (reminder)

$$E_{p(\theta|y)}[f(\theta)] = \int f(\theta) p(\theta \mid y) d\theta,$$

$$\text{where} \quad p(\theta \mid y) = \frac{p(y \mid \theta) p(\theta)}{\int p(y \mid \theta) p(\theta) d\theta}$$

It's all about expectations (reminder)

$$E_{p(\theta|y)}[f(\theta)] = \int f(\theta)p(\theta \mid y)d\theta,$$

$$\text{where} \quad p(\theta \mid y) = \frac{p(y \mid \theta)p(\theta)}{\int p(y \mid \theta)p(\theta)d\theta}$$

We can easily evaluate $p(y \mid \theta)p(\theta)$ for any θ, but the integral $\int p(y \mid \theta)p(\theta)d\theta$ is usually difficult.

It's all about expectations (reminder)

$$E_{p(\theta|y)}[f(\theta)] = \int f(\theta)p(\theta \mid y)d\theta,$$

$$\text{where} \quad p(\theta \mid y) = \frac{p(y \mid \theta)p(\theta)}{\int p(y \mid \theta)p(\theta)d\theta}$$

We can easily evaluate $p(y \mid \theta)p(\theta)$ for any θ, but the integral $\int p(y \mid \theta)p(\theta)d\theta$ is usually difficult.

We can use the unnormalized posterior $q(\theta \mid y) = p(y \mid \theta)p(\theta)$, for example, in

It's all about expectations (reminder)

$$E_{p(\theta|y)}[f(\theta)] = \int f(\theta)p(\theta \mid y)d\theta,$$

$$\text{where} \quad p(\theta \mid y) = \frac{p(y \mid \theta)p(\theta)}{\int p(y \mid \theta)p(\theta)d\theta}$$

We can easily evaluate $p(y \mid \theta)p(\theta)$ for any θ, but the integral $\int p(y \mid \theta)p(\theta)d\theta$ is usually difficult.

We can use the unnormalized posterior $q(\theta \mid y) = p(y \mid \theta)p(\theta)$, for example, in

- Grid (equal spacing) evaluation with self-normalization

$$E_{p(\theta|y)}[f(\theta)] \approx \frac{\sum_{s=1}^{S}\left[f(\theta^{(s)})q(\theta^{(s)} \mid y)\right]}{\sum_{s=1}^{S} q(\theta^{(s)} \mid y)}$$

It's all about expectations (reminder)

$$E_{p(\theta|y)}[f(\theta)] = \int f(\theta)p(\theta \mid y)d\theta,$$

$$\text{where} \quad p(\theta \mid y) = \frac{p(y \mid \theta)p(\theta)}{\int p(y \mid \theta)p(\theta)d\theta}$$

We can easily evaluate $p(y \mid \theta)p(\theta)$ for any θ, but the integral $\int p(y \mid \theta)p(\theta)d\theta$ is usually difficult.

We can use the unnormalized posterior $q(\theta \mid y) = p(y \mid \theta)p(\theta)$, for example, in

- Grid (equal spacing) evaluation with self-normalization

$$E_{p(\theta|y)}[f(\theta)] \approx \frac{\sum_{s=1}^{S}\left[f(\theta^{(s)})q(\theta^{(s)} \mid y)\right]}{\sum_{s=1}^{S} q(\theta^{(s)} \mid y)}$$

- Monte Carlo methods which can sample from $p(\theta^{(s)} \mid y)$ using only $q(\theta^{(s)} \mid y)$

$$E_{p(\theta|y)}[f(\theta)] \approx \frac{1}{S}\sum_{s=1}^{S} f(\theta^{(s)})$$

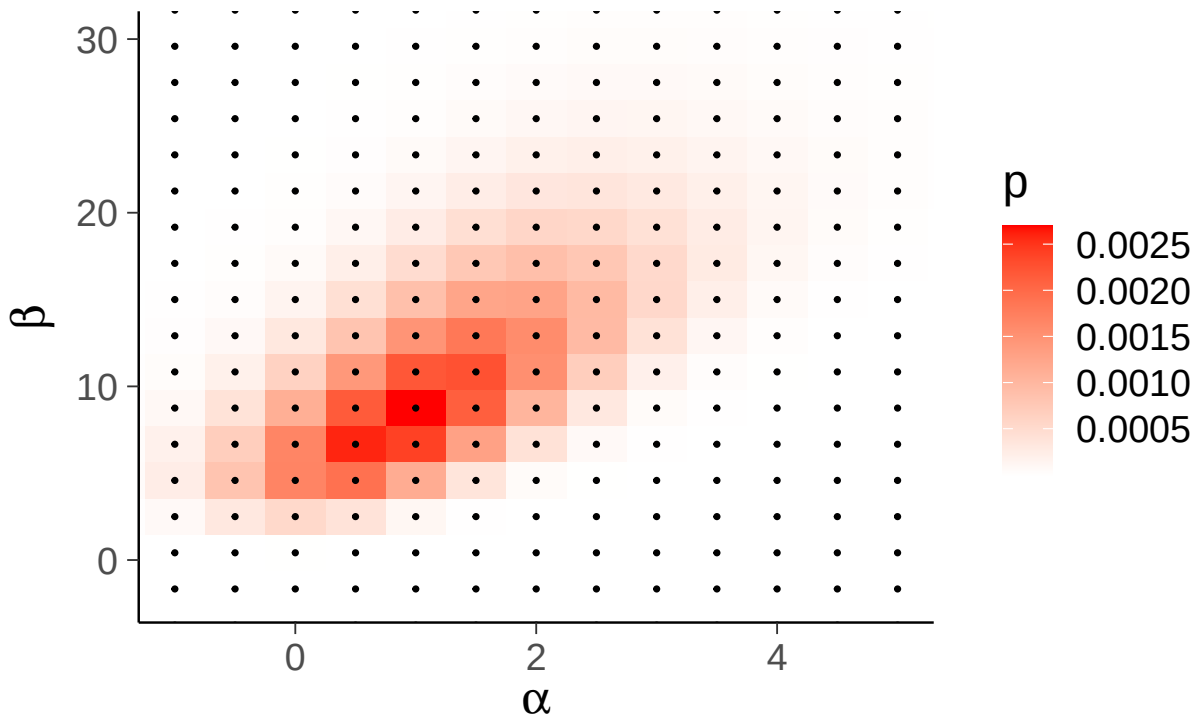
Monte Carlo

- Monte Carlo methods we have discussed so far
  - Inverse CDF works for 1D
  - Analytic transformations work for only certain distributions
  - Factorization works only for certain joint distributions
  - Grid evaluation and sampling works in a few dimensions
  - Rejection sampling works mostly in 1D (truncation is a special case)
  - Importance sampling is reliable only in special cases

Monte Carlo

- Monte Carlo methods we have discussed so far
  - Inverse CDF works for 1D
  - Analytic transformations work for only certain distributions
  - Factorization works only for certain joint distributions
  - Grid evaluation and sampling works in a few dimensions
  - Rejection sampling works mostly in 1D (truncation is a special case)
  - Importance sampling is reliable only in special cases

Posterior density evaluated in a grid

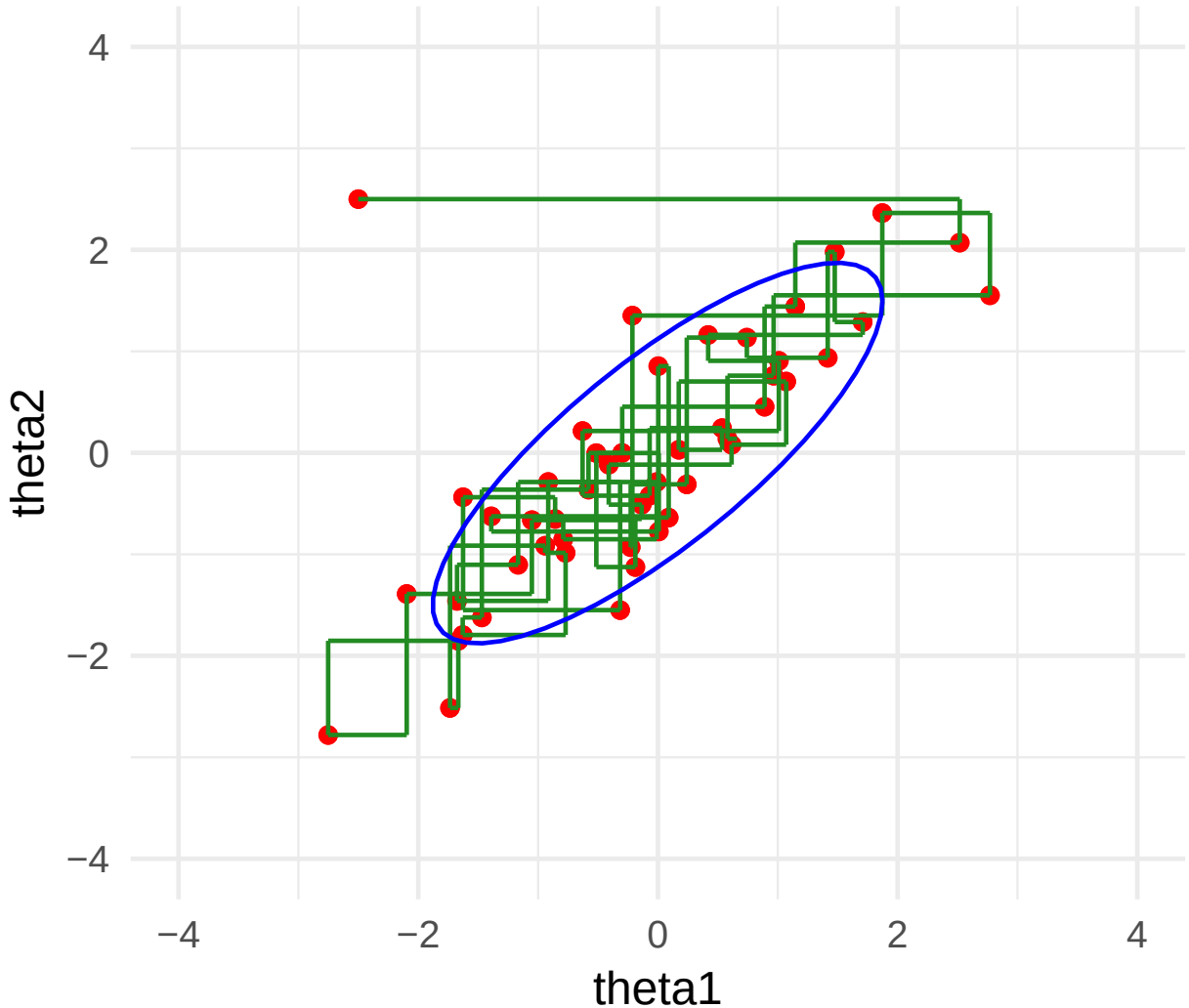


Monte Carlo

- Monte Carlo methods we have discussed so far
  - Inverse CDF works for 1D
  - Analytic transformations work for only certain distributions
  - Factorization works only for certain joint distributions
  - Grid evaluation and sampling works in a few dimensions
  - Rejection sampling works mostly in 1D (truncation is a special case)
  - Importance sampling is reliable only in special cases
- What to do in high dimensions?
  - Markov chain Monte Carlo (Ch 11-12)
  - Laplace, Variational*, EP* (Ch 4,13*)

Markov chain Monte Carlo (MCMC)

- Automatically focuses density evaluations where most of the posterior mass is



Markov chain

- Andrey Markov proved weak law of large numbers and central limit theorem for certain dependent-random sequences, which were later named Markov chains
  - CLT saying the sum / mean converges towards normal if the variance is finite

Markov chain

- Andrey Markov proved weak law of large numbers and central limit theorem for certain dependent-random sequences, which were later named Markov chains
  - CLT saying the sum / mean converges towards normal if the variance is finite
- The probability of each event depends only on the state attained in the previous event (or finite number of previous events)

Markov chain

- Andrey Markov proved weak law of large numbers and central limit theorem for certain dependent-random sequences, which were later named Markov chains
  - CLT saying the sum / mean converges towards normal if the variance is finite
- The probability of each event depends only on the state attained in the previous event (or finite number of previous events)
- Markov estimated the transition probabilities for the 20 000 first wovels and consonants in Pushkin's novel "Yevgeniy Onegin"

Markov chain

- Andrey Markov proved weak law of large numbers and central limit theorem for certain dependent-random sequences, which were later named Markov chains
  - CLT saying the sum / mean converges towards normal if the variance is finite
- The probability of each event depends only on the state attained in the previous event (or finite number of previous events)
- Markov estimated the transition probabilities for the 20 000 first wovels and consonants in Pushkin's novel "Yevgeniy Onegin"
  - Markov's model was a very small language model

Markov chain and language models (off-topic)

- Deep learning language models are super big Markov models
- Brian Hayes (1983, *A progress report on the fine art of turning literature into drivel*), experimented with 0–8th order Markov chains for producing text. For example, using William Faulkner's story "Two Soldiers" to compute 8th order transition probabilities and then randomly generate text:

Markov chain and language models (off-topic)

- Deep learning language models are super big Markov models
- Brian Hayes (1983, *A progress report on the fine art of turning literature into drivel*), experimented with 0–8th order Markov chains for producing text. For example, using William Faulkner's story "Two Soldiers" to compute 8th order transition probabilities and then randomly generate text:

"Who let you in here?" he said. "Go on, beat it." "Durn that," I said, "They got to have wood and water. I can chop it and tote it. Come on," I said, "Where's Pete?" And he looked jest like Pete first soldier hollered. When he got on the table, he come in. He never come out of my own pocket as a measure of protecting the company against riot and bloodshed. And when he said. "You tell me a bus ticket, let alone write out no case histories. Then the law come back with a knife!"

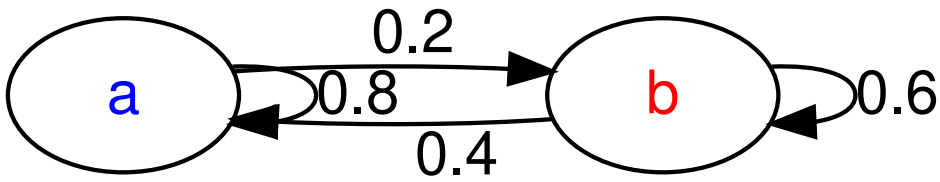
Markov chain and language models (off-topic)

- Deep learning language models are super big Markov models
- Brian Hayes (1983, *A progress report on the fine art of turning literature into drivel*), experimented with 0–8th order Markov chains for producing text. For example, using William Faulkner's story "Two Soldiers" to compute 8th order transition probabilities and then randomly generate text:

  "Who let you in here?" he said. "Go on, beat it." "Durn that," I said, "They got to have wood and water. I can chop it and tote it. Come on," I said, "Where's Pete?" And he looked jest like Pete first soldier hollered. When he got on the table, he come in. He never come out of my own pocket as a measure of protecting the company against riot and bloodshed. And when he said. "You tell me a bus ticket, let alone write out no case histories. Then the law come back with a knife!"
- In Markov chain Monte Carlo we are not interested in the sequences directly, but how likely each state is in a long sequence

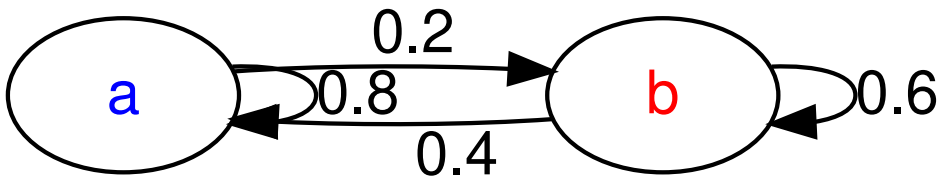
Markov chain

- Example of a simple Markov chain



Markov chain

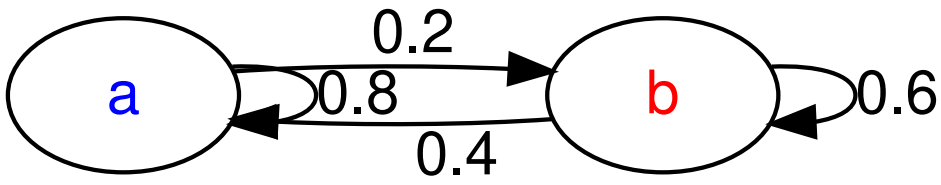
- Example of a simple Markov chain



- Given known transition probabilities, we can simulate the Markov process and count how often each state is visited

Markov chain

- Example of a simple Markov chain

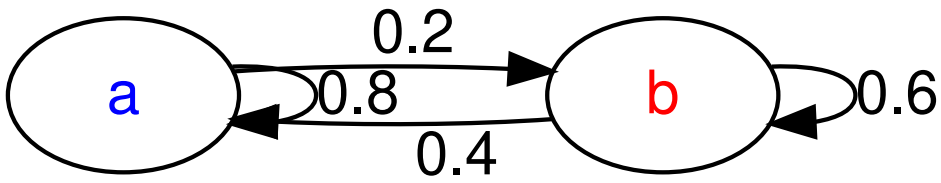


- Given known transition probabilities, we can simulate the Markov process and count how often each state is visited

a a b b a a b b b b b b a a a a a a a a a a a b b a a a a a a a a a b a a a a a b

Markov chain

- Example of a simple Markov chain



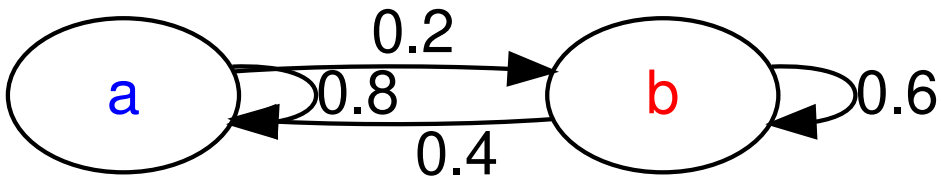
- Given known transition probabilities, we can simulate the Markov process and count how often each state is visited

a a b b a a b b b b b a a a a a a a a a a b b a a a a a a a a a b a a a a a b

$$p(\text{a}) \approx \frac{1}{S} \sum_{s=1}^{S} I(\text{state} = \text{a})$$

Markov chain

- Example of a simple Markov chain



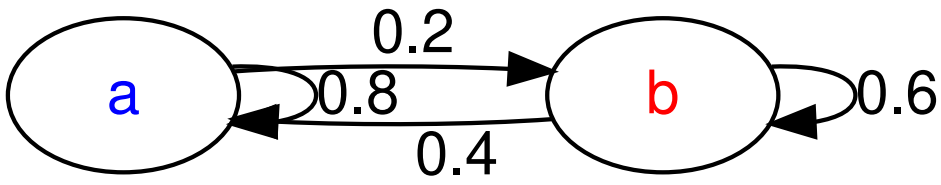
- Given known transition probabilities, we can simulate the Markov process and count how often each state is visited

a a b b a a b b b b b a a a a a a a a a a b b a a a a a a a a a b a a a a a b

$$p(\text{a}) \approx \frac{1}{S} \sum_{s=1}^{S} I(\text{state} = \text{a}) = 0.7$$

Markov chain

- Example of a simple Markov chain



- Given known transition probabilities, we can simulate the Markov process and count how often each state is visited

a a b b a a b b b b b a a a a a a a a a a b b a a a a a a a a a b a a a a a b

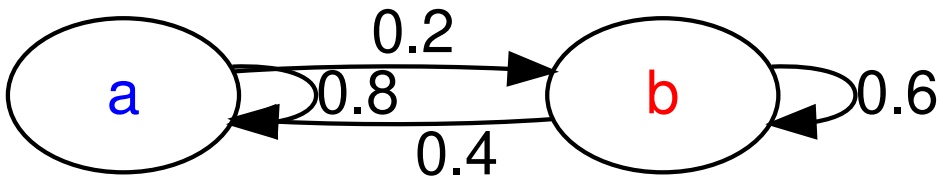
$$p(\text{a}) \approx \frac{1}{S} \sum_{s=1}^{S} I(\text{state} = \text{a}) = 0.7$$

- In discrete case we can also find the marginal probabilities by examining the transition probability matrix

$$A = \begin{pmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix}$$

Markov chain

- Example of a simple Markov chain



- Given known transition probabilities, we can simulate the Markov process and count how often each state is visited

a a b b a a b b b b b a a a a a a a a a a b b a a a a a a a a a b a a a a a b

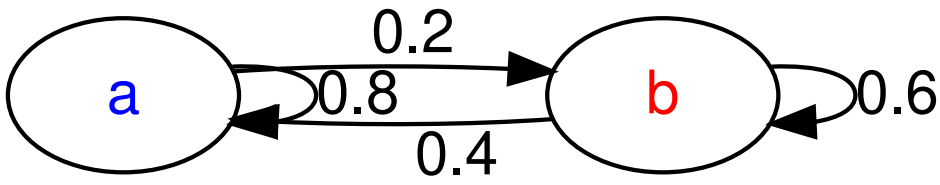
$$p(\text{a}) \approx \frac{1}{S} \sum_{s=1}^{S} I(\text{state} = \text{a}) = 0.7$$

- In discrete case we can also find the marginal probabilities by examining the transition probability matrix

$$A = \begin{pmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix} \quad A^s = \begin{pmatrix} 0.67 & 0.33 \\ 0.67 & 0.33 \end{pmatrix},$$

Markov chain

- Example of a simple Markov chain



- Given known transition probabilities, we can simulate the Markov process and count how often each state is visited

a a b b a a b b b b b a a a a a a a a a a b b a a a a a a a a a b a a a a a b

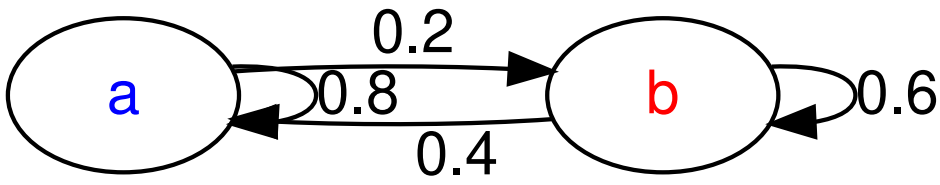
$$p(\text{a}) \approx \frac{1}{S} \sum_{s=1}^{S} I(\text{state} = \text{a}) = 0.7$$

- In discrete case we can also find the marginal probabilities by examining the transition probability matrix

$$A = \begin{pmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix} \quad A^s = \begin{pmatrix} 0.67 & 0.33 \\ 0.67 & 0.33 \end{pmatrix}, \text{ where } S \geq 7 \text{ for 2 digit accuracy}$$

Markov chain

- Example of a simple Markov chain



- Given known transition probabilities, we can simulate the Markov process and count how often each state is visited

a a b b a a b b b b b a a a a a a a a a a b b a a a a a a a a a b a a a a a b

$$p(\text{a}) \approx \frac{1}{S} \sum_{s=1}^{S} I(\text{state} = \text{a}) = 0.7$$

- In discrete case we can also find the marginal probabilities by examining the transition probability matrix

$$A = \begin{pmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix} \quad A^s = \begin{pmatrix} 0.67 & 0.33 \\ 0.67 & 0.33 \end{pmatrix}, \text{ where } S \geq 7 \text{ for 2 digit accuracy}$$

- From A^s we get $p(\text{a}) = 0.67$ and $p(\text{b}) = 0.33$

Markov chain Monte Carlo (MCMC)

- In continuous case, we can't compute the full transition matrix, but we can use conditional transition probabilities to simulate

Markov chain Monte Carlo (MCMC)

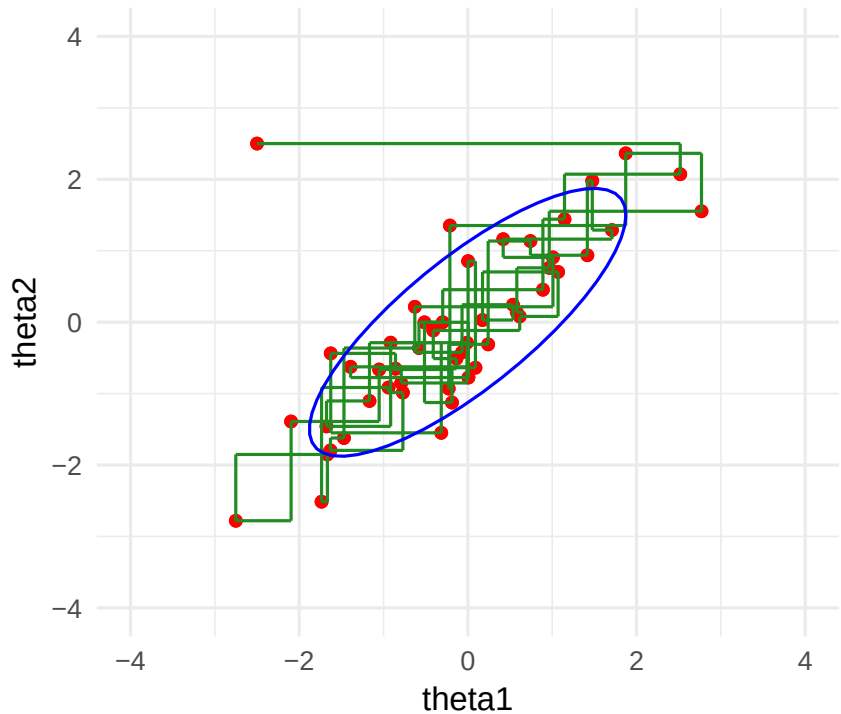
- In continuous case, we can't compute the full transition matrix, but we can use conditional transition probabilities to simulate
- Produce draws $\theta^{(t)}$, given $\theta^{(t-1)}$, from a Markov chain, constructed so that the equilibrium distribution is $p(\theta \mid y)$

Markov chain Monte Carlo (MCMC)

- In continuous case, we can't compute the full transition matrix, but we can use conditional transition probabilities to simulate
- Produce draws $\theta^{(t)}$, given $\theta^{(t-1)}$, from a Markov chain, constructed so that the equilibrium distribution is $p(\theta \mid y)$
  - + generic
  - + combine sequence of easier Monte Carlo draws to form a Markov chain

Markov chain Monte Carlo (MCMC)

- In continuous case, we can't compute the full transition matrix, but we can use conditional transition probabilities to simulate
- Produce draws $\theta^{(t)}$, given $\theta^{(t-1)}$, from a Markov chain, constructed so that the equilibrium distribution is $p(\theta \mid y)$
  - + generic
  - + combine sequence of easier Monte Carlo draws to form a Markov chain
  - + chain goes where most of the posterior mass is
  - + asymptotically chain spends the α% of time where α% posterior mass is



Markov chain Monte Carlo (MCMC)

- In continuous case, we can't compute the full transition matrix, but we can use conditional transition probabilities to simulate
- Produce draws $\theta^{(t)}$, given $\theta^{(t-1)}$, from a Markov chain, constructed so that the equilibrium distribution is $p(\theta \mid y)$
  - + generic
  - + combine sequence of easier Monte Carlo draws to form a Markov chain
  - + chain goes where most of the posterior mass is
  - + asymptotically chain spends the α% of time where α% posterior mass is
  - + central limit theorem holds for expectations (Markov)

Markov chain Monte Carlo (MCMC)

- In continuous case, we can't compute the full transition matrix, but we can use conditional transition probabilities to simulate
- Produce draws $\theta^{(t)}$, given $\theta^{(t-1)}$, from a Markov chain, constructed so that the equilibrium distribution is $p(\theta \mid y)$
  - \+ generic
  - \+ combine sequence of easier Monte Carlo draws to form a Markov chain
  - \+ chain goes where most of the posterior mass is
  - \+ asymptotically chain spends the α% of time where α% posterior mass is
  - \+ central limit theorem holds for expectations (Markov)
  - \- draws are dependent
  - \- construction of efficient Markov chains is not always easy

Markov chain

- Set of random variables $\theta^1, \theta^2, \ldots$, so that with all values of t, θ^t depends only on the previous $\theta^{(t-1)}$

$$p(\theta^t \mid \theta^1, \ldots, \theta^{(t-1)}) = p(\theta^t \mid \theta^{(t-1)})$$

Markov chain

- Set of random variables $\theta^1, \theta^2, \ldots$, so that with all values of t, θ^t depends only on the previous $\theta^{(t-1)}$

$$p(\theta^t \mid \theta^1, \ldots, \theta^{(t-1)}) = p(\theta^t \mid \theta^{(t-1)})$$

- Chain has to be initialized with some starting point θ^0

Markov chain

- Set of random variables $\theta^1, \theta^2, \ldots$, so that with all values of t, θ^t depends only on the previous $\theta^{(t-1)}$

$$p(\theta^t \mid \theta^1, \ldots, \theta^{(t-1)}) = p(\theta^t \mid \theta^{(t-1)})$$

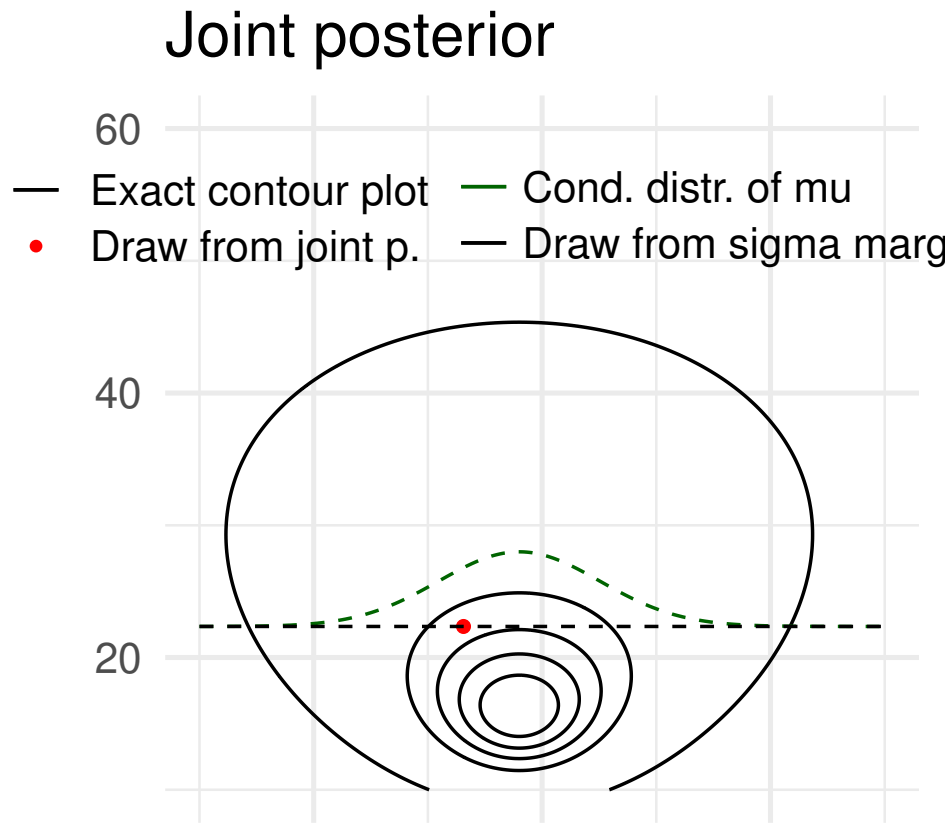
- Chain has to be initialized with some starting point θ^0
- Transition distribution $T_t(\theta^t \mid \theta^{t-1})$ (may depend on t)
  - by choosing a suitable transition distribution, the stationary distribution of Markov chain is $p(\theta \mid y)$

Gibbs sampling

- Alternate sampling from 1D conditional distributions
  - e.g. normal distribution, sample alternating from $p(\mu \mid \sigma^2, y)$ and $p(\sigma^2 \mid \mu, y)$

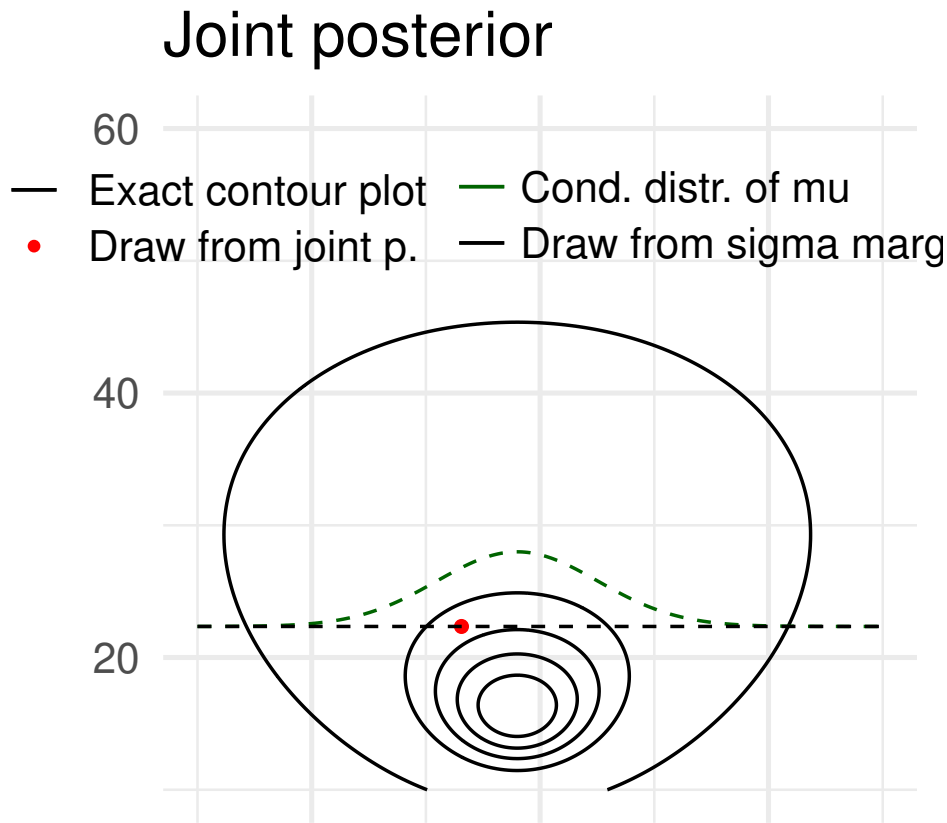
Gibbs sampling

- Alternate sampling from 1D conditional distributions
  - e.g. normal distribution, sample alternating from $p(\mu \mid \sigma^2, y)$ and $p(\sigma^2 \mid \mu, y)$



Gibbs sampling

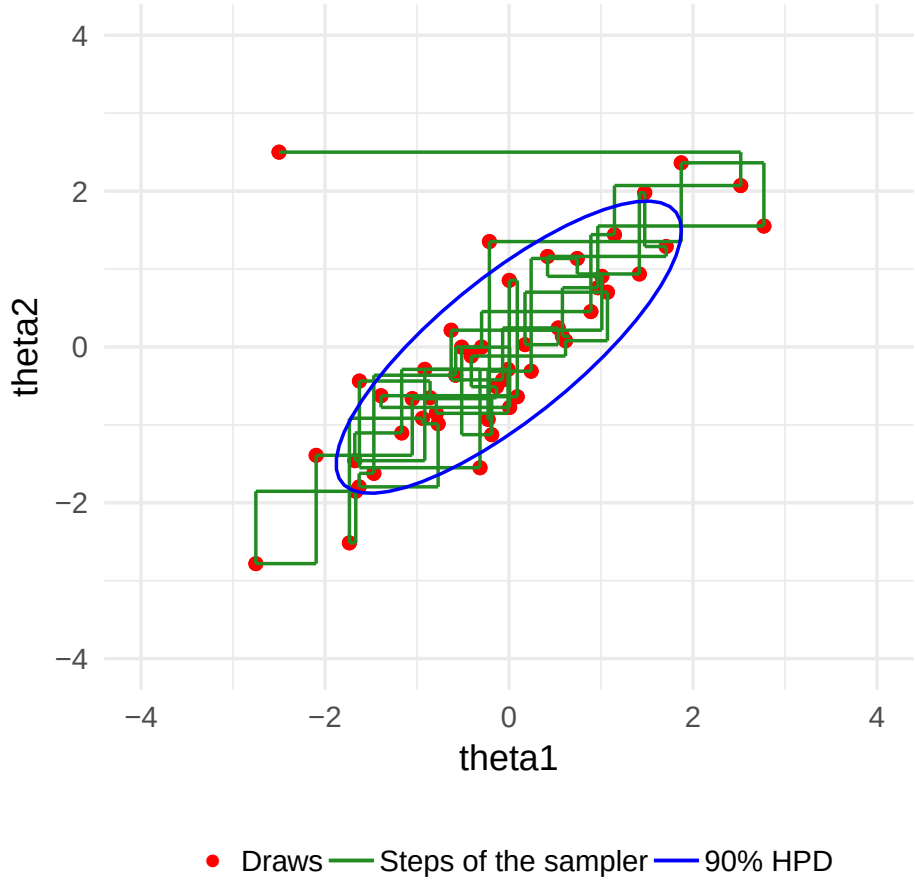
- Alternate sampling from 1D conditional distributions
  - e.g. normal distribution, sample alternating from $p(\mu \mid \sigma^2, y)$ and $p(\sigma^2 \mid \mu, y)$



  - 1D is easy even if no conjugate prior and analytic posterior

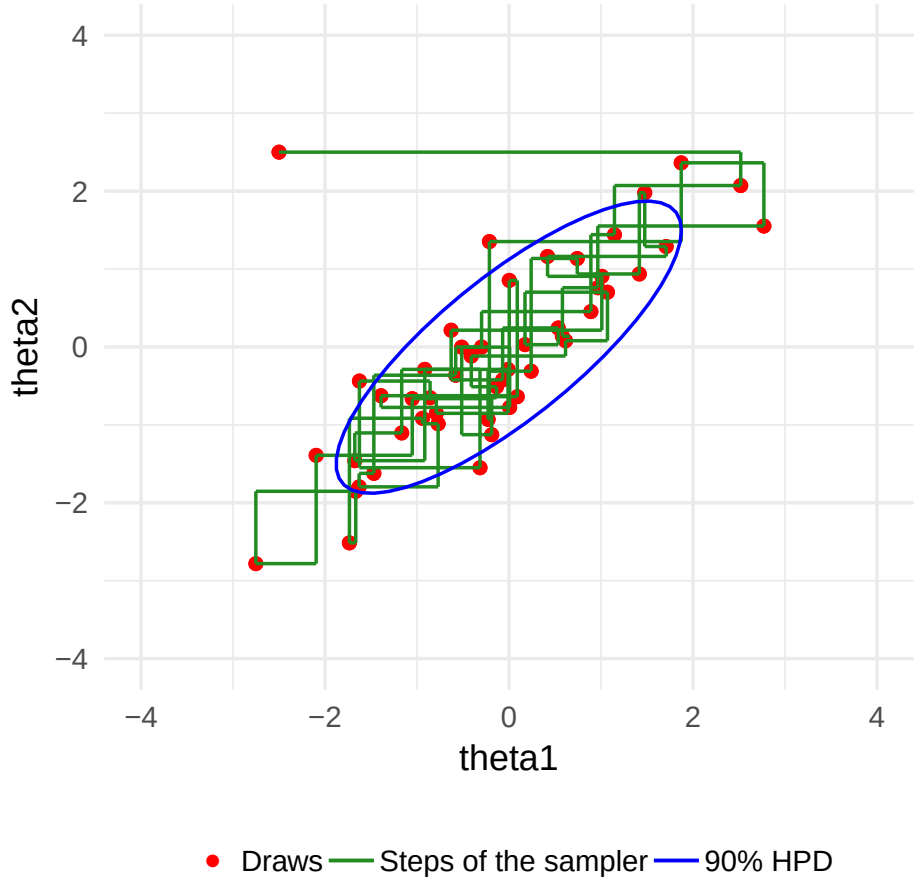
Gibbs sampling

- Alternate sampling from 1D conditional distributions
- demo11_1



Gibbs sampling

- Alternate sampling from 1D conditional distributions
- demo11_1



- Basic algorithm

sample θ_j^t from $p(\theta_j \mid \theta_{-j}^{t-1}, y)$,

where $\theta_{-j}^{t-1} = (\theta_1^t, \ldots, \theta_{j-1}^t, \theta_{j+1}^{t-1}, \ldots, \theta_d^{t-1})$

Gibbs sampling

- With *conditionally* conjugate priors, the sampling from the conditional distributions is easy for wide range of models
  - BUGS/WinBUGS/OpenBUGS/JAGS

Gibbs sampling

- With *conditionally* conjugate priors, the sampling from the conditional distributions is easy for wide range of models
  - BUGS/WinBUGS/OpenBUGS/JAGS
- No algorithm parameters to tune
  (cf. proposal distribution in Metropolis algorithm)

Gibbs sampling

- With *conditionally* conjugate priors, the sampling from the conditional distributions is easy for wide range of models
  - BUGS/WinBUGS/OpenBUGS/JAGS
- No algorithm parameters to tune

  (cf. proposal distribution in Metropolis algorithm)
- For not so easy conditionals, use e.g. inverse-CDF

Gibbs sampling

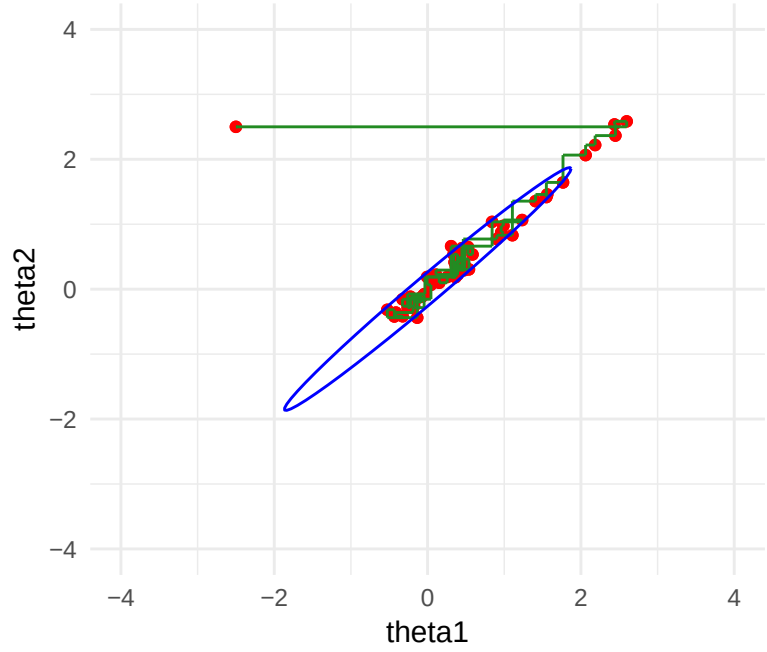
- With *conditionally* conjugate priors, the sampling from the conditional distributions is easy for wide range of models
  - BUGS/WinBUGS/OpenBUGS/JAGS
- No algorithm parameters to tune

  (cf. proposal distribution in Metropolis algorithm)
- For not so easy conditionals, use e.g. inverse-CDF
- Several parameters can be updated in blocks (*blocking*)

Gibbs sampling

- With *conditionally* conjugate priors, the sampling from the conditional distributions is easy for wide range of models
  - BUGS/WinBUGS/OpenBUGS/JAGS
- No algorithm parameters to tune

  (cf. proposal distribution in Metropolis algorithm)
- For not so easy conditionals, use e.g. inverse-CDF
- Several parameters can be updated in blocks (*blocking*)
- Slow if parameters are highly dependent in the posterior
  - demo11_1 continues



Conditional vs joint

- How about sampling θ jointly?
  - e.g. it is easy to sample from multivariate normal

Conditional vs joint

- How about sampling θ jointly?
  - e.g. it is easy to sample from multivariate normal
- Can we use that to form a Markov chain?

Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \ldots$
     (a) pick a proposal θ^* from the proposal distribution $J_t(\theta^* \mid \theta^{t-1})$.
     Proposal distribution has to be symmetric, i.e.
     $J_t(\theta_a \mid \theta_b) = J_t(\theta_b \mid \theta_a)$, for all θ_a, θ_b

Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \ldots$
     (a) pick a proposal θ^* from the proposal distribution $J_t(\theta^* \mid \theta^{t-1})$.
     Proposal distribution has to be symmetric, i.e.
     $J_t(\theta_a \mid \theta_b) = J_t(\theta_b \mid \theta_a)$, for all θ_a, θ_b
     (b) calculate acceptance ratio

$$r = \frac{p(\theta^* \mid y)}{p(\theta^{t-1} \mid y)}$$

Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \ldots$
     (a) pick a proposal θ^* from the proposal distribution $J_t(\theta^* \mid \theta^{t-1})$.
     Proposal distribution has to be symmetric, i.e.
     $J_t(\theta_a \mid \theta_b) = J_t(\theta_b \mid \theta_a)$, for all θ_a, θ_b
     (b) calculate acceptance ratio

$$r = \frac{p(\theta^* \mid y)}{p(\theta^{t-1} \mid y)}$$

     (c) set

$$\theta^t = \begin{cases} \theta^* & \text{with probability } \min(r, 1) \\ \theta^{t-1} & \text{otherwise} \end{cases}$$

Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \ldots$
     (a) pick a proposal θ^* from the proposal distribution $J_t(\theta^* \mid \theta^{t-1})$.
     Proposal distribution has to be symmetric, i.e.
     $J_t(\theta_a \mid \theta_b) = J_t(\theta_b \mid \theta_a)$, for all θ_a, θ_b
     (b) calculate acceptance ratio

$$r = \frac{p(\theta^* \mid y)}{p(\theta^{t-1} \mid y)}$$

     (c) set

$$\theta^t = \begin{cases} \theta^* & \text{with probability } \min(r, 1) \\ \theta^{t-1} & \text{otherwise} \end{cases}$$

     ie, if $p(\theta^* \mid y) > p(\theta^{t-1} \mid y)$ accept the proposal always
     and otherwise accept the proposal with probability r

Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \ldots$
     (a) pick a proposal θ^* from the proposal distribution $J_t(\theta^* \mid \theta^{t-1})$.
     Proposal distribution has to be symmetric, i.e.
     $J_t(\theta_a \mid \theta_b) = J_t(\theta_b \mid \theta_a)$, for all θ_a, θ_b
     (b) calculate acceptance ratio

$$r = \frac{p(\theta^* \mid y)}{p(\theta^{t-1} \mid y)}$$

     (c) set

$$\theta^t = \begin{cases} \theta^* & \text{with probability } \min(r, 1) \\ \theta^{t-1} & \text{otherwise} \end{cases}$$

  - rejection of a proposal increments the time t also by one
    ie, the new state is the same as previous

Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \ldots$
     (a) pick a proposal θ^* from the proposal distribution $J_t(\theta^* \mid \theta^{t-1})$.
     Proposal distribution has to be symmetric, i.e.
     $J_t(\theta_a \mid \theta_b) = J_t(\theta_b \mid \theta_a)$, for all θ_a, θ_b
     (b) calculate acceptance ratio

$$r = \frac{p(\theta^* \mid y)}{p(\theta^{t-1} \mid y)}$$

     (c) set

$$\theta^t = \begin{cases} \theta^* & \text{with probability } \min(r, 1) \\ \theta^{t-1} & \text{otherwise} \end{cases}$$

  - rejection of a proposal increments the time t also by one
    ie, the new state is the same as previous
  - step c is executed by generating a random number from $\mathrm{U}(0, 1)$

Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \ldots$
     (a) pick a proposal θ^* from the proposal distribution $J_t(\theta^* \mid \theta^{t-1})$.
     Proposal distribution has to be symmetric, i.e.
     $J_t(\theta_a \mid \theta_b) = J_t(\theta_b \mid \theta_a)$, for all θ_a, θ_b
     (b) calculate acceptance ratio

$$r = \frac{p(\theta^* \mid y)}{p(\theta^{t-1} \mid y)}$$

     (c) set

$$\theta^t = \begin{cases} \theta^* & \text{with probability } \min(r, 1) \\ \theta^{t-1} & \text{otherwise} \end{cases}$$

  - rejection of a proposal increments the time t also by one
    ie, the new state is the same as previous
  - step c is executed by generating a random number from $\mathrm{U}(0, 1)$
  - $p(\theta^* \mid y)$ and $p(\theta^{t-1} \mid y)$ have the same normalization terms, and thus instead of $p(\cdot \mid y)$, unnormalized $q(\cdot \mid y)$ can be used, as the normalization terms cancel out!

Metropolis algorithm

- Example: one bivariate observation (y_1, y_2)
  - bivariate normal distribution with unknown mean and known covariance
$$\begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix}\Bigg| y \sim \mathrm{N}\left(\begin{pmatrix}y_1\\ y_2\end{pmatrix}, \begin{pmatrix}1 & \rho\\ \rho & 1\end{pmatrix}\right)$$
  - proposal distribution $J_t(\theta^* \mid \theta^{t-1}) = \mathrm{N}(\theta^* \mid \theta^{t-1}, \sigma_p^2)$
  - demo11_2

Metropolis algorithm

- Example: one bivariate observation (y_1, y_2)
  - bivariate normal distribution with unknown mean and known covariance

$$\begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix}\Bigg| y \sim \mathrm{N}\left(\begin{pmatrix}y_1\\ y_2\end{pmatrix}, \begin{pmatrix}1 & \rho\\ \rho & 1\end{pmatrix}\right)$$

  - proposal distribution $J_t(\theta^* \mid \theta^{t-1}) = \mathrm{N}(\theta^* \mid \theta^{t-1}, \sigma_p^2)$
  - demo11_2
- More examples https://chi-feng.github.io/mcmc-demo/

Why Metropolis algorithm works

- Intuitively more draws from the higher density areas as jumps to higher density are always accepted and only some of the jumps to the lower density are accepted

Why Metropolis algorithm works

- Intuitively more draws from the higher density areas as jumps to higher density are always accepted and only some of the jumps to the lower density are accepted

- Theoretically
  1. Prove that simulated series is a Markov chain which has unique stationary distribution
  2. Prove that this stationary distribution is the desired target distribution

Why Metropolis algorithm works

1. Prove that simulated series is a Markov chain which has unique stationary distribution
   a) irreducible
   b) aperiodic
   c) recurrent / not transient

Why Metropolis algorithm works

1. Prove that simulated series is a Markov chain which has unique stationary distribution
   a) irreducible
      = positive probability of eventually reaching any state from any other state
   b) aperiodic
   c) recurrent / not transient

Why Metropolis algorithm works

1. Prove that simulated series is a Markov chain which has unique stationary distribution
   a) irreducible
      = positive probability of eventually reaching any state from any other state
   b) aperiodic
      = aperiodic (return times are not periodic)
      - holds for a random walk on any proper distribution (except for trivial exceptions)
   c) recurrent / not transient

Why Metropolis algorithm works

1. Prove that simulated series is a Markov chain which has unique stationary distribution
   a) irreducible
      = positive probability of eventually reaching any state from any other state
   b) aperiodic
      = aperiodic (return times are not periodic)
      - holds for a random walk on any proper distribution (except for trivial exceptions)
   c) recurrent / not transient
      = probability to return to a state i is 1
      - holds for a random walk on any proper distribution (except for trivial exceptions)

Why Metropolis algorithm works

2. Prove that this stationary distribution is the desired target distribution $p(\theta \mid y)$
   - consider starting algorithm at time $t-1$ with a draw $\theta^{t-1} \sim p(\theta \mid y)$

Why Metropolis algorithm works

2. Prove that this stationary distribution is the desired target distribution $p(\theta \mid y)$
   - consider starting algorithm at time $t-1$ with a draw $\theta^{t-1} \sim p(\theta \mid y)$
   - consider any two such points θ_a and θ_b drawn from $p(\theta \mid y)$ and labeled so that $p(\theta_b \mid y) \geq p(\theta_a \mid y)$

Why Metropolis algorithm works

2. Prove that this stationary distribution is the desired target distribution $p(\theta \mid y)$
   - consider starting algorithm at time $t-1$ with a draw $\theta^{t-1} \sim p(\theta \mid y)$
   - consider any two such points θ_a and θ_b drawn from $p(\theta \mid y)$ and labeled so that $p(\theta_b \mid y) \geq p(\theta_a \mid y)$
   - the unconditional probability density of a transition from θ_a to θ_b is

$$p(\theta^{t-1} = \theta_a, \theta^t = \theta_b) = p(\theta_a \mid y) J_t(\theta_b \mid \theta_a),$$

Why Metropolis algorithm works

2. Prove that this stationary distribution is the desired target distribution $p(\theta \mid y)$
   - consider starting algorithm at time $t-1$ with a draw $\theta^{t-1} \sim p(\theta \mid y)$
   - consider any two such points θ_a and θ_b drawn from $p(\theta \mid y)$ and labeled so that $p(\theta_b \mid y) \geq p(\theta_a \mid y)$
   - the unconditional probability density of a transition from θ_a to θ_b is
   $$p(\theta^{t-1} = \theta_a, \theta^t = \theta_b) = p(\theta_a \mid y) J_t(\theta_b \mid \theta_a),$$
   - the unconditional probability density of a transition from θ_b to θ_a is
   $$\begin{aligned} p(\theta^{t-1} = \theta_b, \theta^t = \theta_a) &= p(\theta_b \mid y) J_t(\theta_a \mid \theta_b) \left(\frac{p(\theta_a \mid y)}{p(\theta_b \mid y)} \right) \\ &= p(\theta_a \mid y) J_t(\theta_a \mid \theta_b), \end{aligned}$$

Why Metropolis algorithm works

2. Prove that this stationary distribution is the desired target distribution $p(\theta \mid y)$
   - consider starting algorithm at time $t-1$ with a draw $\theta^{t-1} \sim p(\theta \mid y)$
   - consider any two such points θ_a and θ_b drawn from $p(\theta \mid y)$ and labeled so that $p(\theta_b \mid y) \geq p(\theta_a \mid y)$
   - the unconditional probability density of a transition from θ_a to θ_b is
   $$p(\theta^{t-1} = \theta_a, \theta^t = \theta_b) = p(\theta_a \mid y) J_t(\theta_b \mid \theta_a),$$
   - the unconditional probability density of a transition from θ_b to θ_a is
   $$\begin{aligned} p(\theta^{t-1} = \theta_b, \theta^t = \theta_a) &= p(\theta_b \mid y) J_t(\theta_a \mid \theta_b) \left(\frac{p(\theta_a \mid y)}{p(\theta_b \mid y)} \right) \\ &= p(\theta_a \mid y) J_t(\theta_a \mid \theta_b), \end{aligned}$$

   which is the same as the probability of transition from θ_a to θ_b, since we have required that $J_t(\cdot \mid \cdot)$ is symmetric

Why Metropolis algorithm works

2. Prove that this stationary distribution is the desired target distribution $p(\theta \mid y)$
   - consider starting algorithm at time $t-1$ with a draw $\theta^{t-1} \sim p(\theta \mid y)$
   - consider any two such points θ_a and θ_b drawn from $p(\theta \mid y)$ and labeled so that $p(\theta_b \mid y) \geq p(\theta_a \mid y)$
   - the unconditional probability density of a transition from θ_a to θ_b is
   $$p(\theta^{t-1} = \theta_a, \theta^t = \theta_b) = p(\theta_a \mid y) J_t(\theta_b \mid \theta_a),$$
   - the unconditional probability density of a transition from θ_b to θ_a is
   $$\begin{aligned} p(\theta^{t-1} = \theta_b, \theta^t = \theta_a) &= p(\theta_b \mid y) J_t(\theta_a \mid \theta_b) \left(\frac{p(\theta_a \mid y)}{p(\theta_b \mid y)} \right) \\ &= p(\theta_a \mid y) J_t(\theta_a \mid \theta_b), \end{aligned}$$
   which is the same as the probability of transition from θ_a to θ_b, since we have required that $J_t(\cdot \mid \cdot)$ is symmetric
   - since their joint distribution is symmetric, θ^{t-1} and θ^t have the same marginal distributions, and so $p(\theta \mid y)$ is the stationary distribution of the Markov chain of θ

Metropolis-Hastings algorithm

- Generalization of Metropolis algorithm for non-symmetric proposal distributions
  - acceptance ratio includes ratio of proposal distributions

$$r = \frac{p(\theta^* \mid y)/J_t(\theta^* \mid \theta^{t-1})}{p(\theta^{t-1} \mid y)/J_t(\theta^{t-1} \mid \theta^*)}$$

Metropolis-Hastings algorithm

- Generalization of Metropolis algorithm for non-symmetric proposal distributions
  - acceptance ratio includes ratio of proposal distributions

$$r = \frac{p(\theta^* \mid y)/J_t(\theta^* \mid \theta^{t-1})}{p(\theta^{t-1} \mid y)/J_t(\theta^{t-1} \mid \theta^*)} = \frac{p(\theta^* \mid y)J_t(\theta^{t-1} \mid \theta^*)}{p(\theta^{t-1} \mid y)J_t(\theta^* \mid \theta^{t-1})}$$

Metropolis-Hastings algorithm

- Ideal proposal distribution is the distribution itself
  - $J(\theta^* \mid \theta) \equiv p(\theta^* \mid y)$ for all θ
  - acceptance probability is 1
  - independent draws
  - not usually feasible

Metropolis-Hastings algorithm

- Ideal proposal distribution is the distribution itself
  - $J(\theta^* \mid \theta) \equiv p(\theta^* \mid y)$ for all θ
  - acceptance probability is 1
  - independent draws
  - not usually feasible
- Good proposal distribution resembles the target distribution
  - if the shape of the target distribution is unknown, usually normal or t distribution is used

Metropolis-Hastings algorithm

- Ideal proposal distribution is the distribution itself
  - $J(\theta^* \mid \theta) \equiv p(\theta^* \mid y)$ for all θ
  - acceptance probability is 1
  - independent draws
  - not usually feasible
- Good proposal distribution resembles the target distribution
  - if the shape of the target distribution is unknown, usually normal or t distribution is used
- After the shape has been selected, it is important to select the scale
  - small scale
    $\rightarrow$ many steps accepted, but the chain moves slowly due to small steps
  - big scale
    $\rightarrow$ long steps proposed, but many of those rejected and again chain moves slowly

Metropolis-Hastings algorithm

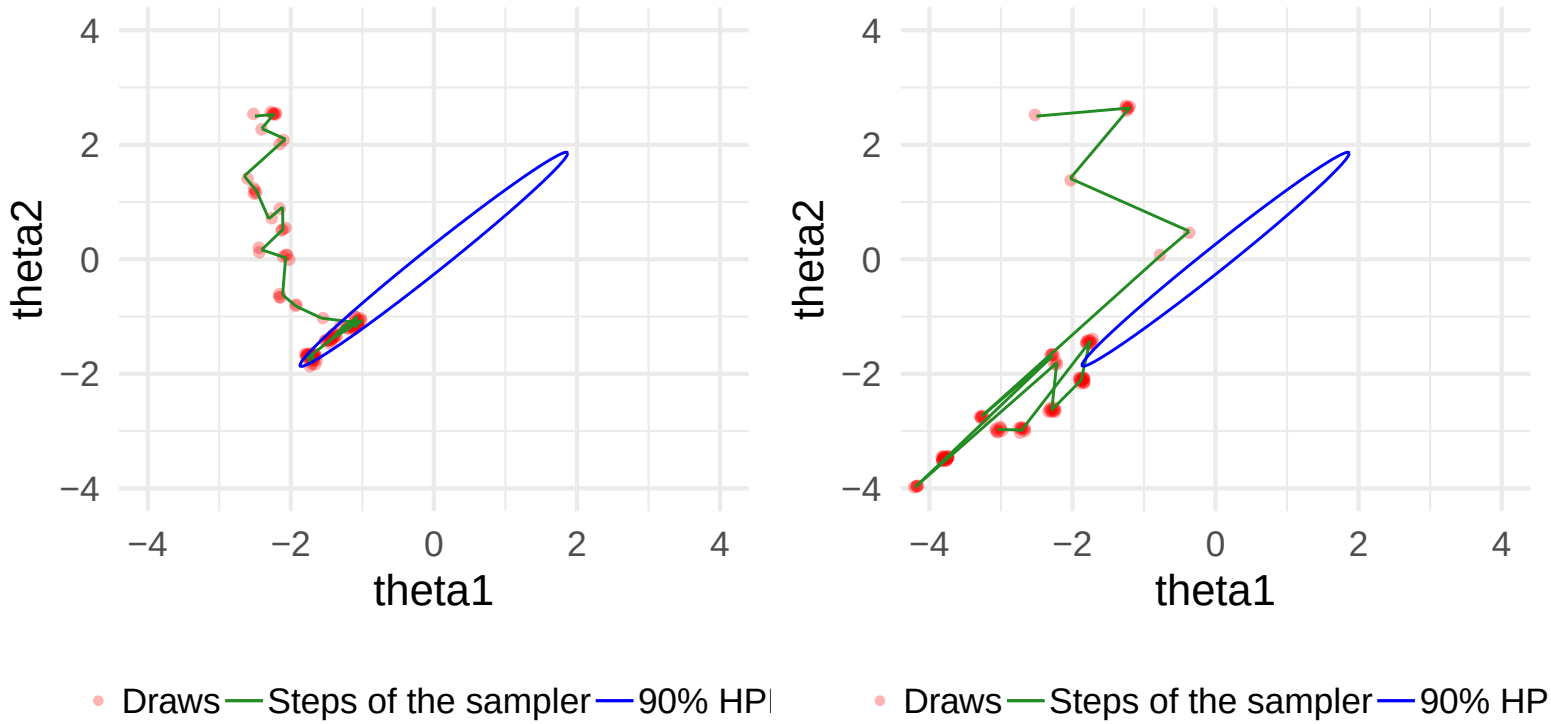
- Ideal proposal distribution is the distribution itself
  - $J(\theta^* \mid \theta) \equiv p(\theta^* \mid y)$ for all θ
  - acceptance probability is 1
  - independent draws
  - not usually feasible
- Good proposal distribution resembles the target distribution
  - if the shape of the target distribution is unknown, usually normal or t distribution is used
- After the shape has been selected, it is important to select the scale
  - small scale
    $\rightarrow$ many steps accepted, but the chain moves slowly due to small steps
  - big scale
    $\rightarrow$ long steps proposed, but many of those rejected and again chain moves slowly
- Generic rule for rejection rate is 60-90% (but depends on dimensionality and a specific algorithm variation)

Gibbs sampling

- Specific case of Metropolis-Hastings algorithm
  - single updated (or blocked)
  - proposal distribution is the conditional distribution
    $\rightarrow$ proposal and target distributions are same
    $\rightarrow$ acceptance probability is 1

Metropolis

- Usually doesn't scale well to high dimensions
  - if the shape doesn't match the whole distribution, the efficiency drops
  - demo11_2



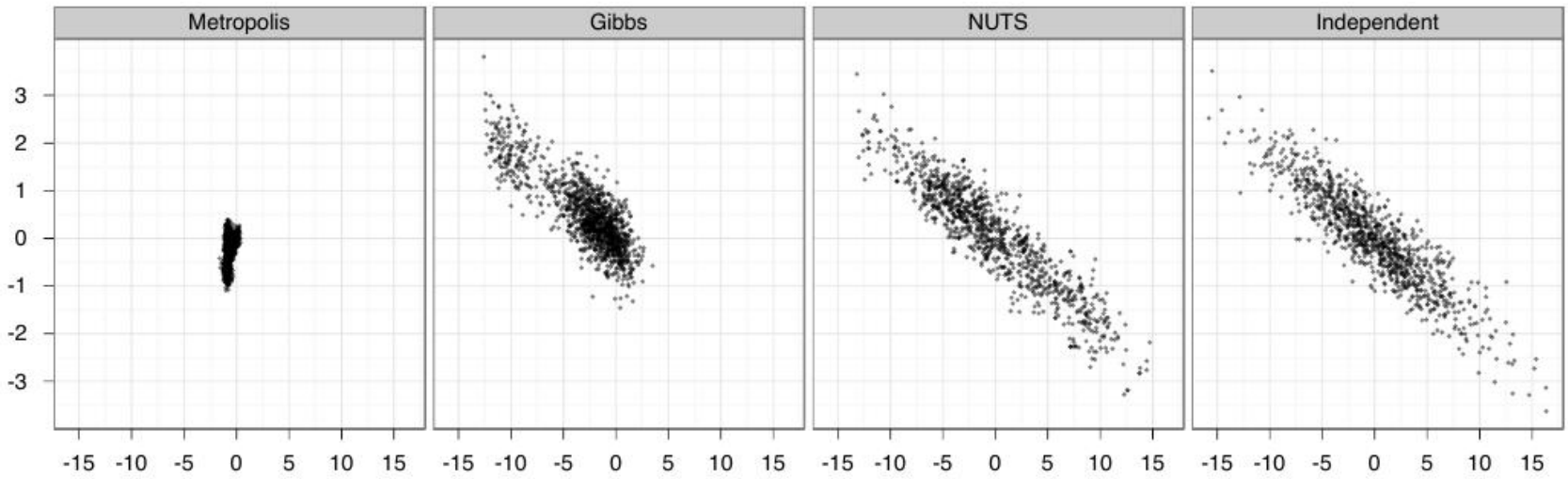
Dynamic Hamiltonian Monte Carlo and NUTS

- Chapter 12 presents some more advanced methods
  - Chapter 12 includes Hamiltonian Monte Carlo and NUTS, which is one of the most efficient methods
    - uses gradient information
    - Hamiltonian dynamic simulation reduces random walk
    - state-of-the-art MCMC used by most modern probabilistic programming frameworks
- More next week

HMC / NUTS

Comparison of algorithms on **highly correlated** 250-dimensional Gaussian distribution

- Do **1,000,000** draws with both Random Walk Metropolis and Gibbs, thinning by 1000
- Do **1,000** draws using Stan's NUTS algorithm (no thinning)
- Do 1,000 independent draws (we can do this for multivariate normal)



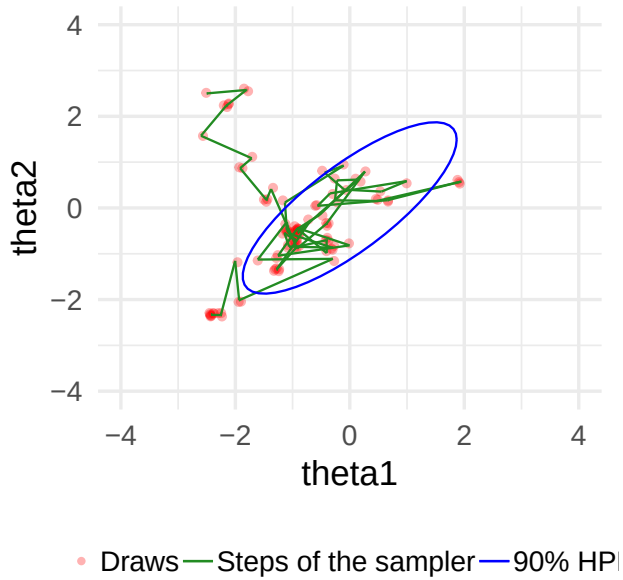
from Hoffman & Gelman (2014)

Warm-up and convergence diagnostics

- Asymptotically chain spends the α% of time where α% posterior mass is

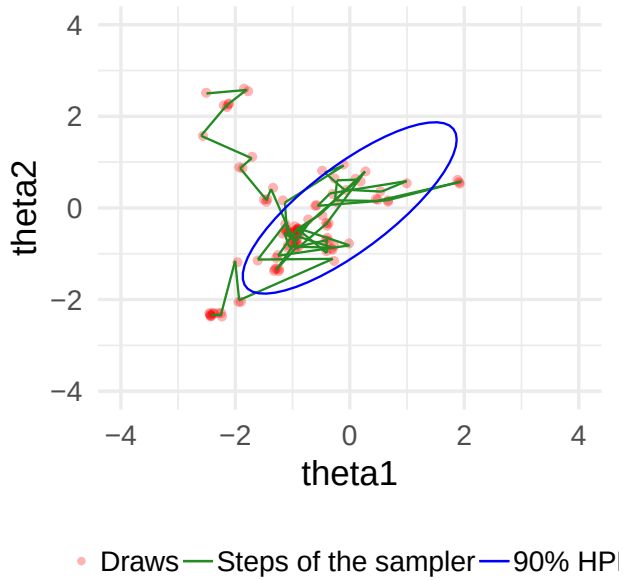
Warm-up and convergence diagnostics

- Asymptotically chain spends the α% of time where α% posterior mass is
  - but in finite time the initial part of the chain may be non-representative and lower error of the estimate can be obtained by throwing it away



Warm-up and convergence diagnostics

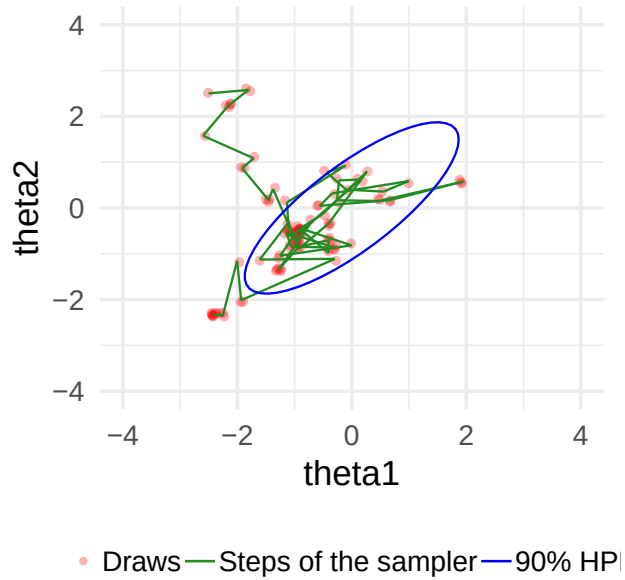
- Asymptotically chain spends the α% of time where α% posterior mass is
  - but in finite time the initial part of the chain may be non-representative and lower error of the estimate can be obtained by throwing it away



- Warm-up = remove draws from the beginning of the chain
  - warm-up may include also phase for adapting algorithm parameters

Warm-up and convergence diagnostics

- Asymptotically chain spends the α% of time where α% posterior mass is
  - but in finite time the initial part of the chain may be non-representative and lower error of the estimate can be obtained by throwing it away



- Warm-up = remove draws from the beginning of the chain
  - warm-up may include also phase for adapting algorithm parameters
- Convergence diagnostics
  - Is the sample representative of the target distribution?

MCMC draws are dependent

- Monte Carlo estimates still valid
  (central limit theorem holds as proved by Andrey Markov)

$$E_{p(\theta|y)}[f(\theta)] \approx \frac{1}{S}\sum_{s=1}^{S} f(\theta^{(s)})$$

- Estimation of Monte Carlo error is more difficult
  - dependency (due to the Markov process) reduces the efficiency

MCMC draws are dependent

- Monte Carlo estimates still valid
  (central limit theorem holds as proved by Andrey Markov)

$$E_{p(\theta|y)}[f(\theta)] \approx \frac{1}{S}\sum_{s=1}^{S} f(\theta^{(s)})$$

- Estimation of Monte Carlo error is more difficult
  - dependency (due to the Markov process) reduces the efficiency
  - evaluation of *effective* sample size (ESS)

MCMC draws are dependent

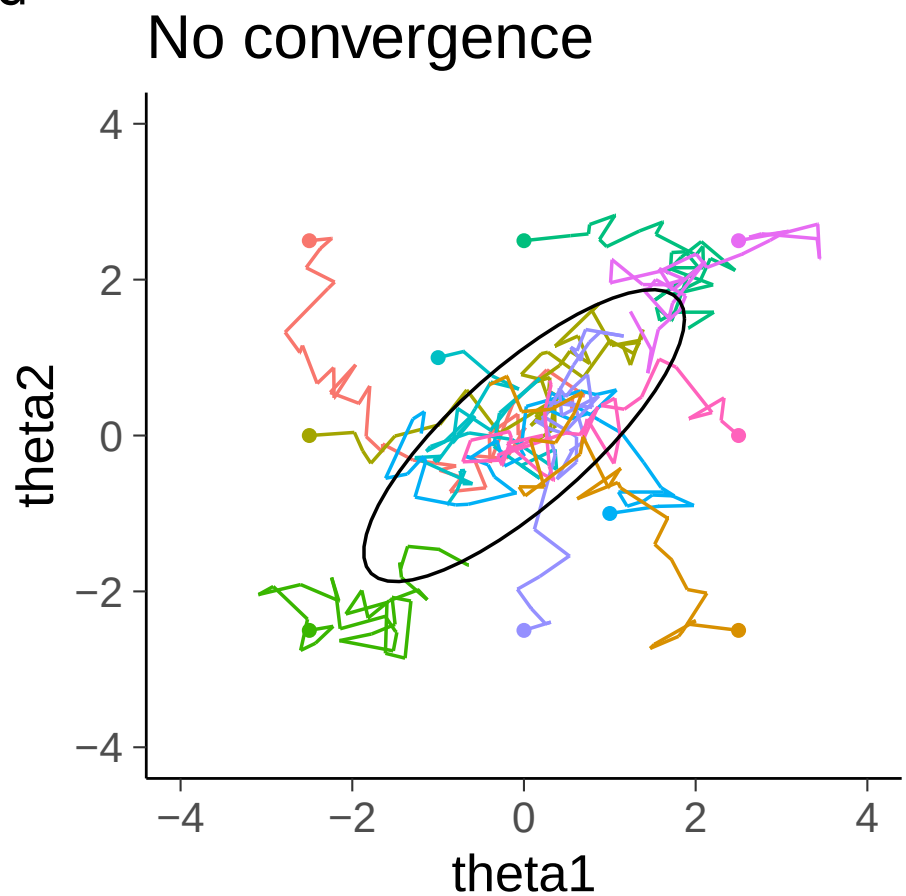
- Monte Carlo estimates still valid
  (central limit theorem holds as proved by Andrey Markov)

$$E_{p(\theta|y)}[f(\theta)] \approx \frac{1}{S}\sum_{s=1}^{S} f(\theta^{(s)})$$

- Estimation of Monte Carlo error is more difficult
  - dependency (due to the Markov process) reduces the efficiency
  - evaluation of *effective* sample size (ESS)
  - given finite variance, the distribution of the expectation approaches normal distribution with variance $\sigma_\theta^2/\text{ESS}$

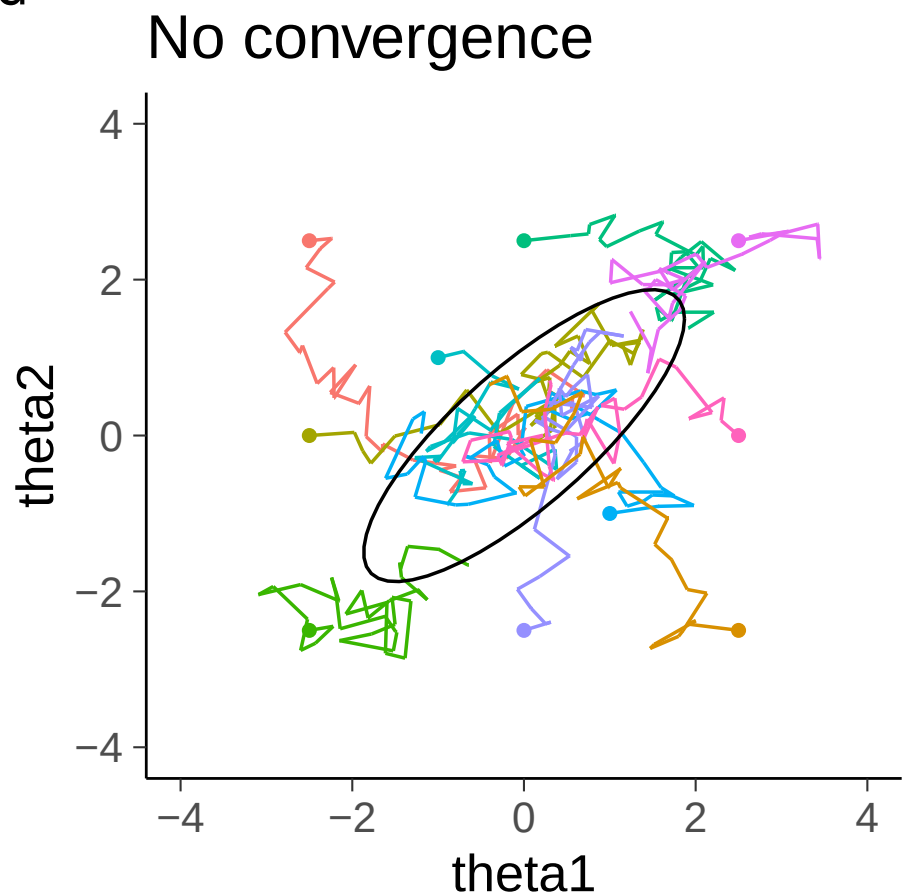
Several chains

- Use of several chains make convergence diagnostics easier
- Start chains from different starting points – preferably overdispersed



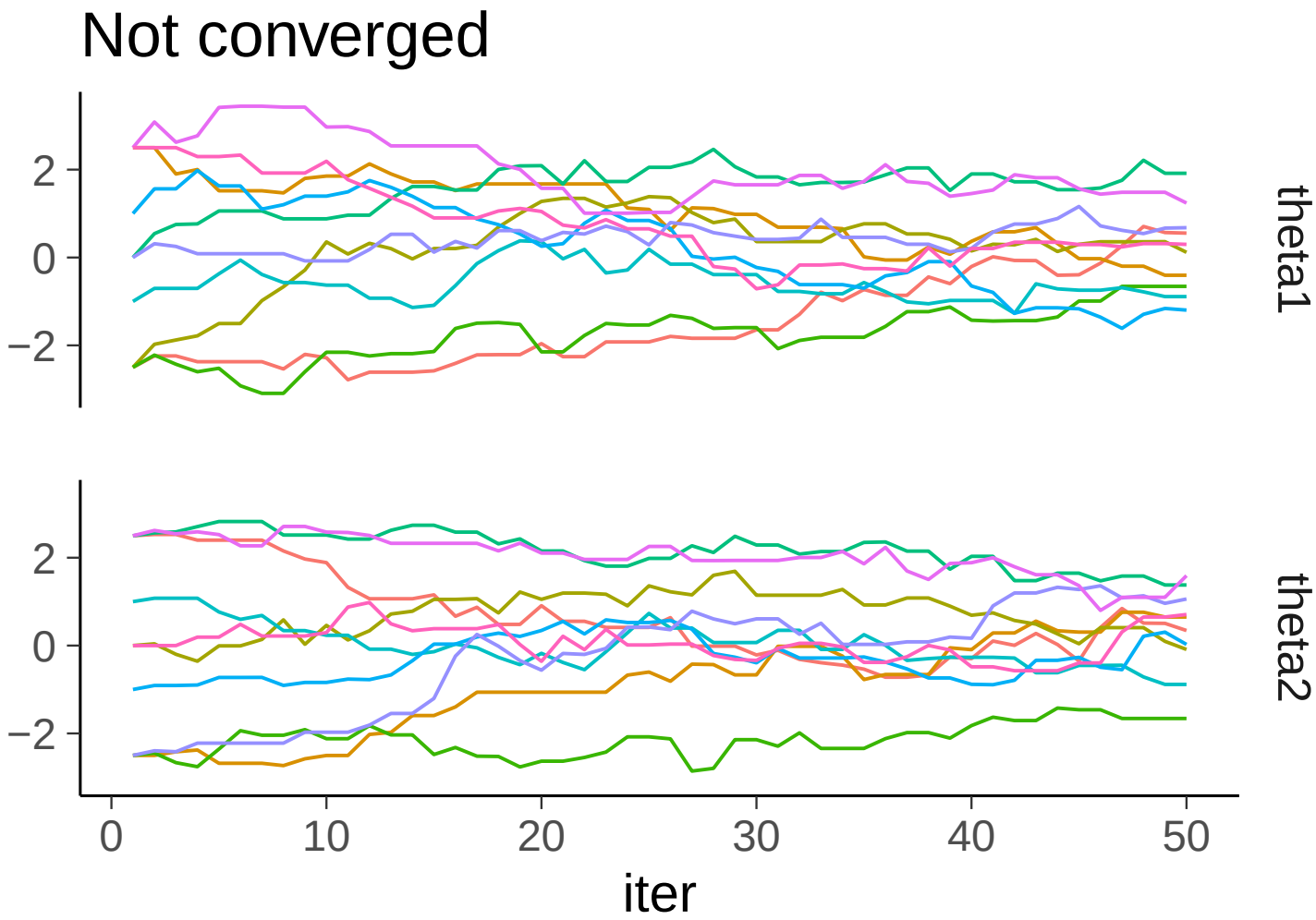
Several chains

- Use of several chains make convergence diagnostics easier
- Start chains from different starting points – preferably overdispersed

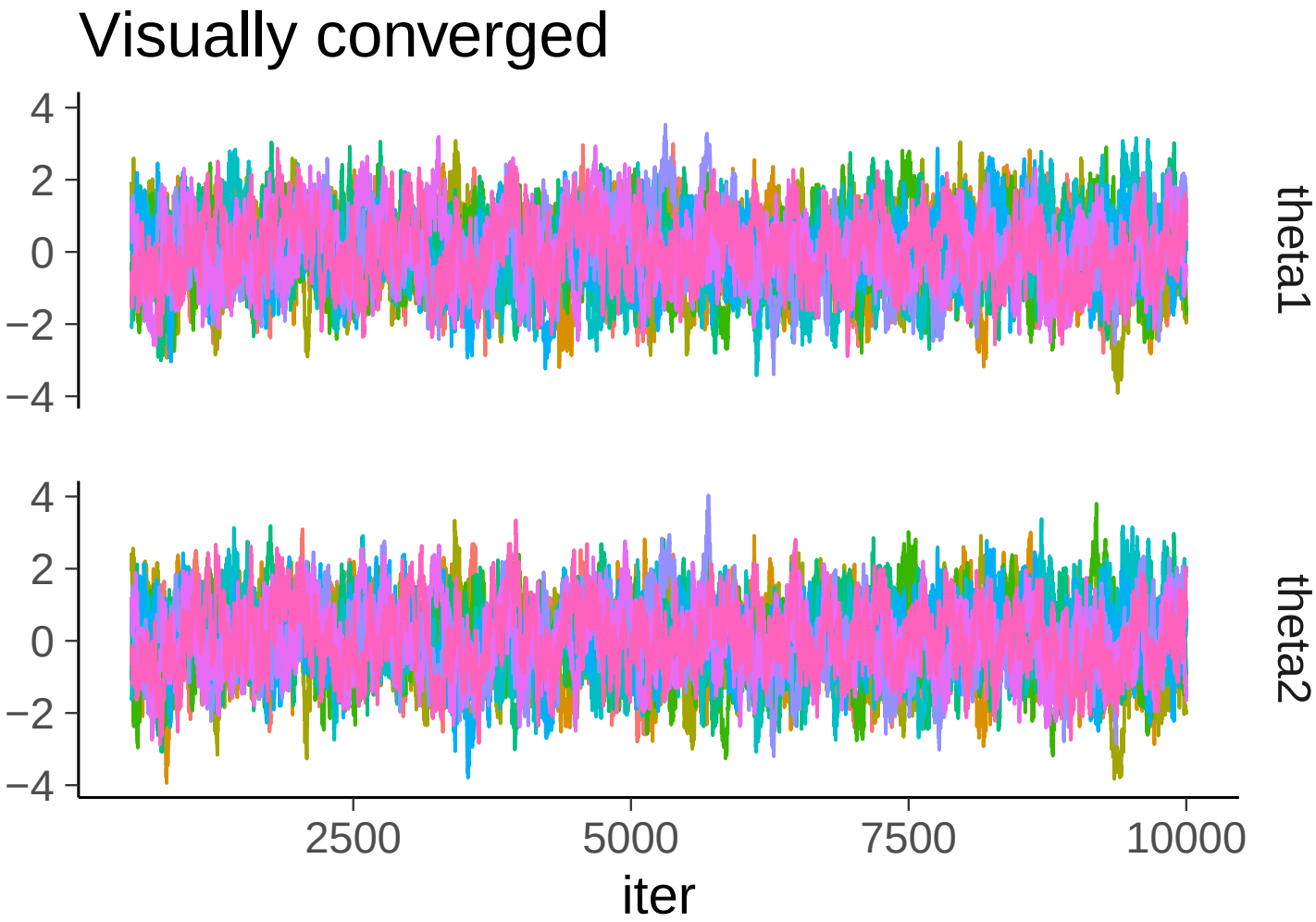


- Remove draws from the beginning of the chains and run chains long enough so that it is not possible to distinguish where each chain started and the chains are well mixed

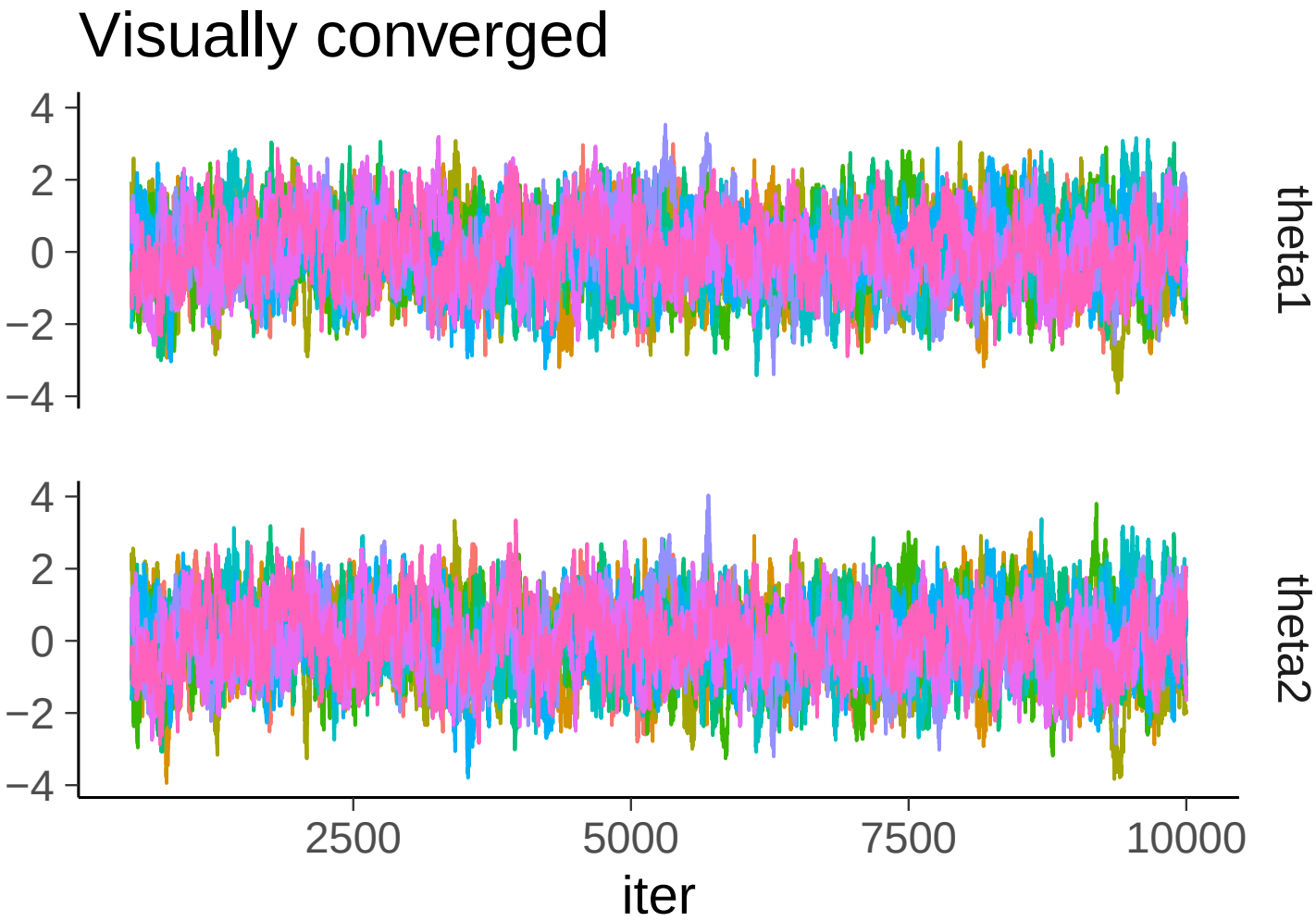
Several chains



Several chains



Several chains



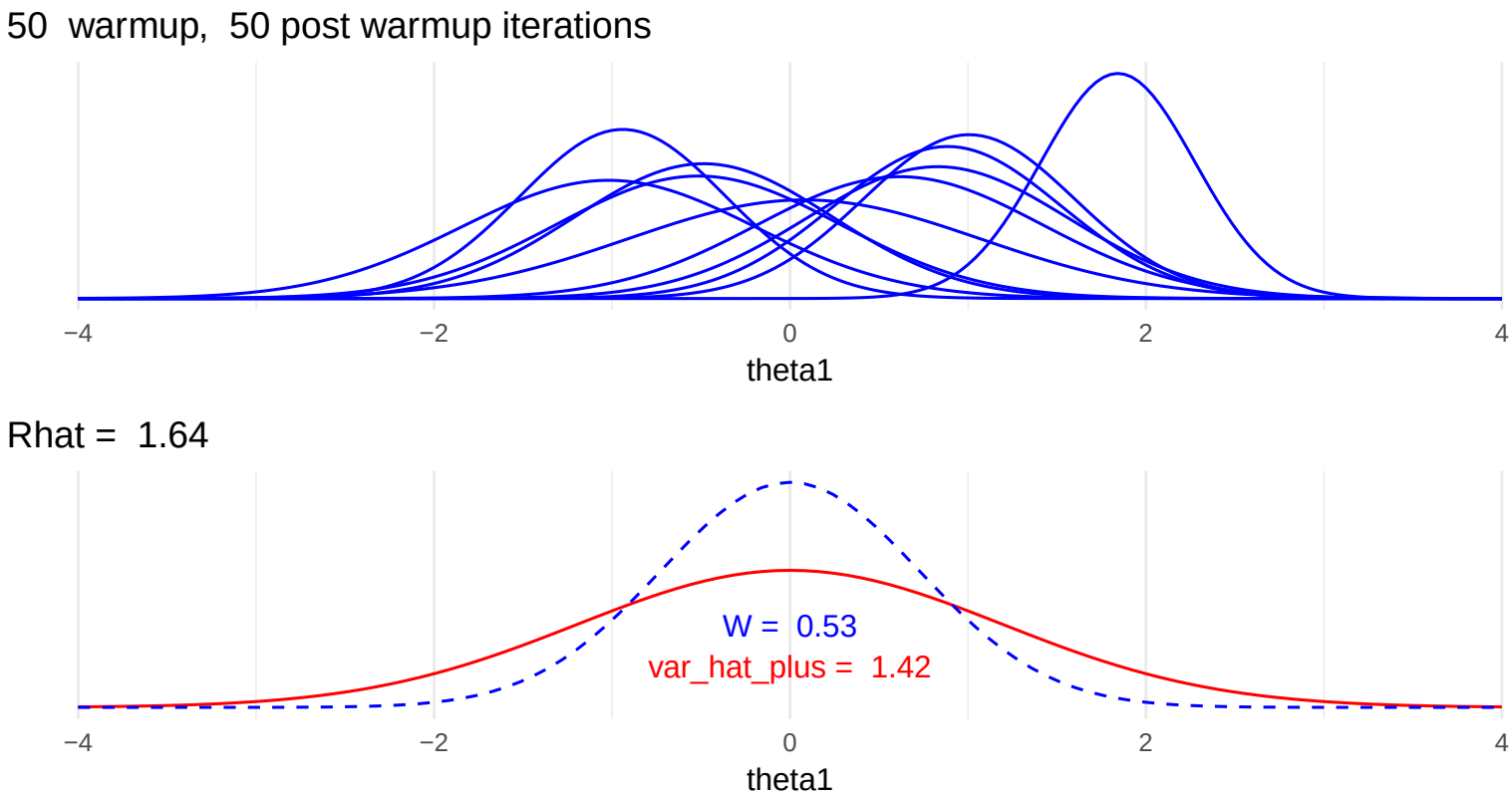
Visual convergence check is not sufficient

$\widehat{R}$: comparison of within and between variances of the chains

- BDA3: $\widehat{R}$ aka *potential scale reduction factor* (PSRF)
- Compare means and variances of the chains

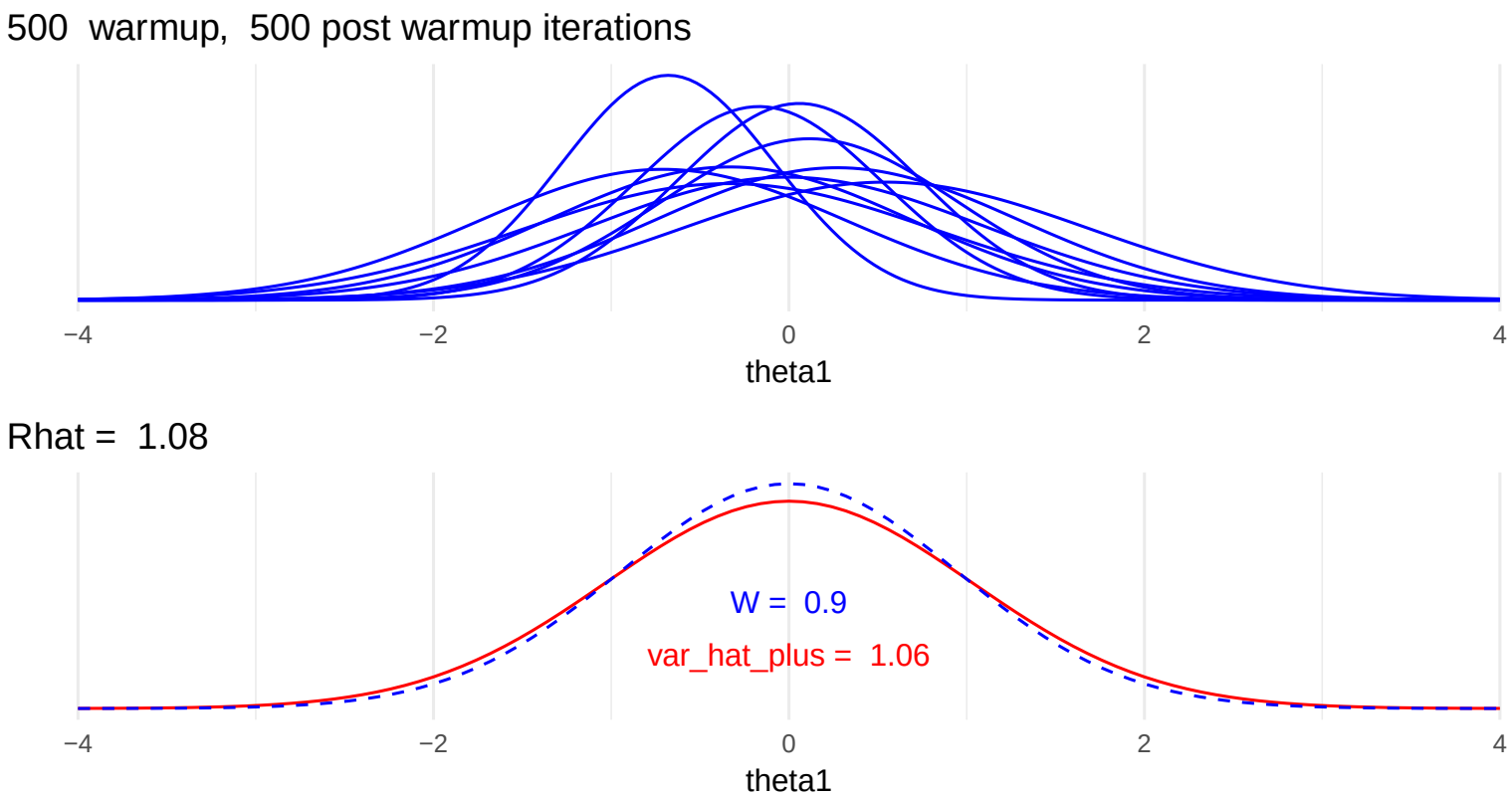
$\widehat{R}$: comparison of within and between variances of the chains

- BDA3: $\widehat{R}$ aka *potential scale reduction factor* (PSRF)
- Compare means and variances of the chains
  W = within chain variance estimate
  var_hat_plus = total variance estimate



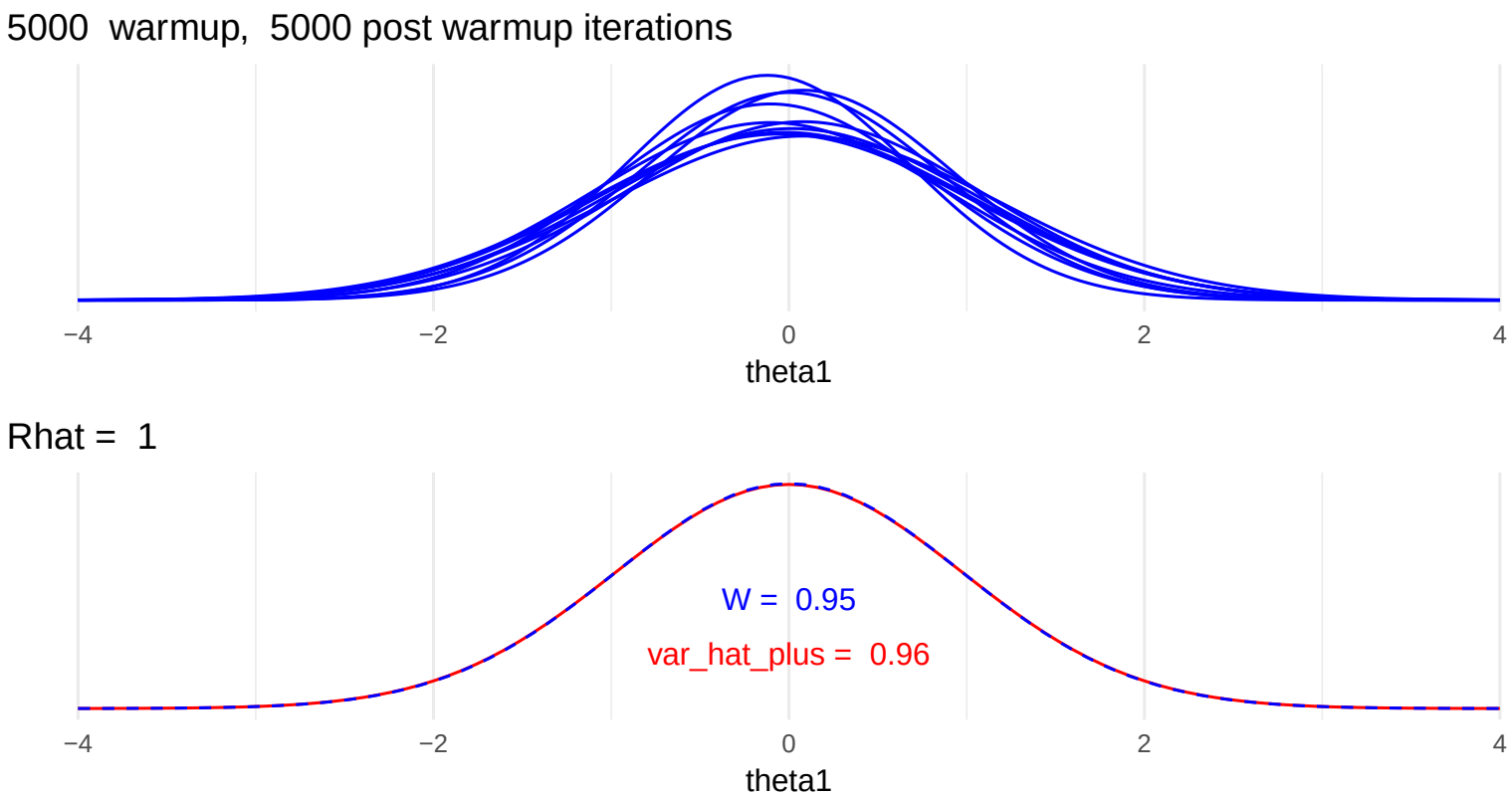
$\widehat{R}$: comparison of within and between variances of the chains

- BDA3: $\widehat{R}$ aka *potential scale reduction factor* (PSRF)
- Compare means and variances of the chains
  W = within chain variance estimate
  var_hat_plus = total variance estimate



$\widehat{R}$: comparison of within and between variances of the chains

- BDA3: $\widehat{R}$ aka *potential scale reduction factor* (PSRF)
- Compare means and variances of the chains
  W = within chain variance estimate
  var_hat_plus = total variance estimate



- M chains, each having N draws (with new $\widehat{R}$ notation)

- M chains, each having N draws (with new $\widehat{R}$ notation)
- Within chains variance W

$$W = \frac{1}{M}\sum_{m=1}^{M} s_m^2, \text{ where } s_m^2 = \frac{1}{N-1}\sum_{n=1}^{N}(\theta_{nm} - \bar{\theta}_{.m})^2$$

- M chains, each having N draws (with new $\widehat{R}$ notation)
- Within chains variance W

$$W = \frac{1}{M}\sum_{m=1}^{M} s_m^2, \text{ where } s_m^2 = \frac{1}{N-1}\sum_{n=1}^{N}(\theta_{nm} - \bar{\theta}_{.m})^2$$

- Between chains variance B

$$B = \frac{N}{M-1}\sum_{m=1}^{M}(\bar{\theta}_{.m} - \bar{\theta}_{..})^2,$$

$$\text{where } \bar{\theta}_{.m} = \frac{1}{N}\sum_{n=1}^{N}\theta_{nm},\ \bar{\theta}_{..} = \frac{1}{M}\sum_{m=1}^{M}\bar{\theta}_{.m}$$

- M chains, each having N draws (with new $\widehat{R}$ notation)
- Within chains variance W

$$W = \frac{1}{M}\sum_{m=1}^{M} s_m^2, \text{ where } s_m^2 = \frac{1}{N-1}\sum_{n=1}^{N}(\theta_{nm} - \bar{\theta}_{.m})^2$$

- Between chains variance B

$$B = \frac{N}{M-1}\sum_{m=1}^{M}(\bar{\theta}_{.m} - \bar{\theta}_{..})^2,$$

$$\text{where } \bar{\theta}_{.m} = \frac{1}{N}\sum_{n=1}^{N}\theta_{nm},\ \bar{\theta}_{..} = \frac{1}{M}\sum_{m=1}^{M}\bar{\theta}_{.m}$$

  - B/N is variance of the means of the chains

- M chains, each having N draws (with new $\widehat{R}$ notation)
- Within chains variance W

$$W = \frac{1}{M}\sum_{m=1}^{M} s_m^2, \text{ where } s_m^2 = \frac{1}{N-1}\sum_{n=1}^{N}(\theta_{nm} - \bar{\theta}_{.m})^2$$

- Between chains variance B

$$B = \frac{N}{M-1}\sum_{m=1}^{M}(\bar{\theta}_{.m} - \bar{\theta}_{..})^2,$$

$$\text{where } \bar{\theta}_{.m} = \frac{1}{N}\sum_{n=1}^{N}\theta_{nm},\ \bar{\theta}_{..} = \frac{1}{M}\sum_{m=1}^{M}\bar{\theta}_{.m}$$

  - B/N is variance of the means of the chains

- Estimate total variance $\text{var}(\theta \mid y)$ as a weighted mean of W and B

$$\widehat{\text{var}}^{+}(\theta \mid y) = \frac{N-1}{N}W + \frac{1}{N}B$$

- Estimate total variance $\mathrm{var}(\theta \mid y)$ as a weighted mean of W and B

$$\widehat{\mathrm{var}}^{+}(\theta \mid y) = \frac{N-1}{N} W + \frac{1}{N} B$$

  - this overestimates marginal posterior variance if the starting points are overdispersed

- Estimate total variance $\mathrm{var}(\theta \mid y)$ as a weighted mean of W and B

$$\widehat{\mathrm{var}}^{+}(\theta \mid y) = \frac{N-1}{N} W + \frac{1}{N} B$$

  - this overestimates marginal posterior variance if the starting points are overdispersed

- Given finite N, W underestimates marginal posterior variance
  - single chains have not yet visited all points in the distribution
  - when $N \to \infty, \quad \mathrm{E}(W) \to \mathrm{var}(\theta \mid y)$

- Estimate total variance $\mathrm{var}(\theta \mid y)$ as a weighted mean of W and B

$$\widehat{\mathrm{var}}^{+}(\theta \mid y) = \frac{N-1}{N} W + \frac{1}{N} B$$

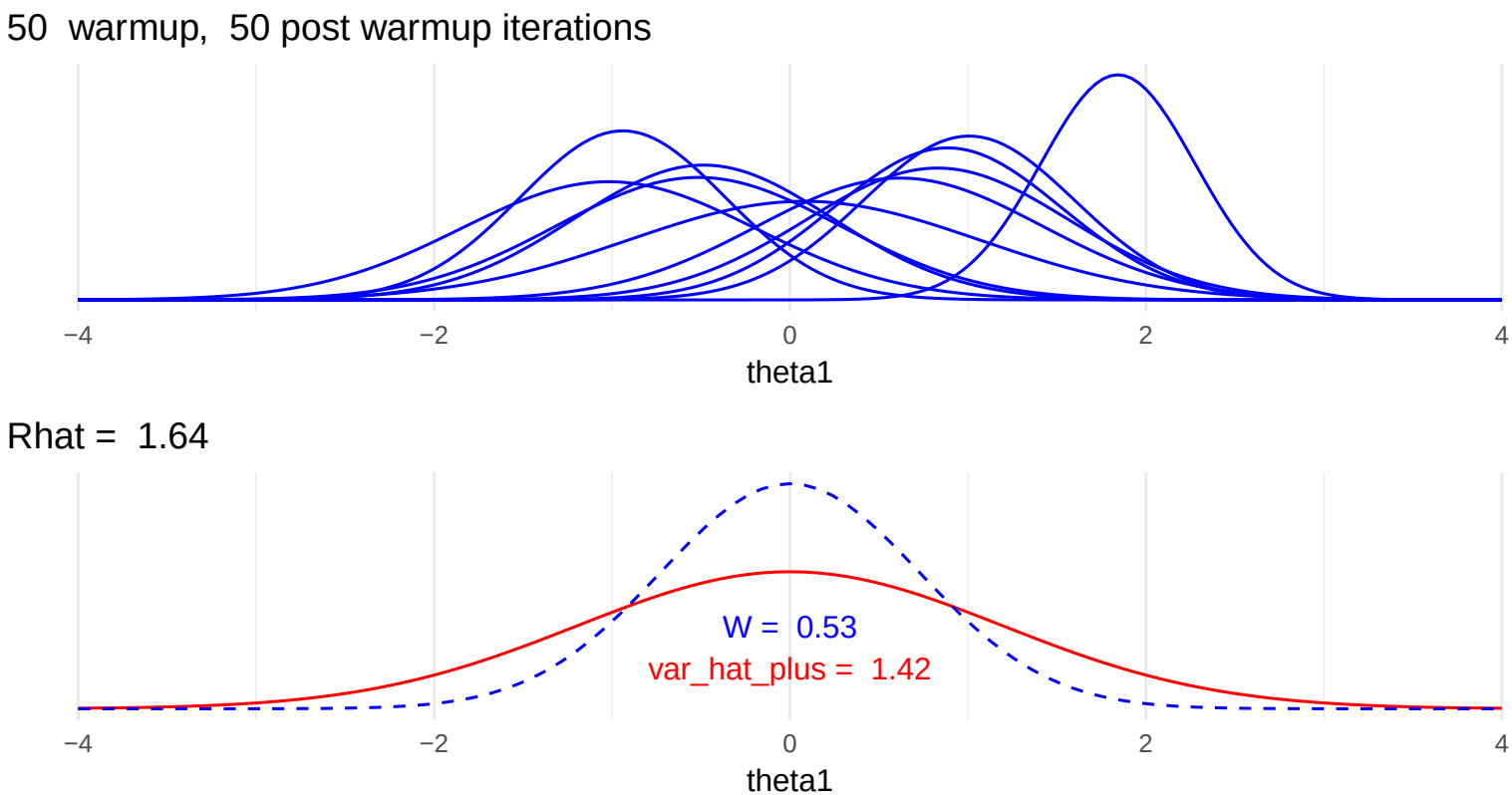
  - this overestimates marginal posterior variance if the starting points are overdispersed

- Given finite N, W underestimates marginal posterior variance
  - single chains have not yet visited all points in the distribution
  - when $N \to \infty, \quad \mathrm{E}(W) \to \mathrm{var}(\theta \mid y)$

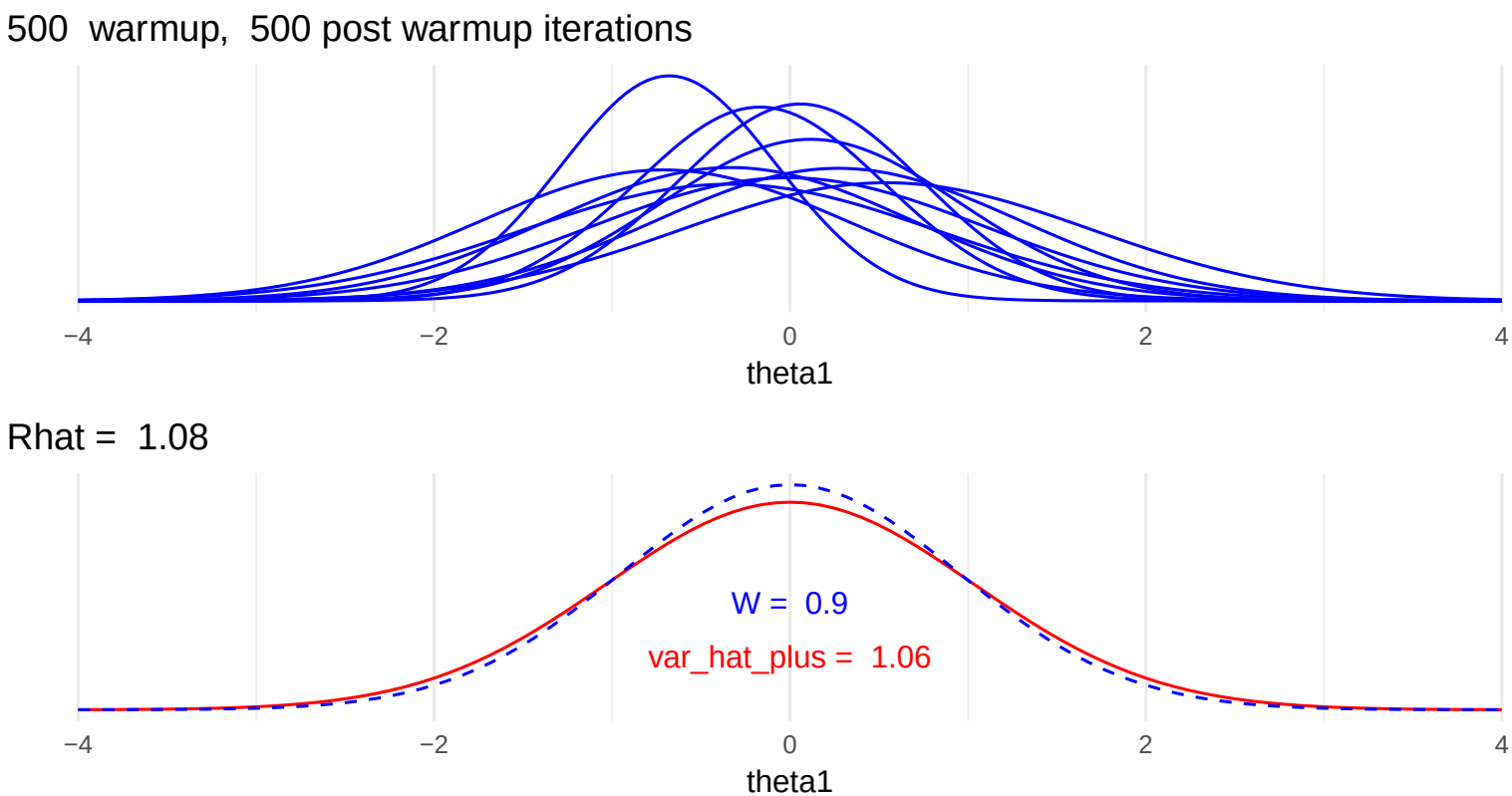
- As $\widehat{\mathrm{var}}^{+}(\theta \mid y)$ overestimates and W underestimates, compute

$$\widehat{R} = \sqrt{\frac{\widehat{\mathrm{var}}^{+}}{W}}$$

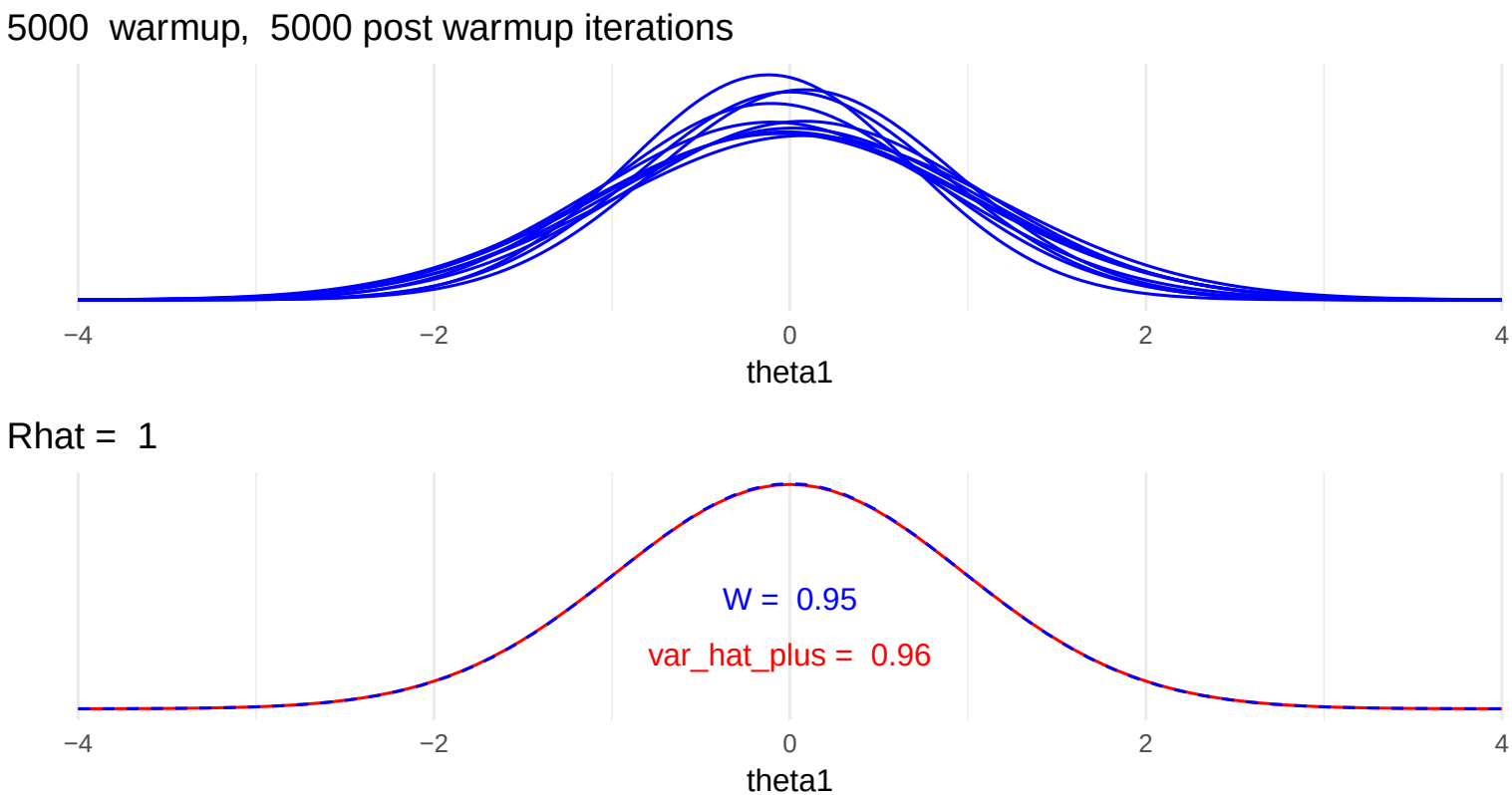
- BDA3: $\widehat{R}$ aka *potential scale reduction factor* (PSRF)
- Compare means and variances of the chains
  W = within chain variance estimate
  var_hat_plus = total variance estimate



- BDA3: $\widehat{R}$ aka *potential scale reduction factor* (PSRF)
- Compare means and variances of the chains
  W = within chain variance estimate
  var_hat_plus = total variance estimate



- BDA3: $\widehat{R}$ aka *potential scale reduction factor* (PSRF)
- Compare means and variances of the chains
  W = within chain variance estimate
  var_hat_plus = total variance estimate



$\widehat{R}$

$$\widehat{R} = \sqrt{\frac{\widehat{\text{var}}^{+}}{W}}$$

- Estimates how much the scale of ψ could reduce if $N \to \infty$
- $\widehat{R} \to 1$, when $N \to \infty$
- if $\widehat{R}$ is big (e.g., $R > 1.01$), keep sampling

$$\widehat{R} = \sqrt{\frac{\widehat{\text{var}}^{+}}{W}}$$

- Estimates how much the scale of ψ could reduce if $N \to \infty$
- $\widehat{R} \to 1$, when $N \to \infty$
- if $\widehat{R}$ is big (e.g., $R > 1.01$), keep sampling
- If $\widehat{R}$ close to 1, it is still possible that chains have not converged
  - if starting points were not overdispersed
  - distribution far from normal (especially if infinite variance)
  - just by chance when N is finite

Split-$\widehat{R}$

- BDA3: split-$\widehat{R}$
- Examines *mixing* and *stationarity* of chains
- To examine stationarity chains are split to two parts
  - after splitting, we have M chains, each having N draws
  - scalar draws $\theta_{nm} \quad (n = 1, \ldots, N; m = 1, \ldots, M)$
  - compare means and variances of the split chains

Rank normalized $\widehat{R}$

- Original $\widehat{R}$ requires that the target distribution has finite mean and variance

Vehtari, Gelman, Simpson, Carpenter, Bürkner (2020). Rank-normalization, folding, and localization: An improved $\widehat{R}$ for assessing convergence of MCMC. Bayesian Analysis, doi:10.1214/20-BA1221. https://projecteuclid.org/euclid.ba/1593828229.

Rank normalized $\widehat{R}$

- Original $\widehat{R}$ requires that the target distribution has finite mean and variance
- Rank normalization fixes this, is also more robust given finite but high variance, and is more sensitive to differences in variance
  - inverse normal-cdf of ranks
  - inverse normal-cdf of ranks of absolute difference from median

Vehtari, Gelman, Simpson, Carpenter, Bürkner (2020). Rank-normalization, folding, and localization: An improved $\widehat{R}$ for assessing convergence of MCMC. Bayesian Analysis, doi:10.1214/20-BA1221. https://projecteuclid.org/euclid.ba/1593828229.

Rank normalized $\widehat{R}$

- Original $\widehat{R}$ requires that the target distribution has finite mean and variance
- Rank normalization fixes this, is also more robust given finite but high variance, and is more sensitive to differences in variance
  - inverse normal-cdf of ranks
  - inverse normal-cdf of ranks of absolute difference from median
- The original $\widehat{R}$ is still needed for ESS/MCSE computation as shown later

Vehtari, Gelman, Simpson, Carpenter, Bürkner (2020). Rank-normalization, folding, and localization: An improved $\widehat{R}$ for assessing convergence of MCMC. Bayesian Analysis, doi:10.1214/20-BA1221. https://projecteuclid.org/euclid.ba/1593828229.

Rank normalized $\widehat{R}$

- Original $\widehat{R}$ requires that the target distribution has finite mean and variance
- Rank normalization fixes this, is also more robust given finite but high variance, and is more sensitive to differences in variance
  - inverse normal-cdf of ranks
  - inverse normal-cdf of ranks of absolute difference from median
- The original $\widehat{R}$ is still needed for ESS/MCSE computation as shown later
- Notation updated compared to BDA3

Vehtari, Gelman, Simpson, Carpenter, Bürkner (2020). Rank-normalization, folding, and localization: An improved $\widehat{R}$ for assessing convergence of MCMC. Bayesian Analysis, doi:10.1214/20-BA1221. https://projecteuclid.org/euclid.ba/1593828229.

$\widehat{R}$ and rank normalized $\widehat{R}$ in `posterior` package

`rhat_basic()` without rank normalization
`rhat()` with rank normalization

$\widehat{R}$ and rank normalized $\widehat{R}$ in `posterior` package

`rhat_basic()` without rank normalization
`rhat()` with rank normalization

```
x <- array(data=c(rnorm(1000,mean=-3),
                  rnorm(1000,mean=3)),
           dim=c(1000, 2, 1))
x <- as_draws_matrix(x)
variables(x) <- "N_2"
```

$\widehat{R}$ and rank normalized $\widehat{R}$ in `posterior` package

`rhat_basic()` without rank normalization
`rhat()` with rank normalization

```
x <- array(data=c(rnorm(1000,mean=-3),
                  rnorm(1000,mean=3)),
           dim=c(1000, 2, 1))
x <- as_draws_matrix(x)
variables(x) <- "N_2"
```

```
x |>
  summarise_draws(mean, sd, mcse_mean, rhat_basic, rhat)
 variable   mean    sd mcse_mean rhat_basic  rhat
 N_2      0.0122  3.18      2.15       3.61  1.83
```

$\widehat{R}$ and rank normalized $\widehat{R}$ in `posterior` package

`rhat_basic()` without rank normalization

`rhat()` with rank normalization

```
x <- array(data=c(rt(1000,df=1)-6,
                  rt(1000,df=1)+6),
           dim=c(1000, 2, 1))
x <- as_draws_matrix(x)
variables(x) <- "t1_2"
```

$\widehat{R}$ and rank normalized $\widehat{R}$ in `posterior` package

`rhat_basic()` without rank normalization
`rhat()` with rank normalization

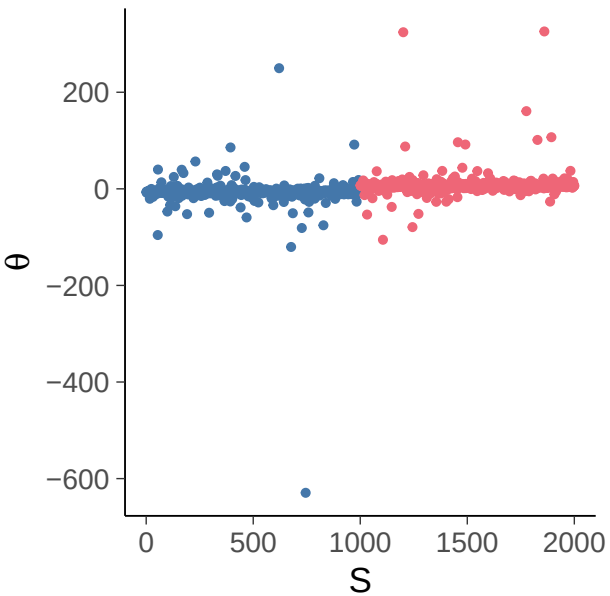
```
x <- array(data=c(rt(1000,df=1)-6,
                  rt(1000,df=1)+6),
           dim=c(1000, 2, 1))
x <- as_draws_matrix(x)
variables(x) <- "t1_2"
```

```
x |>
  summarise_draws(mean, sd, mcse_mean, pareto_khat,
                  rhat_basic, rhat)
 variable  mean    sd mcse_mean pareto_khat rhat_basic  rhat
 t1_2     -1.11  42.1      1.23        1.07       1.01  1.47
```

Rank normalized $\widehat{R}$

For example:

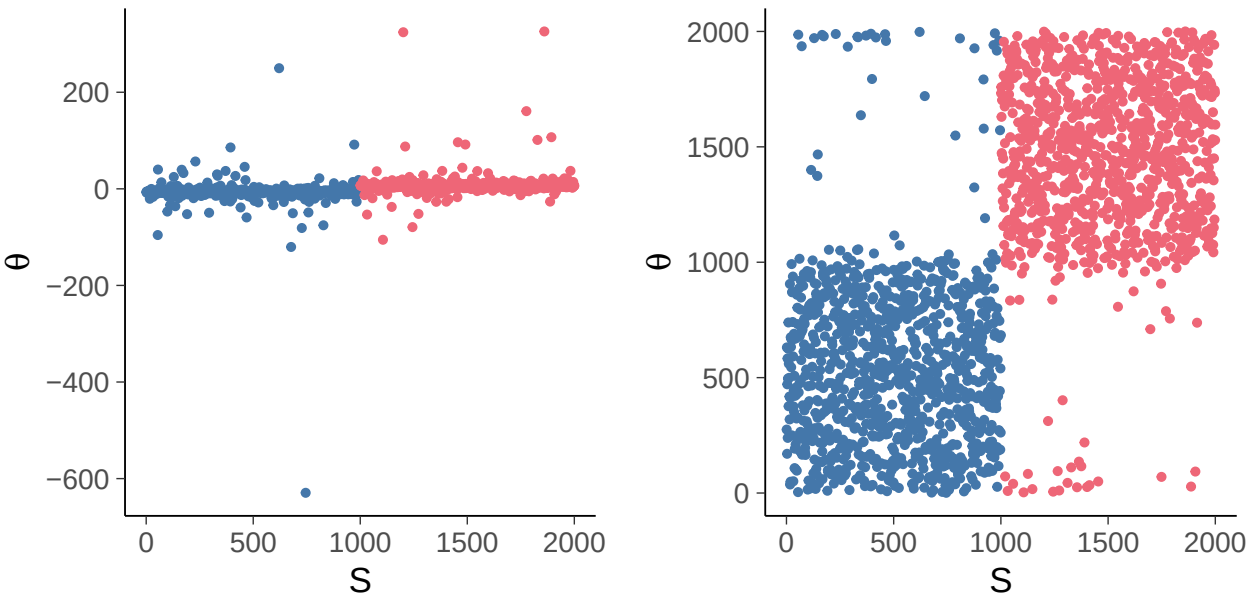
Cauchy mixture



Rank normalized $\widehat{R}$

For example:

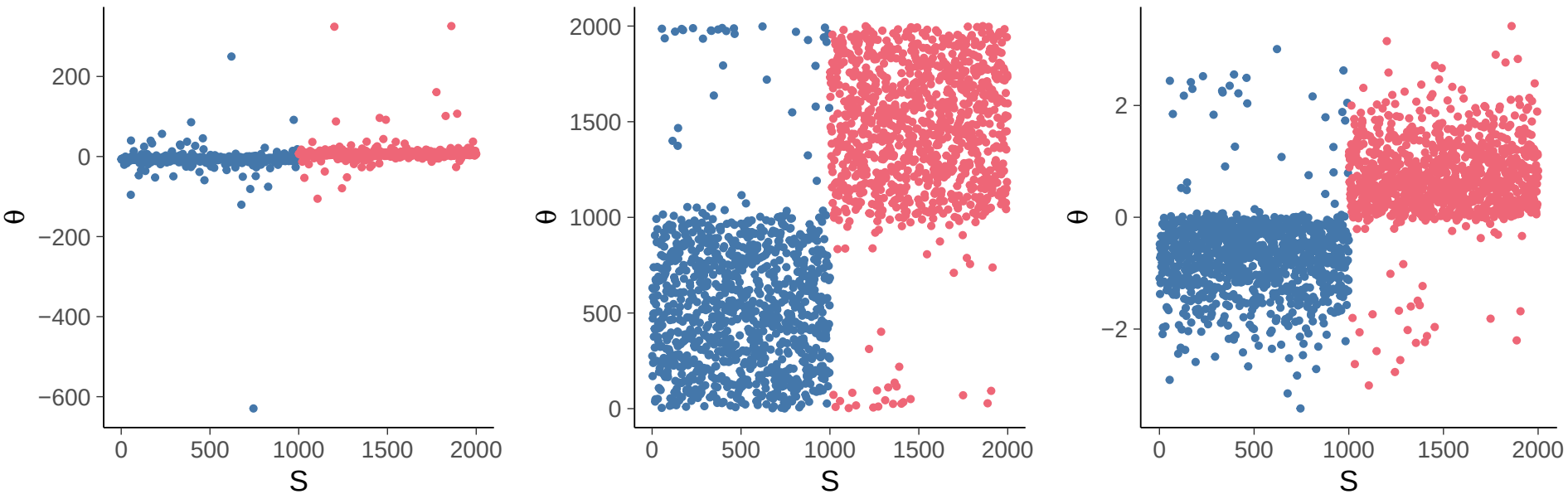
Cauchy mixture $\longrightarrow$ ranks



Rank normalized $\widehat{R}$

For example:

Cauchy mixture → ranks → inverse cdf



$\widehat{R}$ and rank normalized $\widehat{R}$ in `posterior` package

`rhat_basic()` without rank normalization
`rhat()` with **folding** and rank normalization

```
x <- array(data=c(normal(1000),
                  normal(1000)*2),
           dim=c(1000, 2, 1))
x <- as_draws_matrix(x)
variables(x) <- "N2v"
```

$\widehat{R}$ and rank normalized $\widehat{R}$ in `posterior` package

`rhat_basic()` without rank normalization
`rhat()` with **folding** and rank normalization

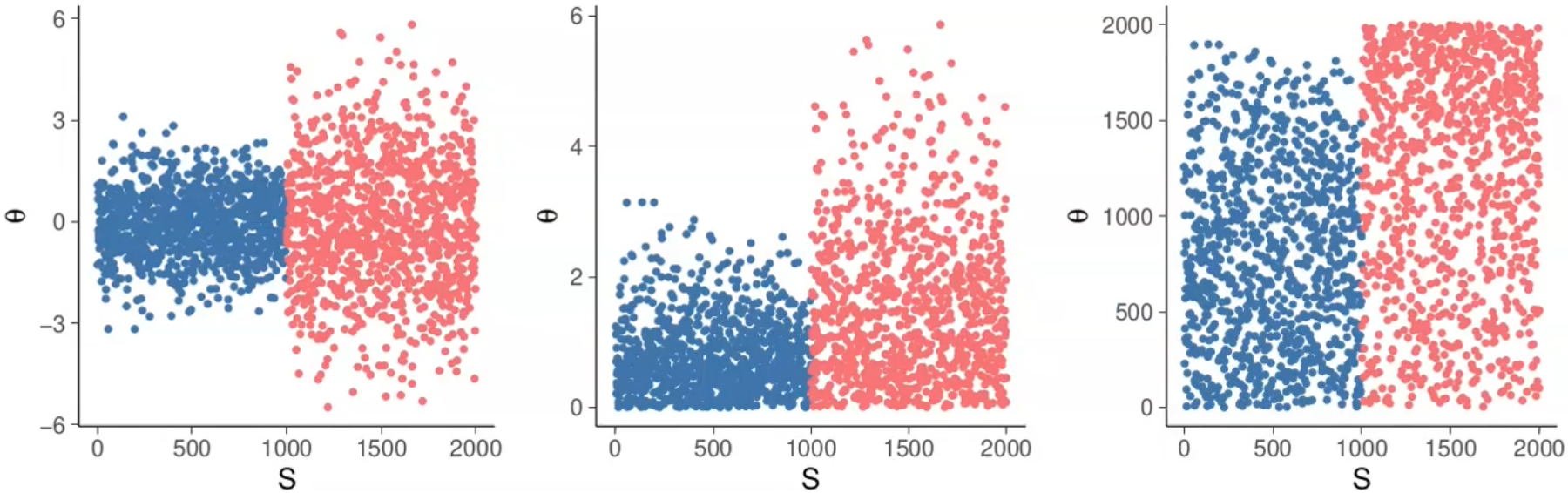
```
x <- array(data=c(normal(1000),
                  normal(1000)*2),
           dim=c(1000, 2, 1))
x <- as_draws_matrix(x)
variables(x) <- "N2v"
```

```
x |>
  summarise_draws(mean, sd, mcse_mean,
                  rhat_basic, rhat)
 variable  mean    sd mcse_mean rhat_basic  rhat
 N2v      -0.05   1.6      0.03       1.00  1.09
```

Rank normalized $\widehat{R}$ with folding

For example:

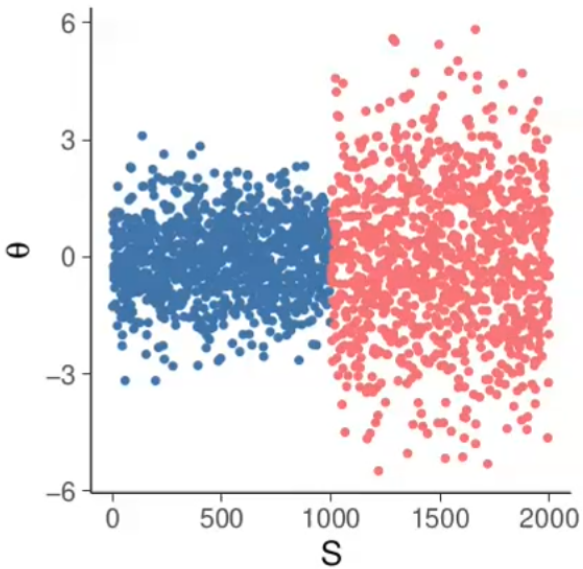
Normal variance mixture $\longrightarrow$ folded $\longrightarrow$ ranks

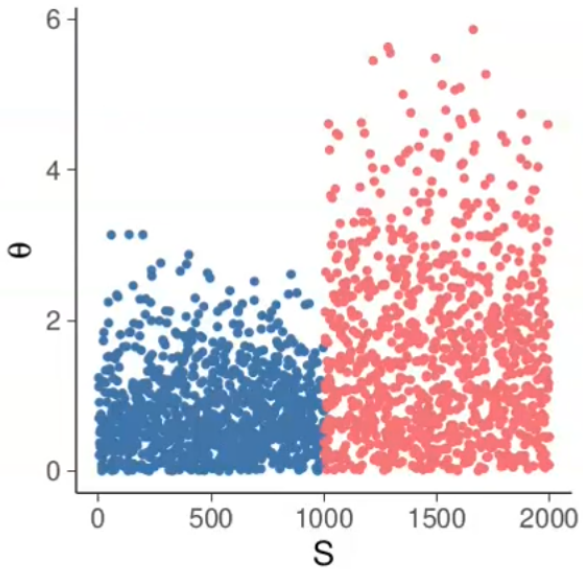


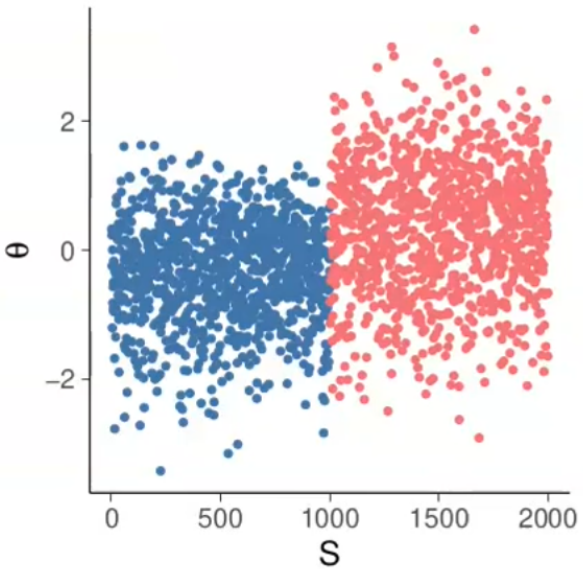
Rank normalized $\widehat{R}$ with folding

For example:

Normal variance mixture $\longrightarrow$ folded $\longrightarrow$ inverse cdf





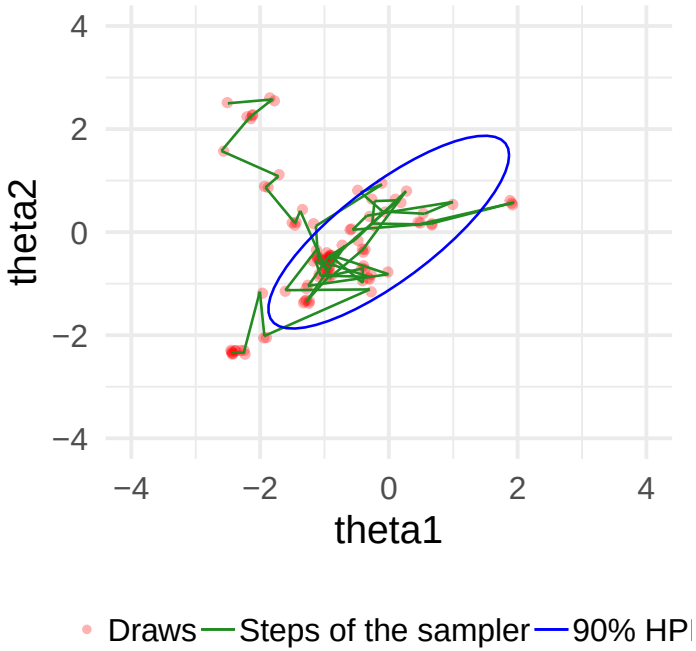


Time series analysis

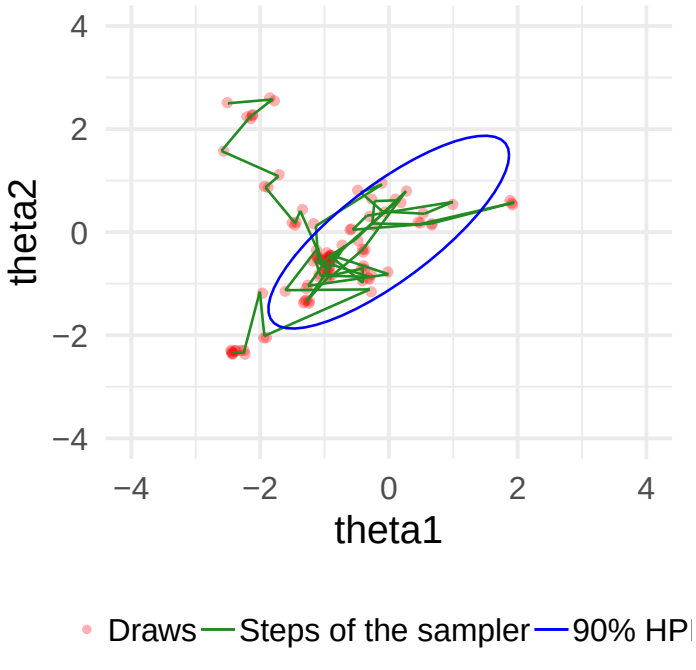
- Autocorrelation function
  - describes the correlation given a certain lag
  - can be used to compare efficiency of MCMC algorithms and parameterizations
  - For real valued, the correlation at lag n

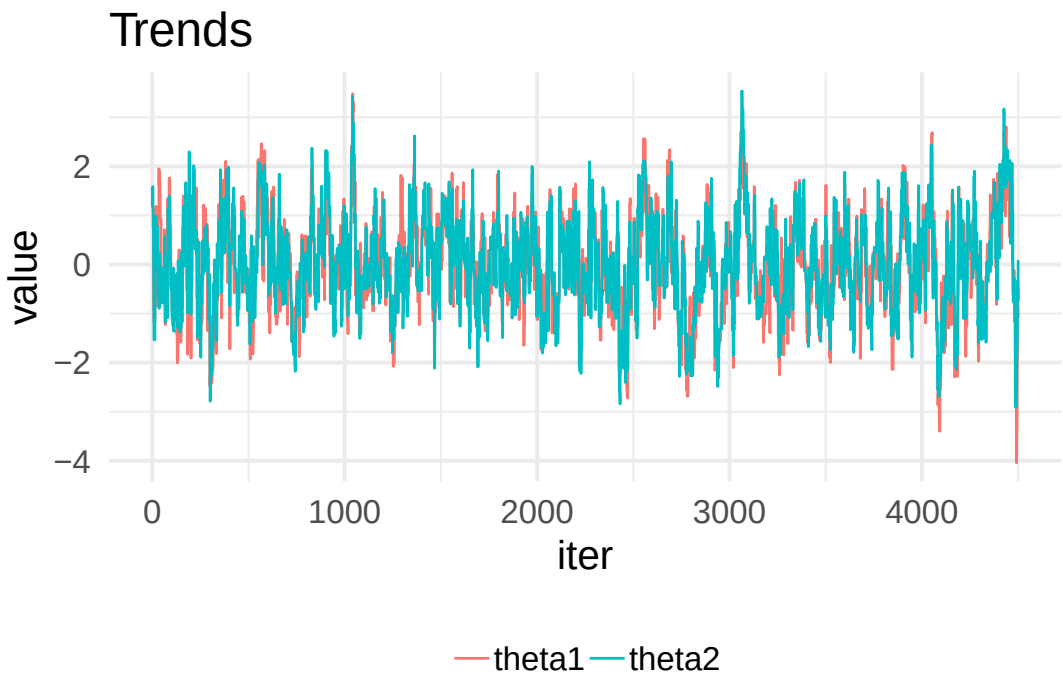
$$\frac{\mathrm{E}\left[(X_{t+n} - \mathrm{E}[X])(X_t - \mathrm{E}[X])\right]}{\mathrm{Var}\left[X\right]}$$

Autocorrelation

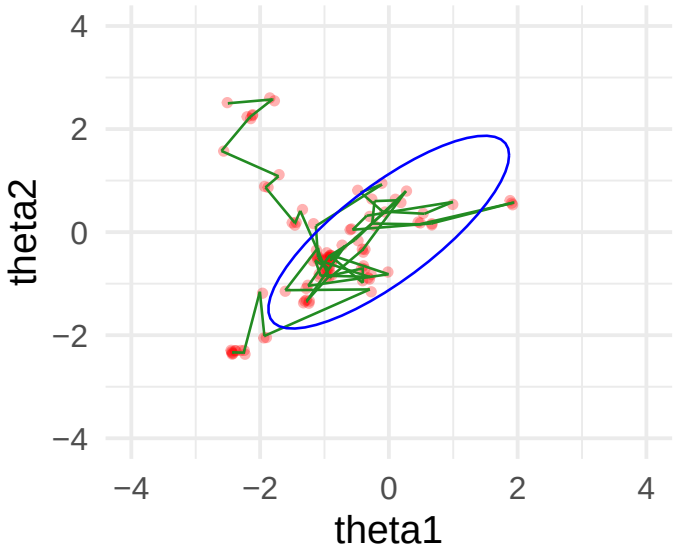


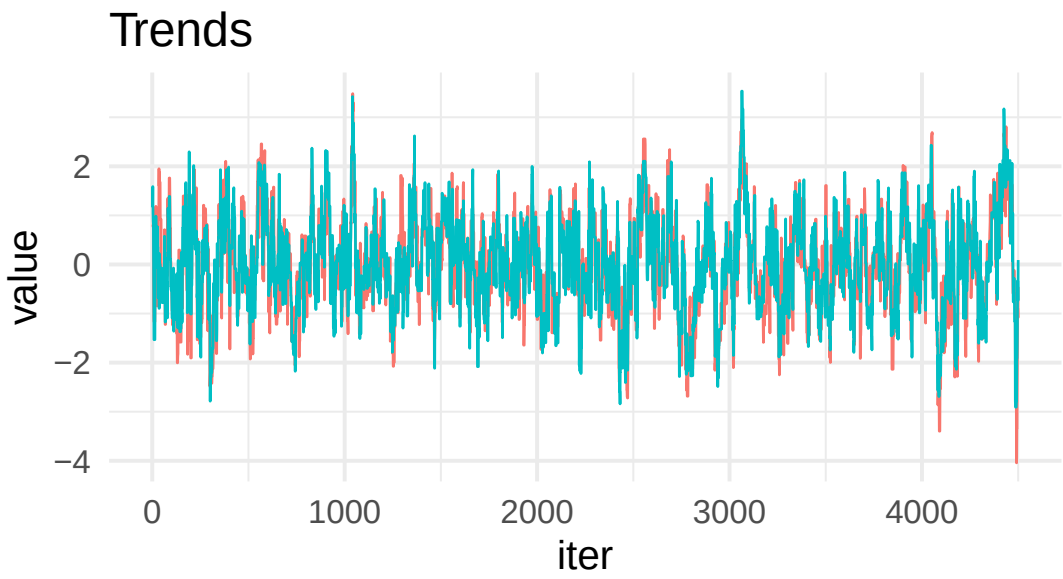
Autocorrelation





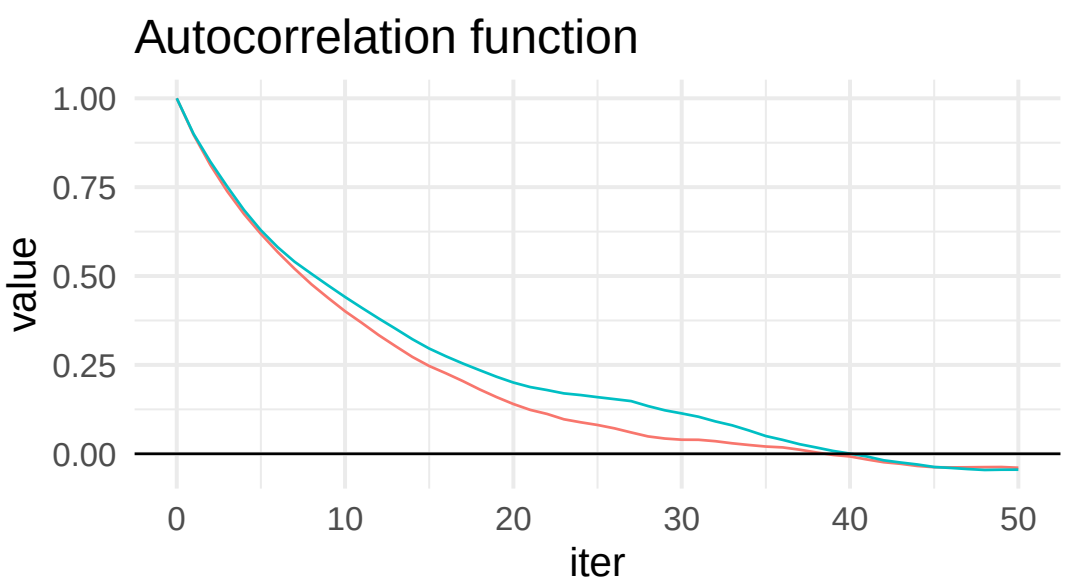
Autocorrelation



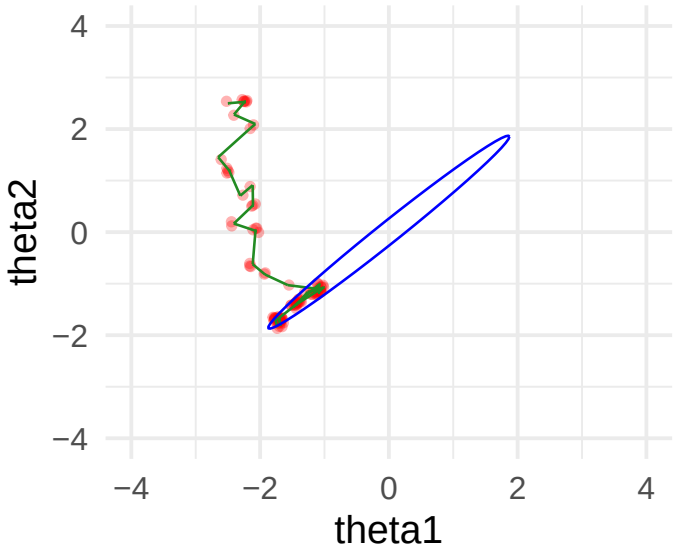


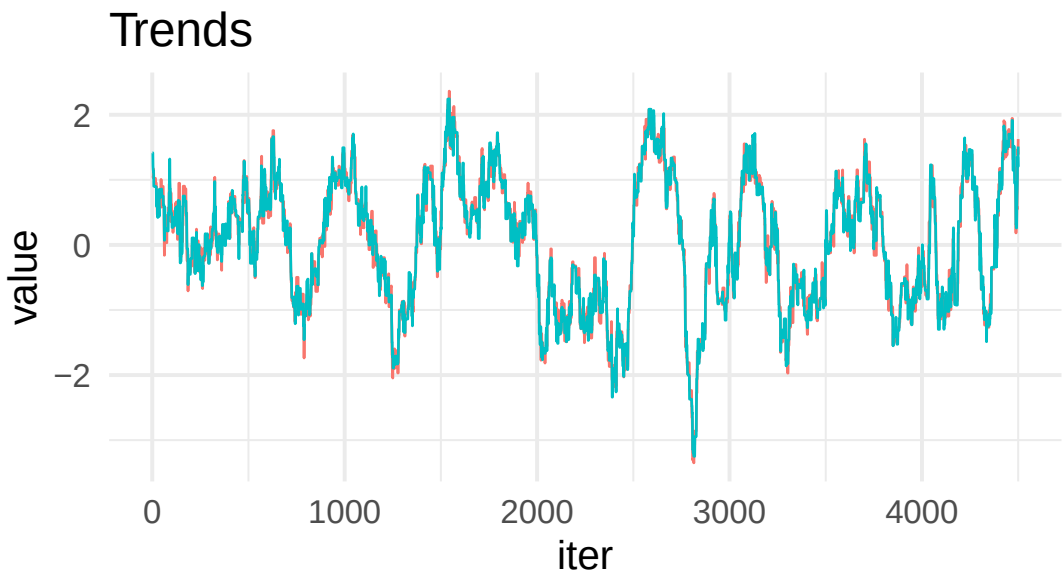
Draws Steps of the sampler 90% HP

theta1 theta2



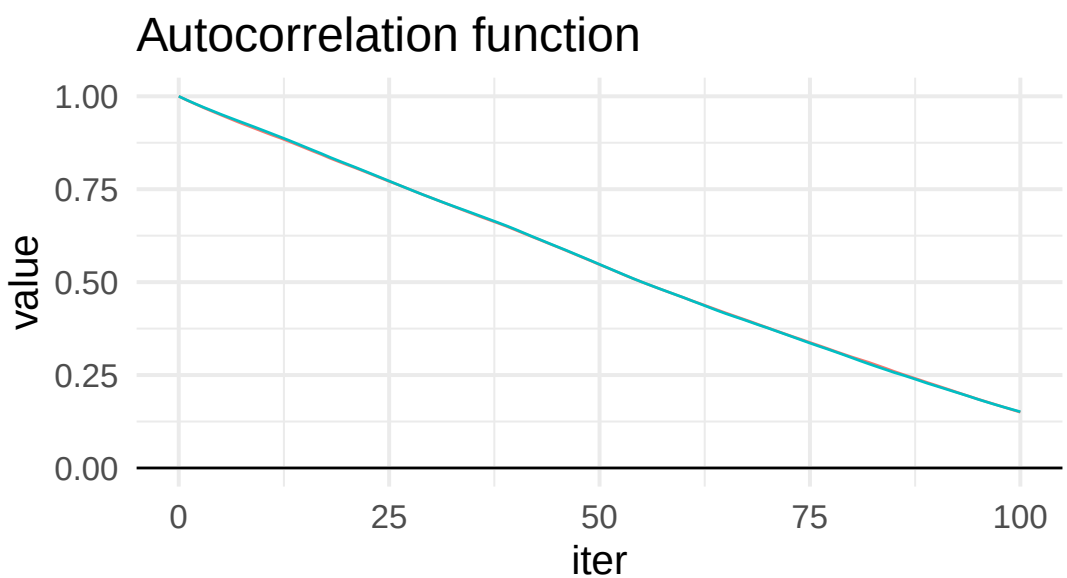
Autocorrelation (slow mixing due to small step size)



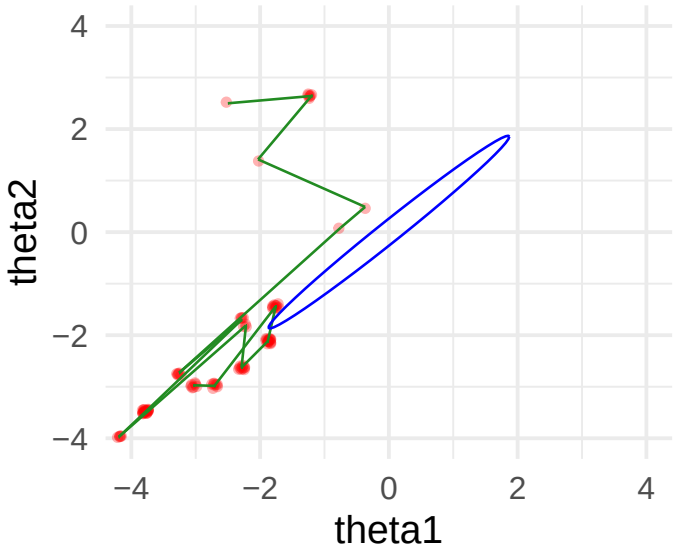


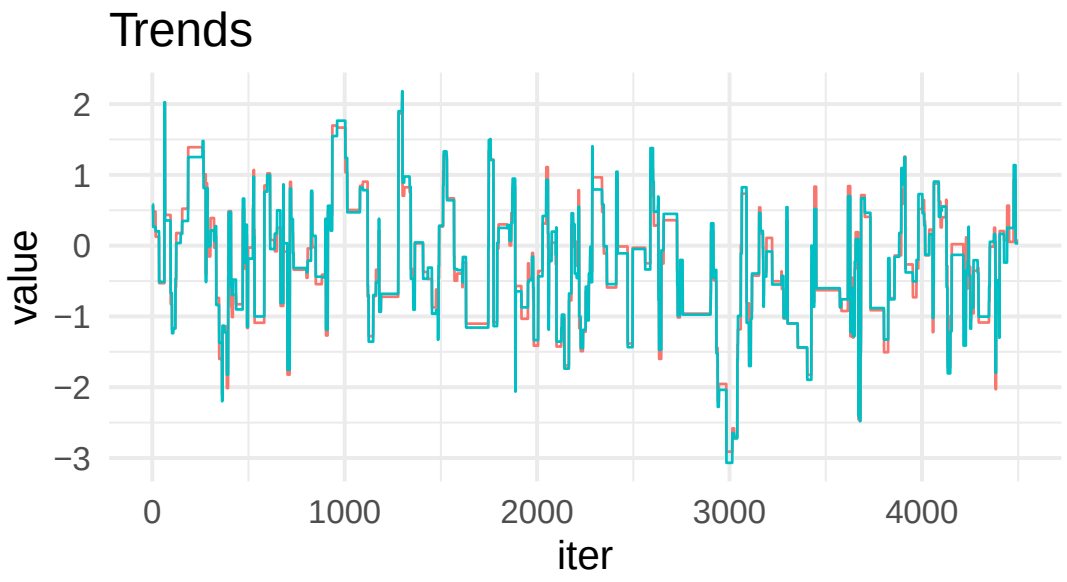
Draws Steps of the sampler 90% HP

theta1 theta2

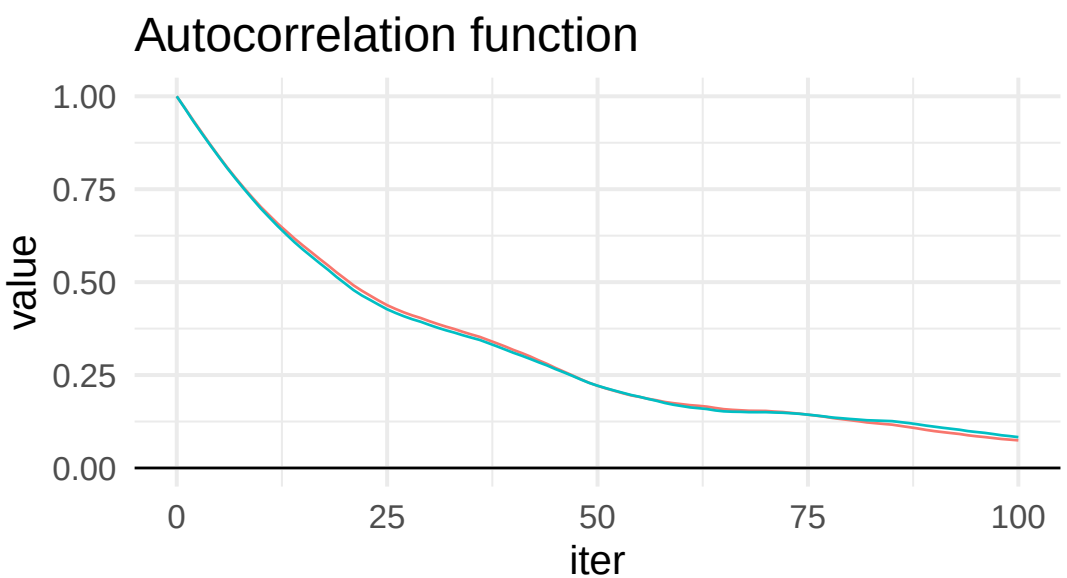


Autocorrelation (slow mixing due to many rejections)





Draws Steps of the sampler 90% HP

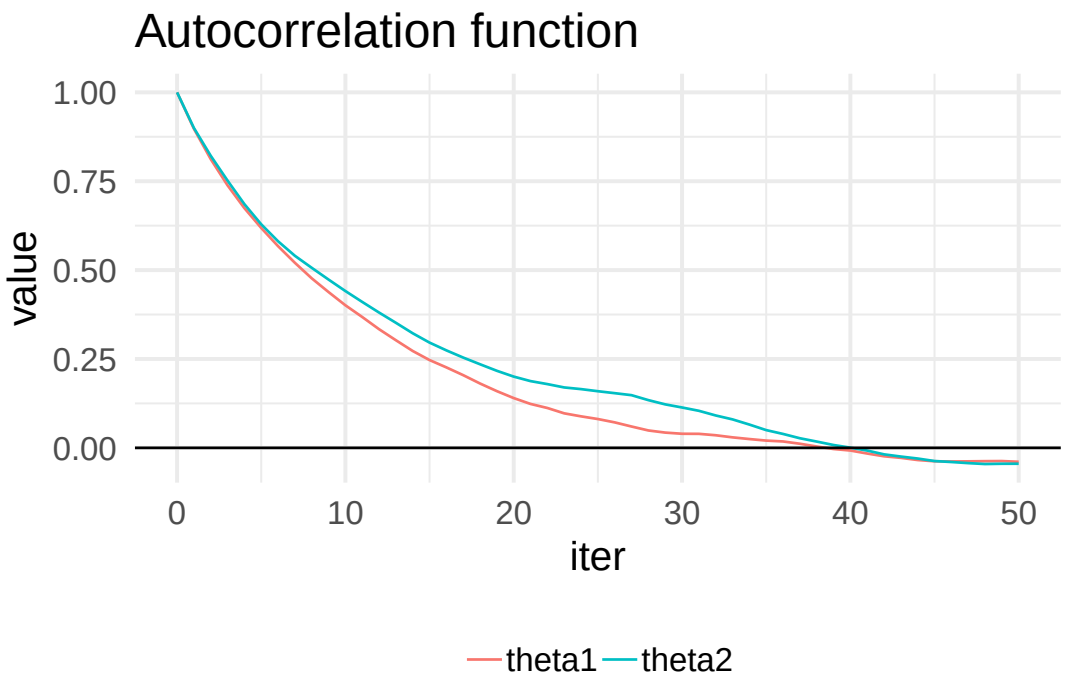


Time series analysis

- Time series analysis can be used to estimate Monte Carlo error in case of MCMC
- For expectation $\bar{\theta}$

$$\mathrm{Var}[\bar{\theta}] = \frac{\sigma_\theta^2}{S_{\mathrm{eff}}}$$

where $S_{\mathrm{eff}} = S/\tau$ (=ESS),
and τ is sum of autocorrelations



Time series analysis

- Time series analysis can be used to estimate Monte Carlo error in case of MCMC
- For expectation $\bar{\theta}$

$$\mathrm{Var}[\bar{\theta}] = \frac{\sigma_\theta^2}{S_{\mathrm{eff}}}$$

where $S_{\mathrm{eff}} = S/\tau$ (=ESS),
and τ is sum of autocorrelations

  - τ describes the relative inefficiency due to the dependency

Time series analysis

- Time series analysis can be used to estimate Monte Carlo error in case of MCMC
- For expectation $\bar{\theta}$

$$\mathrm{Var}[\bar{\theta}] = \frac{\sigma_\theta^2}{S_{\mathrm{eff}}}$$

where $S_{\mathrm{eff}} = S/\tau$ (=ESS),
and τ is sum of autocorrelations

  - τ describes the relative inefficiency due to the dependency
  - new $\widehat{R}$ paper $S = NM$ (in BDA3 $N = nm$ and $n_{\mathrm{eff}} = N/\tau$)

Time series analysis

- Time series analysis can be used to estimate Monte Carlo error in case of MCMC
- For expectation $\bar{\theta}$

$$\mathrm{Var}[\bar{\theta}] = \frac{\sigma_\theta^2}{S_{\text{eff}}}$$

where $S_{\text{eff}} = S/\tau$ (=ESS),
and τ is sum of autocorrelations

  - τ describes the relative inefficiency due to the dependency
  - new $\widehat{R}$ paper $S = NM$ (in BDA3 $N = nm$ and $n_{\text{eff}} = N/\tau$)
  - BDA3 focuses on S_{eff} and not the Monte Carlo error directly
    new $\widehat{R}$ paper discusses more about MCSEs for different quantities

Time series analysis

- Estimation of the autocorrelation using several chains

$$\hat{\rho}_n = 1 - \frac{W - \frac{1}{M}\sum_{m=1}^{M} \hat{\rho}_{n,m}}{2\widehat{\mathrm{var}}^{+}}$$

where $\hat{\rho}_{n,m}$ is autocorrelation at lag n for chain m, and W and $\widehat{\mathrm{var}}^{+}$ are the same as in $\widehat{R}$ (without rank normalization)

Time series analysis

- Estimation of the autocorrelation using several chains

$$\hat{\rho}_n = 1 - \frac{W - \frac{1}{M}\sum_{m=1}^{M} \hat{\rho}_{n,m}}{2\widehat{\mathrm{var}}^{+}}$$

  where $\hat{\rho}_{n,m}$ is autocorrelation at lag n for chain m, and W and $\widehat{\mathrm{var}}^{+}$ are the same as in $\widehat{R}$ (without rank normalization)

- This combines $\widehat{R}$ and autocorrelation estimates
  - takes into account if the chains are not mixing (the chains have not converged)

Time series analysis

- Estimation of the autocorrelation using several chains

$$\hat{\rho}_n = 1 - \frac{W - \frac{1}{M}\sum_{m=1}^{M} \hat{\rho}_{n,m}}{2\widehat{\mathrm{var}}^{+}}$$

  where $\hat{\rho}_{n,m}$ is autocorrelation at lag n for chain m, and W and $\widehat{\mathrm{var}}^{+}$ are the same as in $\widehat{R}$ (without rank normalization)
- This combines $\widehat{R}$ and autocorrelation estimates
  - takes into account if the chains are not mixing (the chains have not converged)
- BDA3 has slightly different and less accurate equation. The above equation is used in Stan 2.18+

Time series analysis

- Estimation of the autocorrelation using several chains

$$\hat{\rho}_n = 1 - \frac{W - \frac{1}{M}\sum_{m=1}^{M} \hat{\rho}_{n,m}}{2\widehat{\mathrm{var}}^{+}}$$

  where $\hat{\rho}_{n,m}$ is autocorrelation at lag n for chain m, and W and $\widehat{\mathrm{var}}^{+}$ are the same as in $\widehat{R}$ (without rank normalization)
- This combines $\widehat{R}$ and autocorrelation estimates
  - takes into account if the chains are not mixing (the chains have not converged)
- BDA3 has slightly different and less accurate equation. The above equation is used in Stan 2.18+
- Compared to a method which computes the autocorrelation from a single chain, the multi-chain estimate has smaller variance

Time series analysis

- Estimation of τ

$$\tau = 1 + 2 \sum_{t=1}^{\infty} \hat{\rho}_t$$

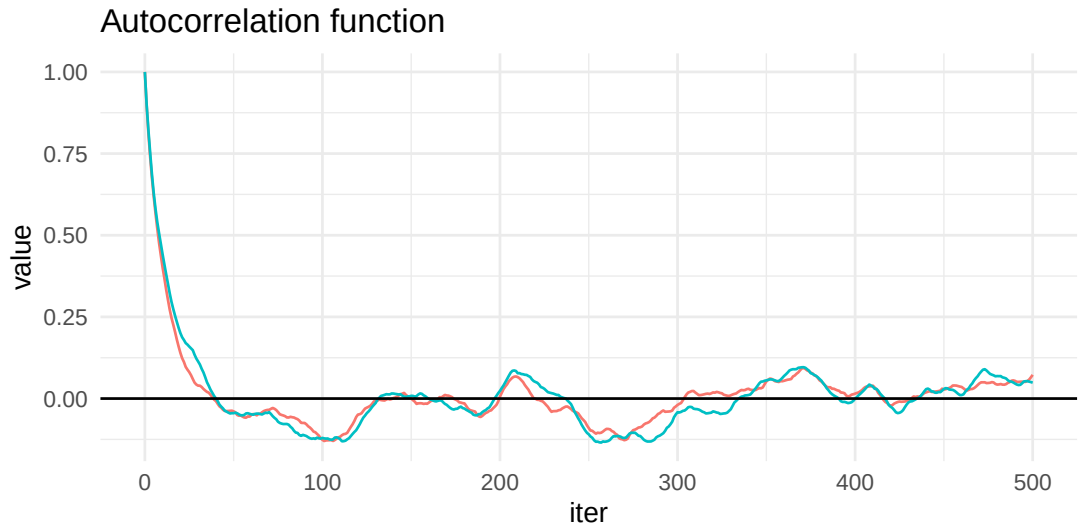
where $\hat{\rho}_t$ is empirical autocorrelation

Time series analysis

- Estimation of τ

$$\tau = 1 + 2 \sum_{t=1}^{\infty} \hat{\rho}_t$$

where $\hat{\rho}_t$ is empirical autocorrelation



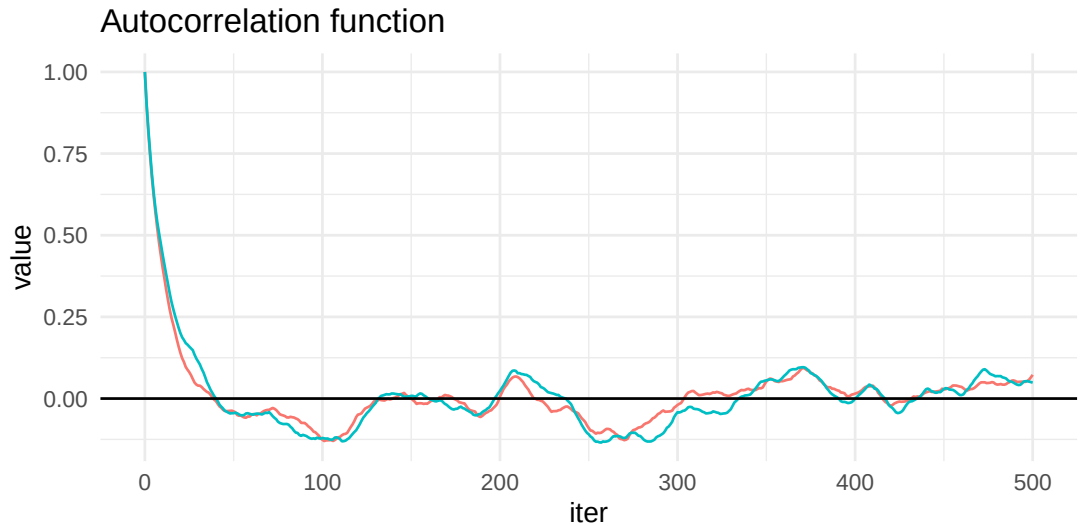
- empirical autocorrelation function is noisy and thus estimate of τ is noisy
- noise is larger for longer lags (less observations)

Time series analysis

- Estimation of τ

$$\tau = 1 + 2 \sum_{t=1}^{\infty} \hat{\rho}_t$$

where $\hat{\rho}_t$ is empirical autocorrelation



- empirical autocorrelation function is noisy and thus estimate of τ is noisy
- noise is larger for longer lags (less observations)
- less noisy estimate is obtained by truncating

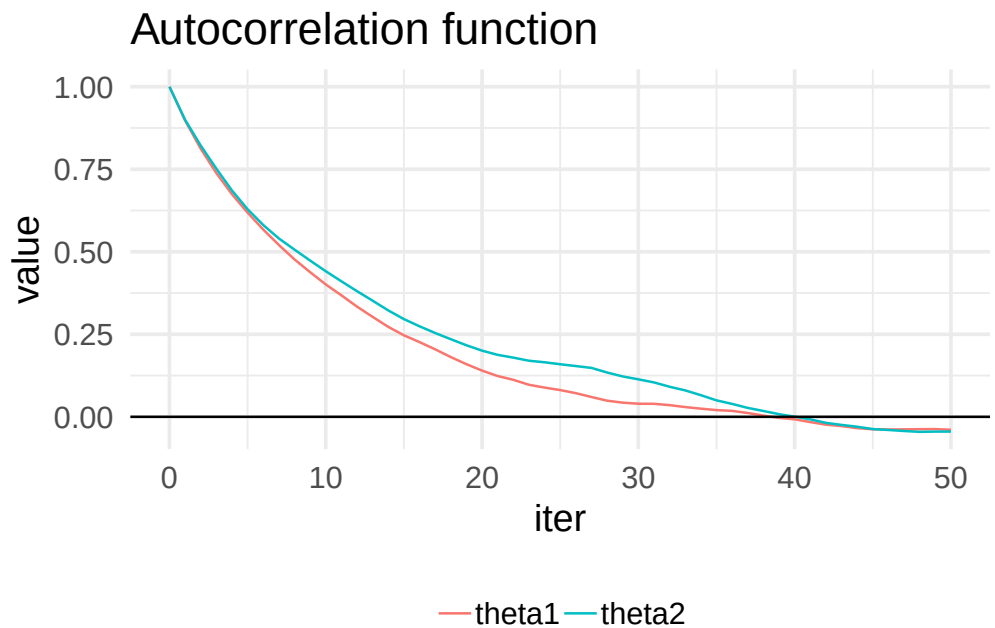
$$\hat{\tau} = 1 + 2 \sum_{t=1}^{T} \hat{\rho}_t$$

Geyer's adaptive window estimator

- Truncation can be decided adaptively
  - for stationary, irreducible, recurrent Markov chain
  - let $\Gamma_m = \rho_{2m} + \rho_{2m+1}$, which is sum of two consequent autocorrelations
  - Γ_m is positive, decreasing and convex function of m

Geyer's adaptive window estimator

- Truncation can be decided adaptively
  - for stationary, irreducible, recurrent Markov chain
  - let $\Gamma_m = \rho_{2m} + \rho_{2m+1}$, which is sum of two consequent autocorrelations
  - Γ_m is positive, decreasing and convex function of m
- Initial positive sequence estimator (Geyer's IPSE)
  - Choose the largest m so, that all values of the sequence $\hat{\Gamma}_1, \ldots, \hat{\Gamma}_m$ are positive

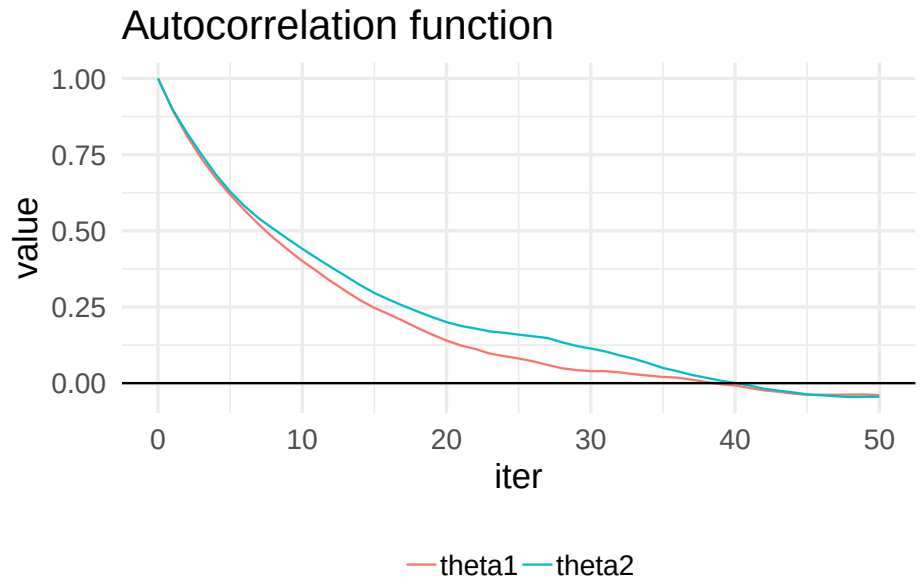


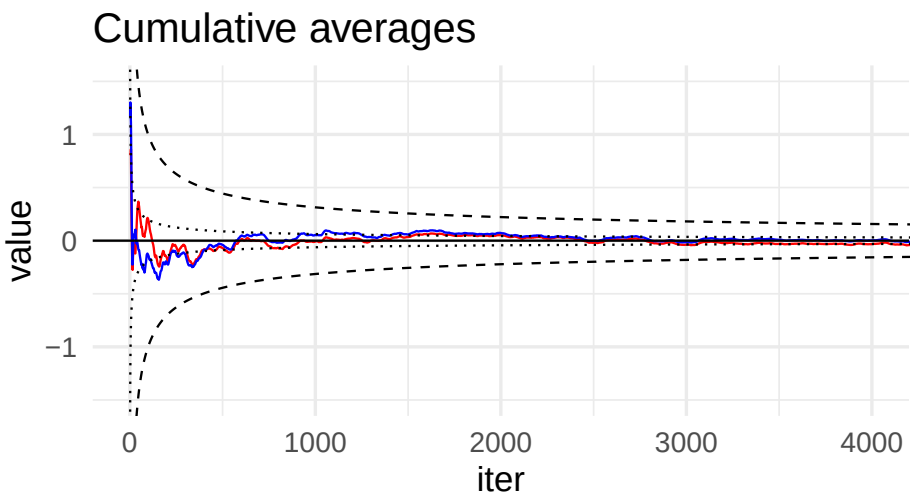
Effective sample size

Effective sample size ESS $= S_{\text{eff}} \approx S/\hat{\tau}$

Effective sample size

Effective sample size ESS $= S_{\text{eff}} \approx S/\hat{\tau}$





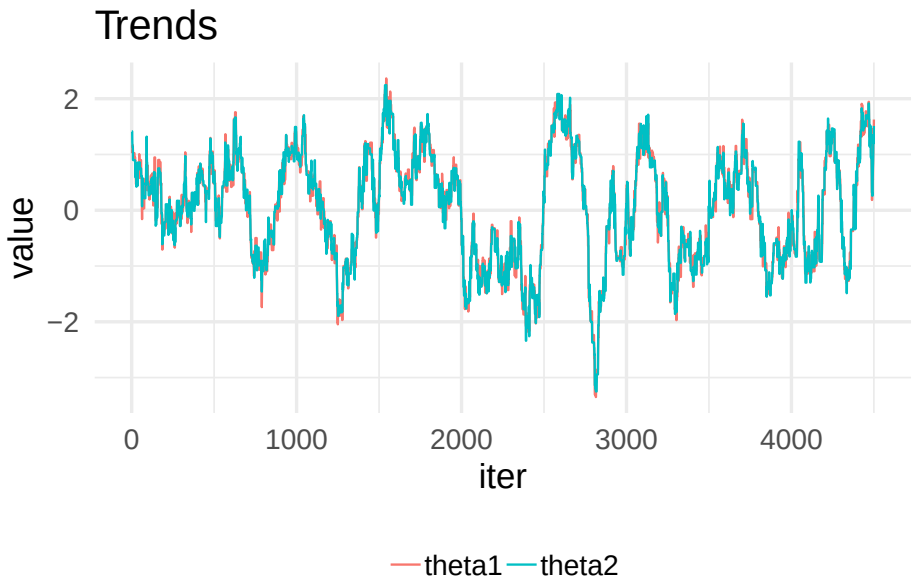
$$\hat{\tau} = 1 + 2\sum_{t=1}^{T} \hat{\rho}_t$$

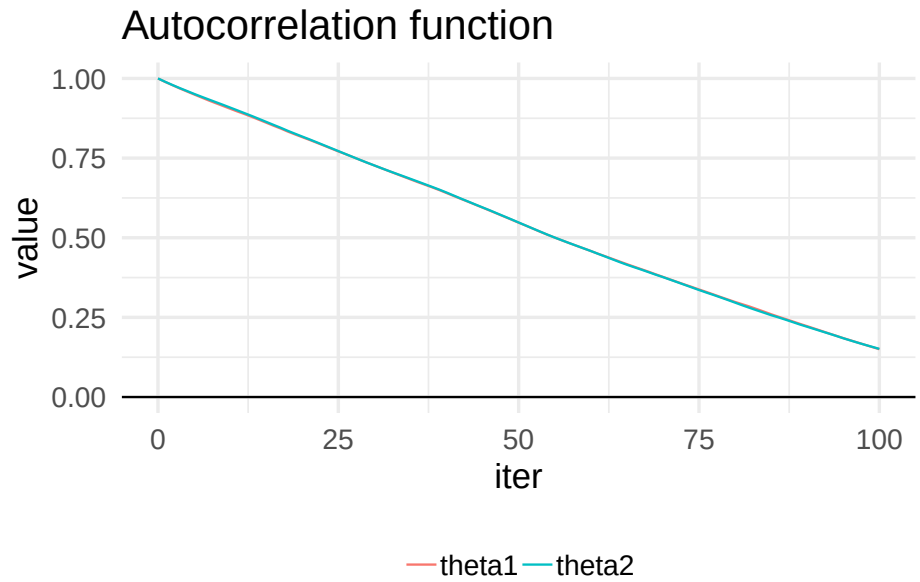
$$\approx 24$$

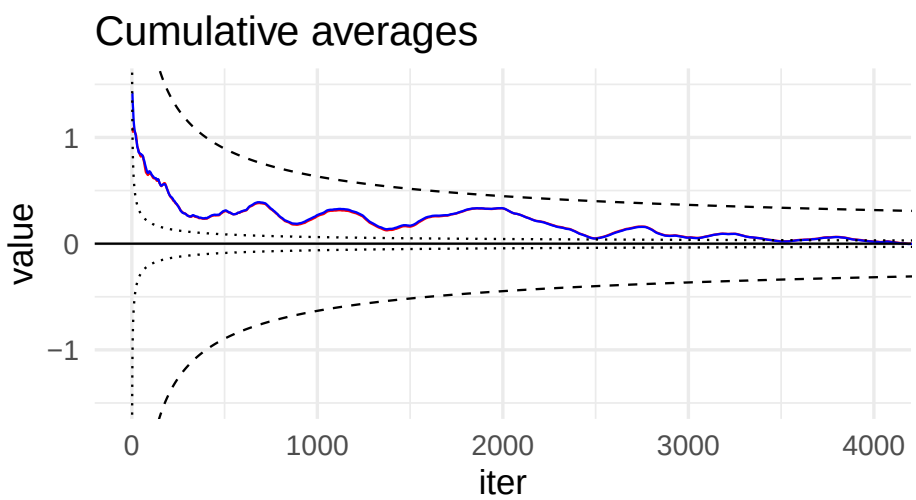
theta1 theta2 95% interval for MCMC error 95% interval for indeper

Effective sample size

Effective sample size ESS $= S_{\text{eff}} \approx S/\hat{\tau}$







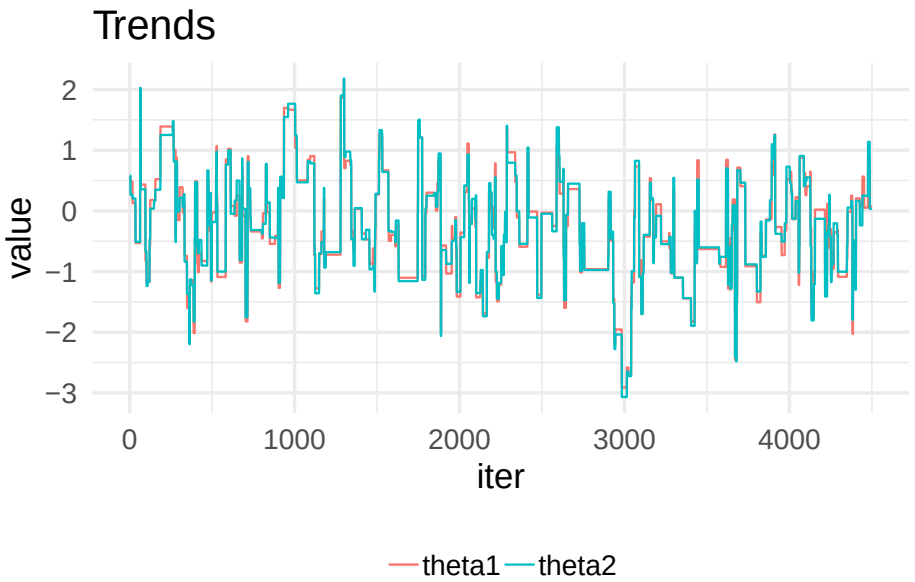
$$\hat{\tau} = 1 + 2\sum_{t=1}^{T} \hat{\rho}_t$$

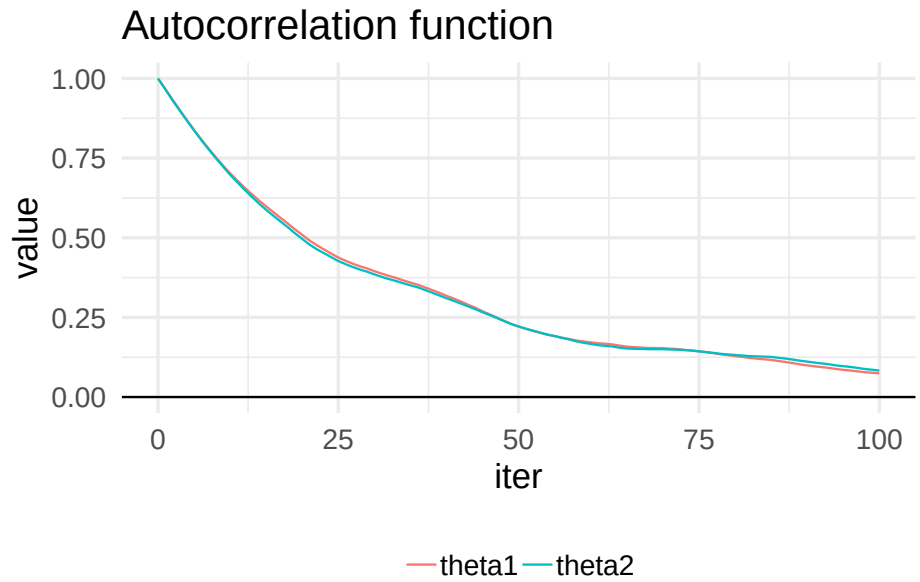
$$\approx 104$$

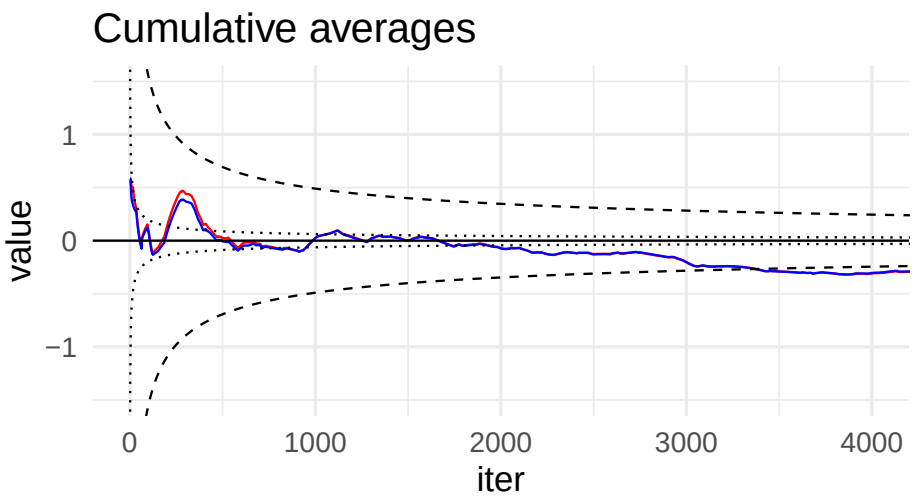
theta1 theta2 95% interval for MCMC error 95% interval for indeper

Effective sample size

Effective sample size ESS $= S_{\text{eff}} \approx S/\hat{\tau}$







$$\hat{\tau} = 1 + 2\sum_{t=1}^{T} \hat{\rho}_t$$

$$\approx 63$$

theta1 theta2 95% interval for MCMC error 95% interval for indeper

Monte Carlo standard error (MCSE)

- MCSE is obtained as discussed in lecture 4, but replacing the sample size S with the effective sample size ESS.
- See Digits case study for how many iterations to run and how many digits to report
  https://avehtari.github.io/casestudies/Digits/digits.html

ESS and MCSE in `posterior` package

Simulated 4 chains with AR(0.3) process

```
drt |> summarise_draws(mean, sd, pareto_khat,
                       ess_mean, mcse_mean)
```

ESS and MCSE in `posterior` package

Simulated 4 chains with AR(0.3) process

```
drt |> summarise_draws(mean, sd, pareto_khat,
                       ess_mean, mcse_mean)
```

variable	mean	sd	pareto_khat	ess_mean	mcse_mean
xn	0.01	0.99	-0.07	2280.	0.02
xt3	0.02	1.6	0.33	2452.	0.03
xt2	0.05	2.9	0.52	2903.	0.05
xt1	0.33	93.	1.0	3976.	1.5

ESS and MCSE in `posterior` package

Simulated 4 chains with AR(0.3) process

```
drt |> summarise_draws(mean, pareto_khat,
                       ess_mean, mcse_mean)
                       ess_quantile, mcse_quantile)
```

```
/variable mean pareto_khat ess_mean mcse_mean ess_q95 mcse_q95
xn        0.01       -0.07     2280.     0.02   3251.     0.04
xt3       0.02        0.33     2452.     0.03   3251.     0.09
xt2       0.05        0.52     2903.     0.05   3251.     0.13
xt1       0.33        1.0      3976.     1.5    3251.     0.49
```

Bulk-ESS and Tail-ESS in `posterior` package

- ESS depends on the quantity
- For quick diagnostic purposes the default summary shows
  - median and median absolute deviation (`mad`), which are valid in case of infinite mean and variance, too
  - if `mad` is much smaller than `sd`, suspect infinite variance
  - Rank-normalized $\widehat{R}$ `rhat`
  - Bulk-ESS (`ess_bulk`) is generic ESS for sampling efficiency in bulk using rank normalized values (works for infinite variance)
  - Tail-ESS (`ess_tail`) is the minimum ESS for 5%- and 95%-quantiles

```
drt |> summarise_draws()
```

variable	mean	median	sd	mad	q5	q95	rhat	ess_bulk	ess_tail
xn	0.01	0.00	0.99	0.99	-1.6	1.6	1.00	2284.	3189.
xt3	0.02	0.00	1.6	1.1	-2.3	2.3	1.00	2284.	3189.
xt2	0.05	0.00	2.9	1.2	-2.8	2.9	1.00	2284.	3189.
xt1	0.33	0.00	93.	1.5	-5.8	6.1	1.00	2284.	3189.

ESS and MCSE

- ESS and MCSE depend on the quantity
  - Bulk-ESS and Tail-ESS are useful diagnostic summaries, but eventually need to look at the ESS / MCSE for the quantity of interest

Diagnostic tools

For this week's assignment:

- $\widehat{R}$, ESS, MCSE
  - `library(posterior)`
  - `th |> summarise_draws(Rhat=basic_rhat, ESS=mean_ess)`
  - `th |> summarise_draws(mean, mean_mcse)`
  - `th |> summarize_draws(~quantile(.x, probs = c(0.05, 0.95)))`
  - see demo11_5 and Digits case study for the examples how to use these

Diagnostic tools

For this week's assignment:

- $\widehat{R}$, ESS, MCSE
  - `library(posterior)`
  - `th |> summarise_draws(Rhat=basic_rhat, ESS=mean_ess)`
  - `th |> summarise_draws(mean, mean_mcse)`
  - `th |> summarize_draws(~quantile(.x, probs = c(0.05, 0.95)))`
  - see demo11_5 and Digits case study for the examples how to use these
- trace, autocorrelation, density, scatter plots in R
  - `library(bayesplot)`
  - `mcmc_trace(th)`, `mcmc_acf(th)`, `mcmc_areas(th)`, ...
  - see demo11_5 for the examples for more bayesplot examples

Diagnostic tools

For this week's assignment:

- $\widehat{R}$, ESS, MCSE
  - `library(posterior)`
  - `th |> summarise_draws(Rhat=basic_rhat, ESS=mean_ess)`
  - `th |> summarise_draws(mean, mean_mcse)`
  - `th |> summarize_draws(~quantile(.x, probs = c(0.05, 0.95)))`
  - see demo11_5 and Digits case study for the examples how to use these
- trace, autocorrelation, density, scatter plots in R
  - `library(bayesplot)`
  - `mcmc_trace(th)`, `mcmc_acf(th)`, `mcmc_areas(th)`, ...
  - see demo11_5 for the examples for more bayesplot examples
- Python
  - see ArviZ package

Problematic distributions

- Nonlinear dependencies
  - optimal proposal depends on location

Problematic distributions

- Nonlinear dependencies
  - optimal proposal depends on location
- Funnels
  - optimal proposal depends on location

Problematic distributions

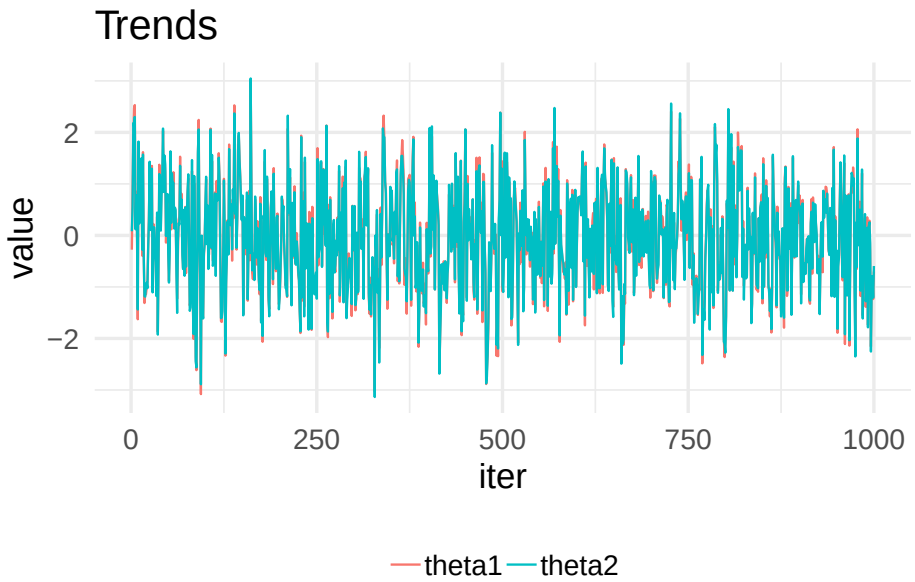
- Nonlinear dependencies
  - optimal proposal depends on location
- Funnels
  - optimal proposal depends on location
- Multimodal
  - difficult to move from one mode to another

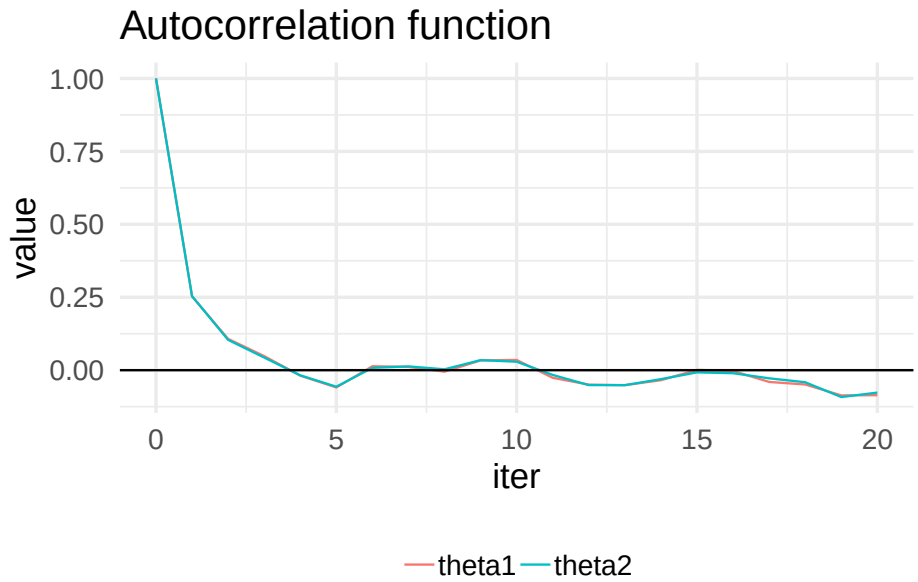
Problematic distributions

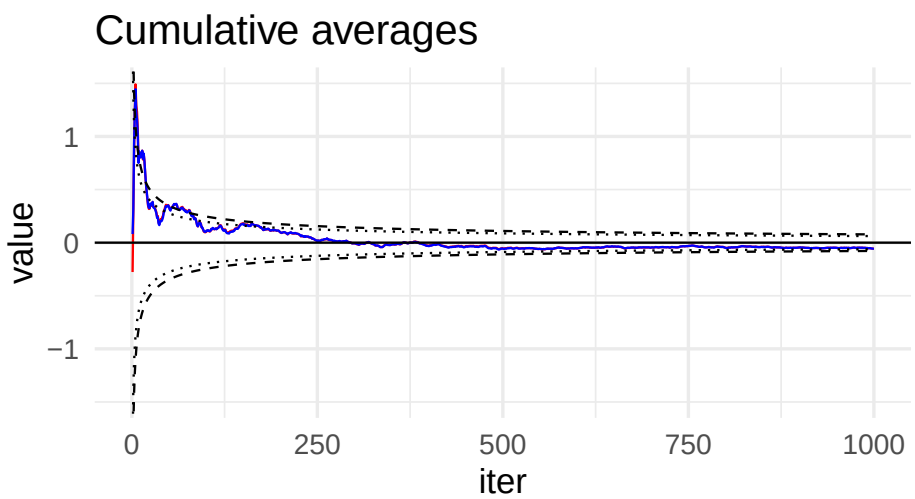
- Nonlinear dependencies
  - optimal proposal depends on location
- Funnels
  - optimal proposal depends on location
- Multimodal
  - difficult to move from one mode to another
- Long-tailed with non-finite variance and mean
  - central limit theorem for expectations does not hold

Next week: HMC, NUTS, and dynamic HMC

Effective sample size ESS $= S_{\text{eff}} \approx S/\hat{\tau}$







$$\hat{\tau} = 1 + 2\sum_{t=1}^{T} \hat{\rho}_t$$

$$\approx 1.6$$

theta1 theta2 95% interval for MCMC error 95% interval for indeper

Further diagnostics

- Pareto-$\hat{k}$ diagnostic for checking whether variance is finite
- Dynamic HMC/NUTS has additional diagnostics
  - divergences
  - tree depth exceedences

MCMC summary

- Construct a Markov chain which has desired stationary distribution
  - most of the density evaluations will be made where most of posterior mass is, which helps to scale in higher dimensions
  - better Markov chains are more efficient per density evaluation

MCMC summary

- Construct a Markov chain which has desired stationary distribution
  - most of the density evaluations will be made where most of posterior mass is, which helps to scale in higher dimensions
  - better Markov chains are more efficient per density evaluation
- MCMC estimate is biased towards the initial value
  - this bias can be non-negligible especially for short chains
  - the bias can be reduced by discarding the initial part of the chain
  - convergence diagnostics help to decide when the bias can be expected to be negligible

MCMC summary

- Construct a Markov chain which has desired stationary distribution
  - most of the density evaluations will be made where most of posterior mass is, which helps to scale in higher dimensions
  - better Markov chains are more efficient per density evaluation
- MCMC estimate is biased towards the initial value
  - this bias can be non-negligible especially for short chains
  - the bias can be reduced by discarding the initial part of the chain
  - convergence diagnostics help to decide when the bias can be expected to be negligible
- MCMC draws are correlated in time, but CLT holds (given finite variance)
  - effective sample size estimates help to decide how many correlated draws are needed

MCMC summary

- Construct a Markov chain which has desired stationary distribution
  - most of the density evaluations will be made where most of posterior mass is, which helps to scale in higher dimensions
  - better Markov chains are more efficient per density evaluation
- MCMC estimate is biased towards the initial value
  - this bias can be non-negligible especially for short chains
  - the bias can be reduced by discarding the initial part of the chain
  - convergence diagnostics help to decide when the bias can be expected to be negligible
- MCMC draws are correlated in time, but CLT holds (given finite variance)
  - effective sample size estimates help to decide how many correlated draws are needed
- Probabilistic programming frameworks
  - provide efficient MCMC algorithms that work well without manual tuning for many posterior distributions (more next week)