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 $\widehat{R}$
- 11.5 Effective number of simulation draws (important)
    - effective sample size N_{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 N_{eff}

It's all about expectations

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

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

It's all about expectations

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

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

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

It's all about expectations

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

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

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

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

It's all about expectations

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

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

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

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

- Grid (equal spacing) evaluation with self-normalization

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

It's all about expectations

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

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

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

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

- Grid (equal spacing) evaluation with self-normalization

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

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

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

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 less than 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 less than 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

- Andrey Markov proved weak law of large numbers and central limit theorem for certain dependent-random sequences, which were later named Markov chains

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
- 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
- The probability of each event depends only on the state attained in the previous event (or finite number of previous events)
- Markov's one example was the sequence of letters in Pushkin's novel "Yevgeniy Onegin"

Markov chain

- Example of a simple Markov chain on black board

Markov chain Monte Carlo (MCMC)

- Produce draws $\theta^{(t)}$ given $\theta^{(t-1)}$ from a Markov chain, which has been constructed so that its equilibrium distribution is $p(\theta|y)$

Markov chain Monte Carlo (MCMC)

- Produce draws $\theta^{(t)}$ given $\theta^{(t-1)}$ from a Markov chain, which has been constructed so that its equilibrium distribution is $p(\theta|y)$
  - + generic
  - + combine sequence of easier Monte Carlo draws to form a Markov chain

Markov chain Monte Carlo (MCMC)

- Produce draws $\theta^{(t)}$ given $\theta^{(t-1)}$ from a Markov chain, which has been constructed so that its equilibrium distribution is $p(\theta|y)$
  - + 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)

- Produce draws $\theta^{(t)}$ given $\theta^{(t-1)}$ from a Markov chain, which has been constructed so that its equilibrium distribution is $p(\theta|y)$
  - \+ 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 chain Monte Carlo (MCMC)

- Produce draws $\theta^{(t)}$ given $\theta^{(t-1)}$ from a Markov chain, which has been constructed so that its equilibrium distribution is $p(\theta|y)$
  - \+ 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
  - \- 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 $\theta^{(t-1)}$

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

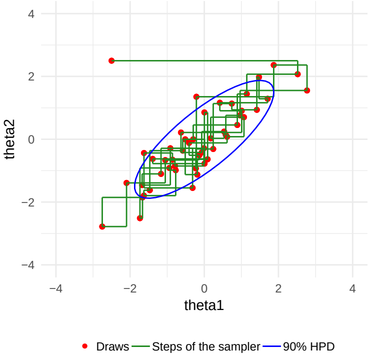
- Chain has to be initialized with some starting point θ^0
- Transition distribution $T_t(\theta^t|\theta^{t-1})$ (may depend on t)
- By choosing a suitable transition distribution, the stationary distribution of Markov chain is $p(\theta|y)$

Gibbs sampling

- Alternate sampling from 1D conditional distributions
  - 1D is easy even if no conjugate prior and analytic posterior

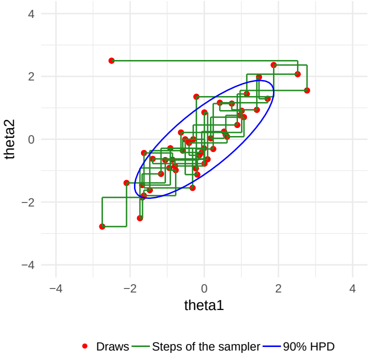
Gibbs sampling

- Alternate sampling from 1D conditional distributions
  - 1D is easy even if no conjugate prior and analytic posterior
- demo11_1



Gibbs sampling

- Alternate sampling from 1D conditional distributions
  - 1D is easy even if no conjugate prior and analytic posterior
- demo11_1



- Basic algorithm

$$
\begin{aligned}
\text{sample } \theta_j^t \text{ from} \quad & p(\theta_j|\theta_{-j}^{t-1}, y), \\
\text{where} \quad & \theta_{-j}^{t-1} = (\theta_1^t, \ldots, \theta_{j-1}^t, \theta_{j+1}^{t-1}, \ldots, \theta_d^{t-1})
\end{aligned}
$$

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

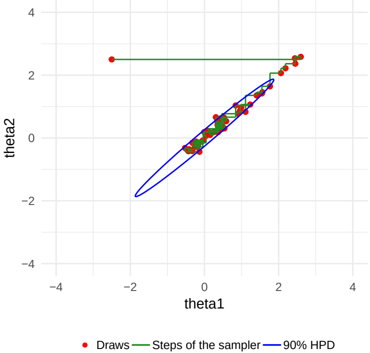
- With *conditionally* conjugate priors, the sampling from the conditional distributions is easy for wide range of models
  - BUGS/WinBUGS/OpenBUGS/JAGS
- No algorithm parameters to tune
  (cf. proposal distribution in Metropolis algorithm)
- For not so easy conditionals, use e.g. inverse-CDF

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?
  http://elevanth.org/blog/2017/11/28/build-a-better-markov-chain/

Metropolis algorithm

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

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

Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \ldots$
     (a) pick a proposal θ^* from the proposal distribution $J_t(\theta^*|\theta^{t-1})$. Proposal distribution has to be symmetric, i.e. $J_t(\theta_a|\theta_b) = J_t(\theta_b|\theta_a)$, for all θ_a, θ_b
     (b) calculate acceptance ratio
     $$r = \frac{p(\theta^*|y)}{p(\theta^{t-1}|y)}$$
     (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 $J_t(\theta^*|\theta^{t-1})$. Proposal distribution has to be symmetric, i.e. $J_t(\theta_a|\theta_b) = J_t(\theta_b|\theta_a)$, for all θ_a, θ_b
     (b) calculate acceptance ratio
     $$r = \frac{p(\theta^*|y)}{p(\theta^{t-1}|y)}$$
     (c) set
     $$\theta^t = \begin{cases} \theta^* & \text{with probability } \min(r, 1) \\ \theta^{t-1} & \text{otherwise} \end{cases}$$
     ie, if $p(\theta^*|y) > p(\theta^{t-1})$ accept the proposal always and otherwise reject 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 $J_t(\theta^*|\theta^{t-1})$. Proposal distribution has to be symmetric, i.e. $J_t(\theta_a|\theta_b) = J_t(\theta_b|\theta_a)$, for all θ_a, θ_b
     (b) calculate acceptance ratio
     $$r = \frac{p(\theta^*|y)}{p(\theta^{t-1}|y)}$$
     (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 $J_t(\theta^*|\theta^{t-1})$. Proposal distribution has to be symmetric, i.e. $J_t(\theta_a|\theta_b) = J_t(\theta_b|\theta_a)$, for all θ_a, θ_b
     (b) calculate acceptance ratio

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

     (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 $U(0, 1)$

Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \dots$
     (a) pick a proposal θ^* from the proposal distribution $J_t(\theta^*|\theta^{t-1})$.
     Proposal distribution has to be symmetric, i.e.
     $J_t(\theta_a|\theta_b) = J_t(\theta_b|\theta_a)$, for all θ_a, θ_b
     (b) calculate acceptance ratio
     $$r = \frac{p(\theta^*|y)}{p(\theta^{t-1}|y)}$$
     (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
    $U(0, 1)$
  - instead of $p(\theta|y)$, unnormalized $q(\theta|y)$ 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 $J_t(\theta^*|\theta^{t-1}) = \mathrm{N}(\theta^*|\theta^{t-1}, \sigma_p^2)$
- Demo
  http://elevanth.org/blog/2017/11/28/build-a-better-markov-chain/

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 distrbution (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 distrbution (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 distrbution (except for trivial exceptions)

Why Metropolis algorithm works

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

Why Metropolis algorithm works

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

Why Metropolis algorithm works

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

Why Metropolis algorithm works

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

Why Metropolis algorithm works

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

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

Why Metropolis algorithm works

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

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

Metropolis-Hastings algorithm

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

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

Metropolis-Hastings algorithm

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

Metropolis-Hastings algorithm

- Ideal proposal distribution is the distribution itself
  - $J(\theta^*|\theta) \equiv p(\theta^*|y)$ 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
  - $J(\theta^*|\theta) \equiv p(\theta^*|y)$ 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

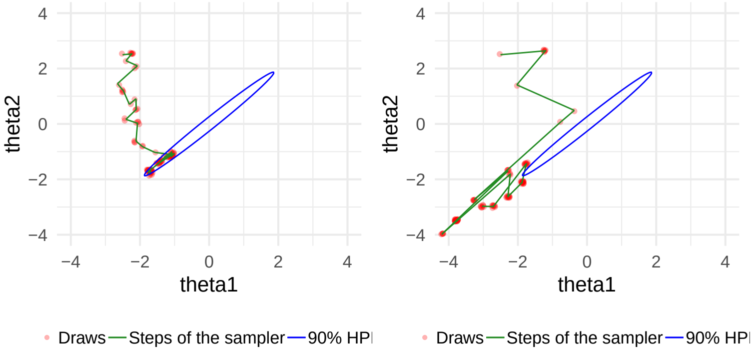
- Ideal proposal distribution is the distribution itself
  - $J(\theta^*|\theta) \equiv p(\theta^*|y)$ for all θ
  - acceptance probability is 1
  - independent draws
  - not usually feasible
- Good proposal distribution resembles the target distribution
  - if the shape of the target distribution is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

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



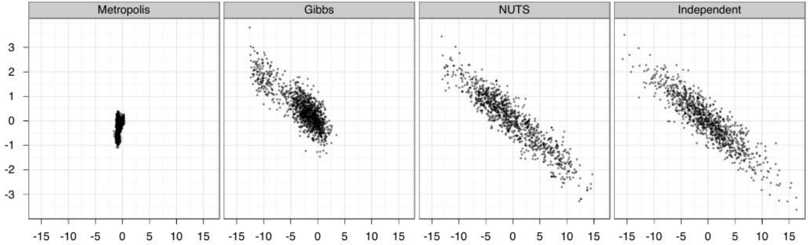
Dynamic Hamiltonian Monte Carlo and NUTS

- 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
    - Demo http://elevanth.org/blog/2017/11/28/build-a-better-markov-chain/

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)



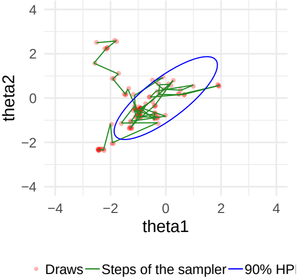
Source: Jonah Gabry

Warm-up and convergence diagnostics

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

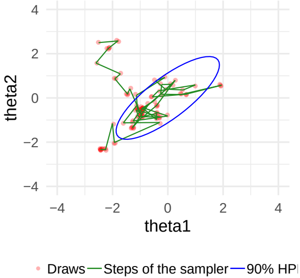
Warm-up and convergence diagnostics

- Asymptotically chain spends the $\alpha\%$ of time where $\alpha\%$ 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

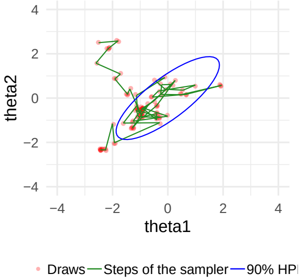
- Asymptotically chain spends the $\alpha\%$ of time where $\alpha\%$ posterior mass is
  - but in finite time the initial part of the chain may be non-representative and lower error of the estimate can be obtained by throwing it away



- 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 $\alpha\%$ of time where $\alpha\%$ 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
  - Do we get samples from the target distribution?

MCMC draws are dependent

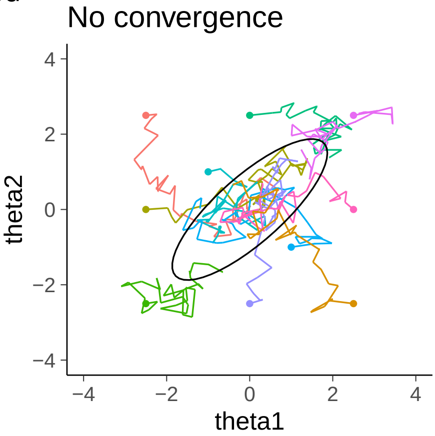
- Monte Carlo estimates still valid (central limit theorem holds)

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

- Estimation of Monte Carlo error is more difficult
  - evaluation of *effective* sample size

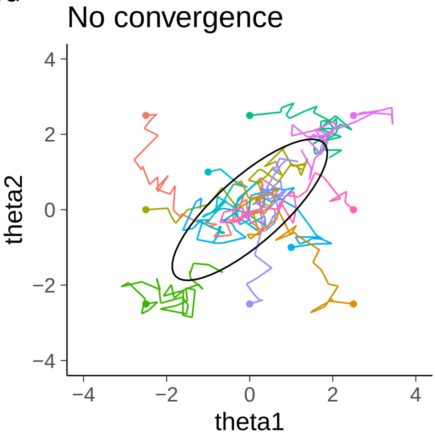
Several chains

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



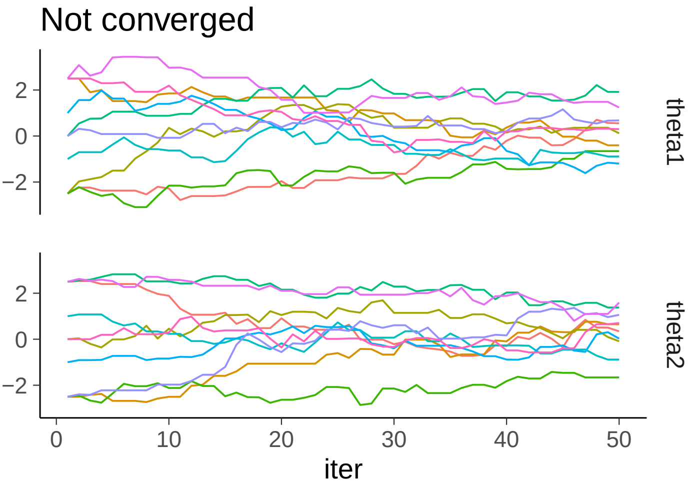
Several chains

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

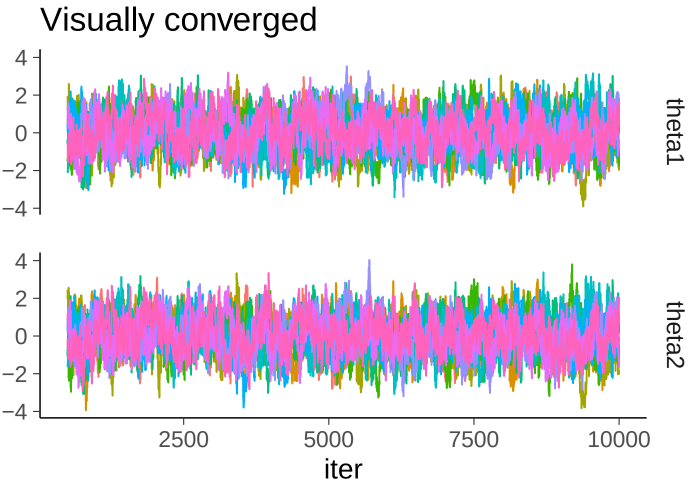


- Remove draws from the beginning of the chains and run chains long enough so that it is not possible to distinguish where each chain started and the chains are well mixed

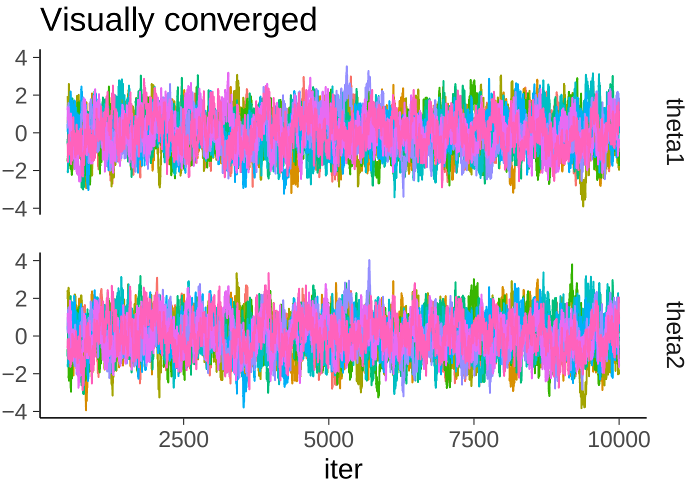
Several chains



Several chains



Several chains



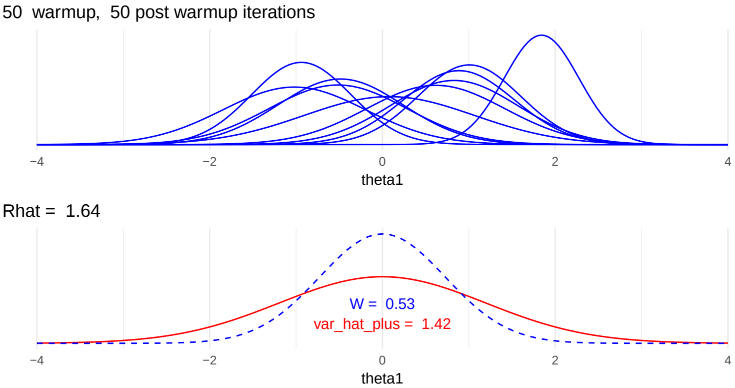
Visual convergence check is not sufficient

$\widehat{R}$: 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

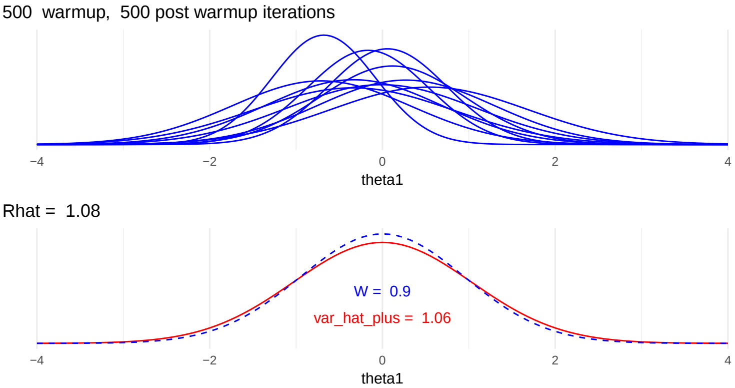


$\widehat{R}$: comparison of within and between variances of the chains

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

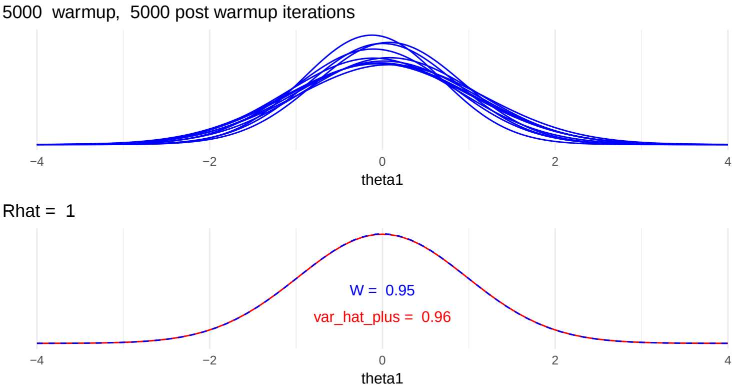
  W = within chain variance estimate

  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



$\widehat{R}$

- Within chains variance W

$$W = \frac{1}{m}\sum_{j=1}^{m} s_j^2, \text{ where } s_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\psi_{ij} - \bar{\psi}_{.j})^2$$

$\widehat{R}$

- Within chains variance W

$$W = \frac{1}{m}\sum_{j=1}^{m} s_j^2, \text{ where } s_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\psi_{ij} - \bar{\psi}_{.j})^2$$

- Between chains variance B

$$B = \frac{n}{m-1}\sum_{j=1}^{m}(\bar{\psi}_{.j} - \bar{\psi}_{..})^2, \text{ where } \bar{\psi}_{.j} = \frac{1}{n}\sum_{i=1}^{n}\psi_{ij}, \bar{\psi}_{..} = \frac{1}{m}\sum_{j=1}^{m}\bar{\psi}_{.j}$$

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

$\widehat{R}$

- Within chains variance W

$$W = \frac{1}{m}\sum_{j=1}^{m} s_j^2, \text{ where } s_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\psi_{ij} - \bar{\psi}_{.j})^2$$

- Between chains variance B

$$B = \frac{n}{m-1}\sum_{j=1}^{m}(\bar{\psi}_{.j} - \bar{\psi}_{..})^2, \text{ where } \bar{\psi}_{.j} = \frac{1}{n}\sum_{i=1}^{n}\psi_{ij}, \bar{\psi}_{..} = \frac{1}{m}\sum_{j=1}^{m}\bar{\psi}_{.j}$$

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

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

$$\widehat{\mathrm{var}}^{+}(\psi|y) = \frac{n-1}{n}W + \frac{1}{n}B$$

$\widehat{R}$

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

$$\widehat{\mathrm{var}}^{+}(\psi|y) = \frac{n-1}{n}W + \frac{1}{n}B$$

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

$\widehat{R}$

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

$$\widehat{\mathrm{var}}^{+}(\psi|y) = \frac{n-1}{n}W + \frac{1}{n}B$$

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

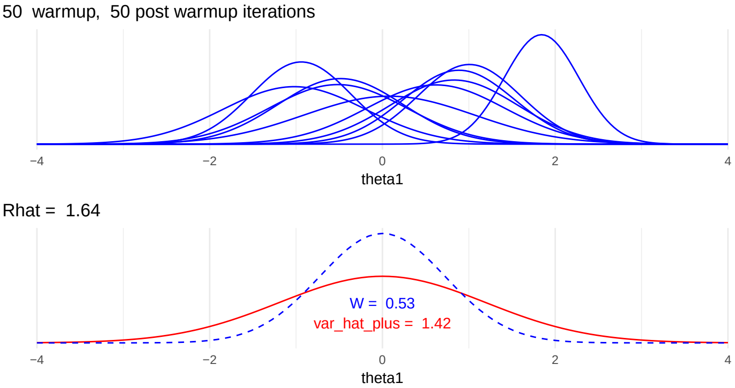
- Given finite n, W underestimates marginal posterior variance
  - single chains have not yet visited all points in the distribution
  - when $n \to \infty, \quad \mathrm{E}(W) \to \mathrm{var}(\psi|y)$

$\widehat{R}$

- Estimate total variance $\mathrm{var}(\psi|y)$ as a weighted mean of W and B
$$\widehat{\mathrm{var}}^{+}(\psi|y) = \frac{n-1}{n}W + \frac{1}{n}B$$
    - 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 $n \to \infty, \quad \mathrm{E}(W) \to \mathrm{var}(\psi|y)$
- As $\widehat{\mathrm{var}}^{+}(\psi|y)$ overestimates and W underestimates, compute
$$\widehat{R} = \sqrt{\frac{\widehat{\mathrm{var}}^{+}}{W}}$$

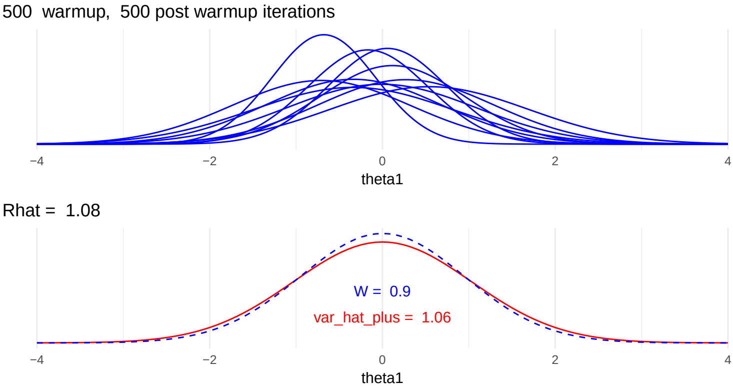
$\widehat{R}$

- 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



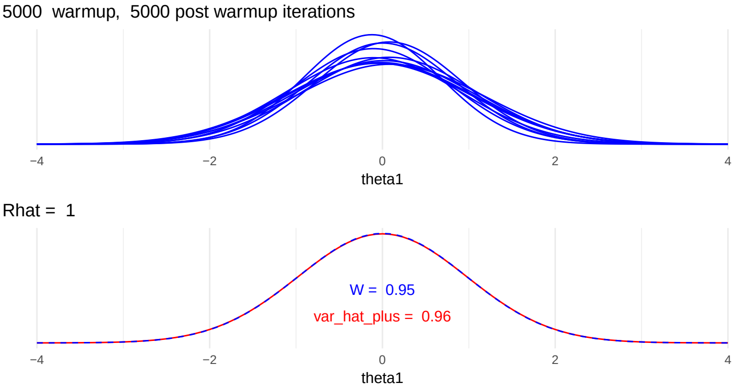
$\widehat{R}$

- 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



$\widehat{R}$

- 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 $n \to \infty$
- $R \to 1$, when $n \to \infty$
- if R is big (e.g., $R > 1.01$), keep sampling

$\widehat{R}$

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

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

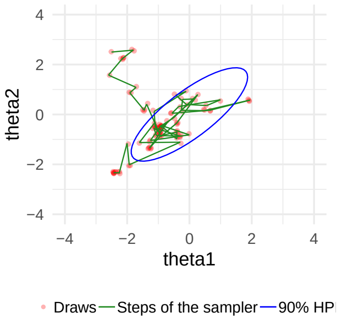
Split-$\widehat{R}$

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

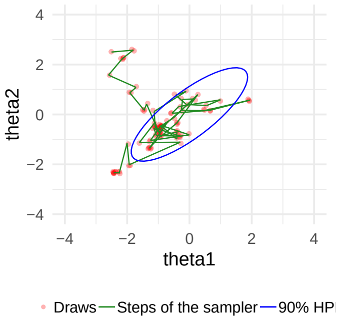
Time series analysis

- Auto correlation function
  - describes the correlation given a certain lag
  - can be used to compare efficiency of MCMC algorithms and parameterizations

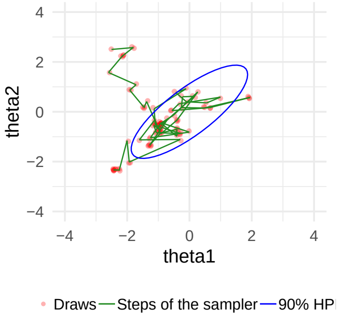
Auto correlation

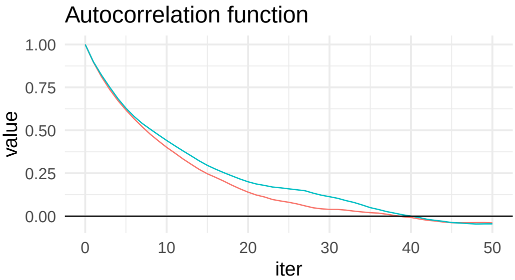


Auto correlation

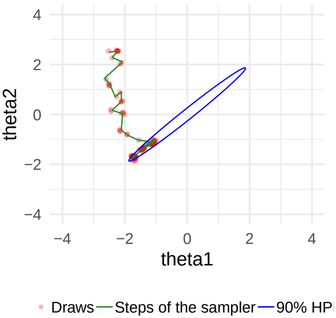


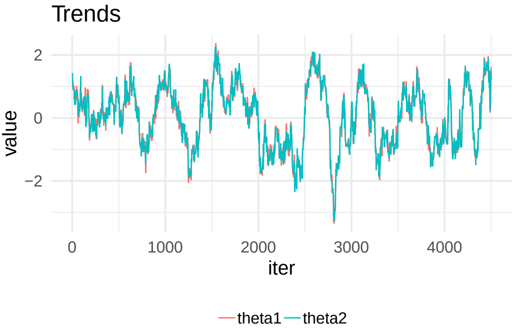
Auto correlation

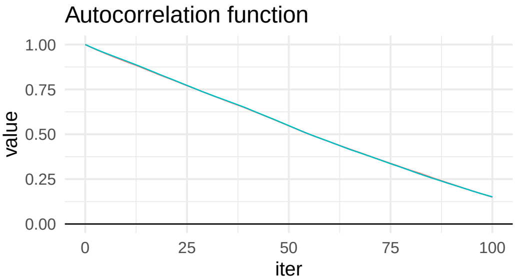




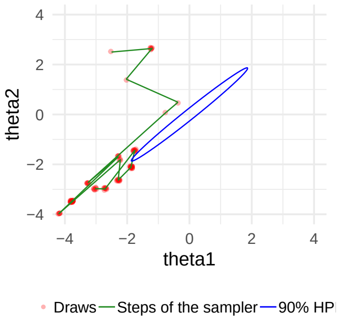
Auto correlation

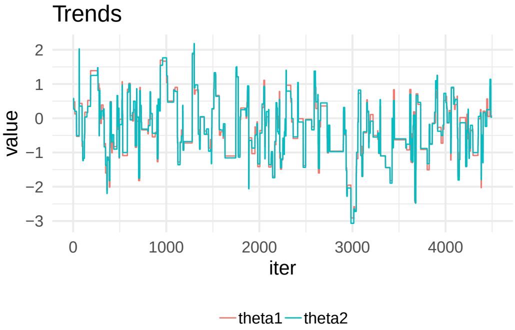


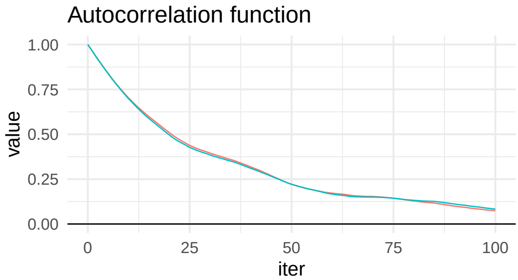




Auto correlation







Time series analysis

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

$$\mathrm{Var}[\bar{\theta}] = \frac{\sigma_\theta^2}{N/\tau}$$

where τ is sum of autocorrelations

  - τ describes how many dependent draws correspond to one independent sample
  - in BDA3 $N = nm$
  - $n_{\text{eff}} = nm/\tau$
  - BDA3 focuses on n_{eff} and not the Monte Carlo error directly

Time series analysis

- Estimation of the autocorrelation using several chains

$$\hat{\rho}_t = 1 - \frac{W - \frac{1}{M}\sum_{j=1}^{m} \hat{\rho}_{t,j}}{2\widehat{\text{var}}^{+}}$$

where $\hat{\rho}_{t,j}$ is autocorrelation at lag t for chain j

Time series analysis

- Estimation of the autocorrelation using several chains

$$\hat{\rho}_t = 1 - \frac{W - \frac{1}{M}\sum_{j=1}^{m} \hat{\rho}_{t,j}}{2\widehat{\text{var}}^{+}}$$

  where $\hat{\rho}_{t,j}$ is autocorrelation at lag t for chain j
- BDA3 has slightly different less accurate equation. The above equation is used in Stan 2.18+

Time series analysis

- Estimation of the autocorrelation using several chains

$$\hat{\rho}_t = 1 - \frac{W - \frac{1}{M}\sum_{j=1}^{m} \hat{\rho}_{t,j}}{2\widehat{\text{var}}^{+}}$$

  where $\hat{\rho}_{t,j}$ is autocorrelation at lag t for chain j
- BDA3 has slightly different less accurate equation. The above equation is used in Stan 2.18+
- Compared to usual method which computes the autocorrelation from a single chain, this 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
  - 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

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

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

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

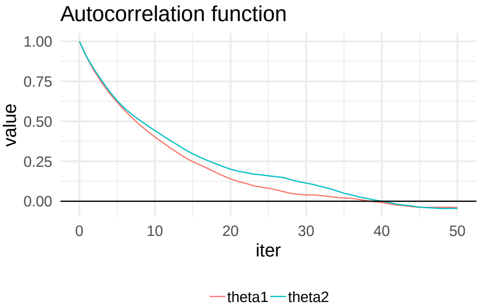
- As τ is estimated from a finite number of draws, it's expectation is overoptimistic
  - if $\tau > mn/20$ then the estimate is unreliable

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

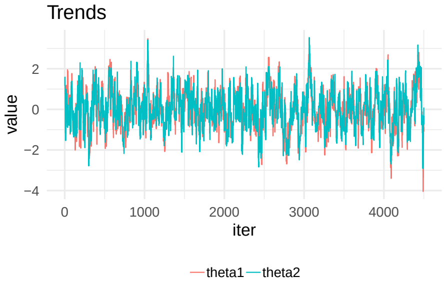


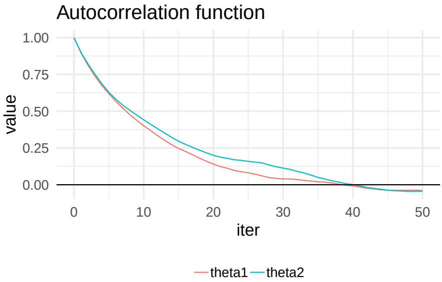
Effective sample size

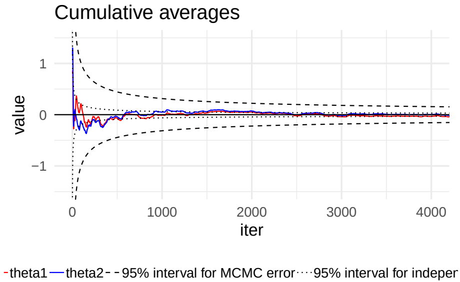
Effective number of draws $n_{\text{eff}} \approx N/\tau$

Effective sample size

Effective number of draws $n_{\text{eff}} \approx N/\tau$



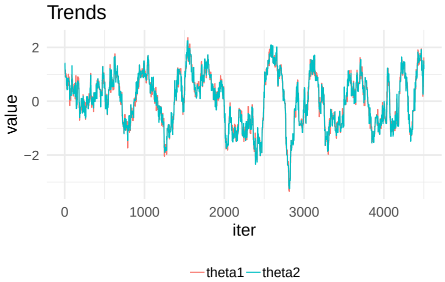


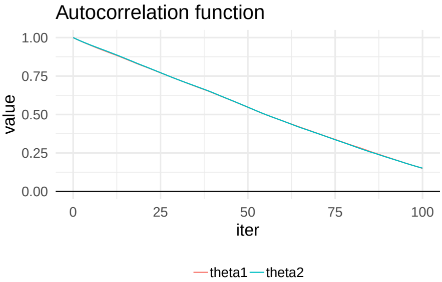


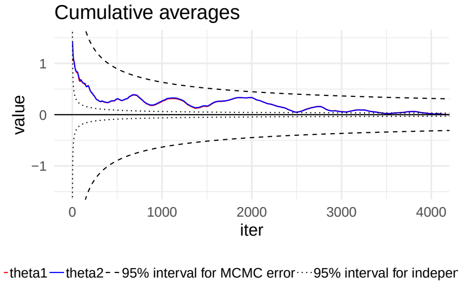
$$
\begin{aligned}
\tau &\approx 1 + 2\sum_{t=1}^{T} \hat{\rho}_t \\
&\approx 24
\end{aligned}
$$

Effective sample size

Effective number of draws $n_{\text{eff}} \approx N/\tau$





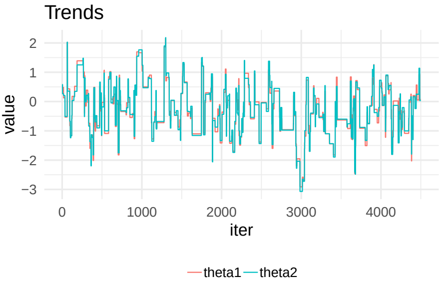


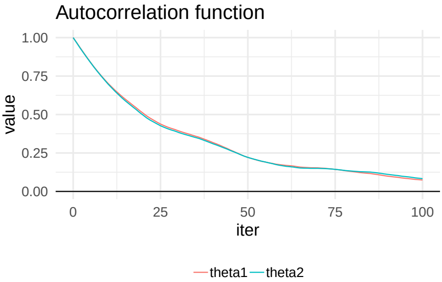
$$\tau \approx 1 + 2\sum_{t=1}^{T} \hat{\rho}_t$$

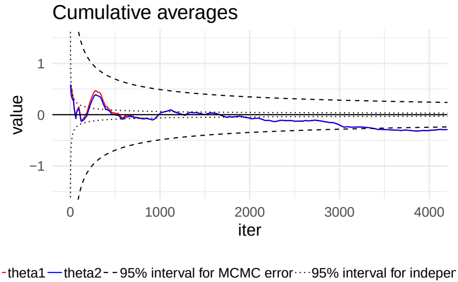
$$\approx 104$$

Effective sample size

Effective number of draws $n_{\text{eff}} \approx N/\tau$







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

$$\approx 63$$

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

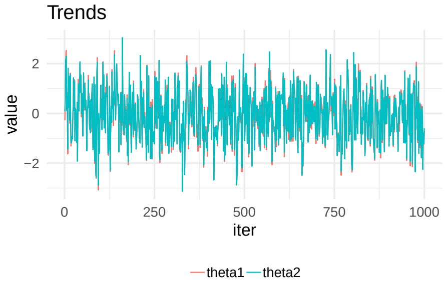
- Nonlinear dependencies
  - optimal proposal depends on location
- Funnels
  - optimal proposal depends on location
- Multimodal
  - difficult to move from one mode to another

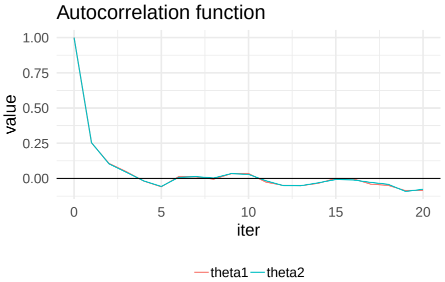
Problematic distributions

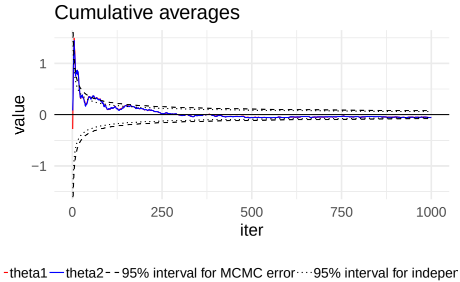
- Nonlinear dependencies
  - optimal proposal depends on location
- Funnels
  - optimal proposal depends on location
- Multimodal
  - difficult to move from one mode to another
- Long-tailed with non-finite variance and mean
  - central limit theorem for expectations does not hold

Next week: HMC, NUTS, and dynamic HMC

Effective number of draws $n_{\text{eff}} \approx N/\tau$







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

$$\approx 1.6$$

Further diagnostics

- Dynamic HMC/NUTS has additional diagnostics
  - divergences
  - tree depth exceedences
  - fraction of missing information