UPPSALA
UNIVERSITET

Bayesian Statistics and Data Analysis
Lecture 5

Måns Magnusson
Department of Statistics, Uppsala University
Thanks to Aki Vehtari, Aalto University

UPPSALA UNIVERSITET

It's all about expectations

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

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

UPPSALA UNIVERSITET

- Monte Carlo recap
- Markov Chain Monte Carlo (MCMC)
  - Gibbs sampling
  - Metropolis-Hastings
- Diagnostics
  - Warm-up
  - Convergence
  - S_{eff}, MCSE, and autocorrelation
- Difficult geometries

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 evaluate $p(y|\theta)p(\theta)$ 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 evaluate $p(y|\theta)p(\theta)$ 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

UPPSALA UNIVERSITET

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 evaluate $p(y|\theta)p(\theta)$ 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

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

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

UPPSALA UNIVERSITET

Monte Carlo

- Monte Carlo methods we have discussed so far
  - Inverse CDF works for 1D

UPPSALA UNIVERSITET

Monte Carlo

- Monte Carlo methods we have discussed so far
  - Inverse CDF works for 1D
  - Analytic transformations work for only certain distributions

UPPSALA UNIVERSITET

Monte Carlo

- Monte Carlo methods we have discussed so far
  - Inverse CDF works for 1D
  - Analytic transformations work for only certain distributions
  - Grid methods works in less than a few dimensions

UPPSALA UNIVERSITET

Monte Carlo

- Monte Carlo methods we have discussed so far
  - Inverse CDF works for 1D
  - Analytic transformations work for only certain distributions
  - Grid methods works in less than a few dimensions
  - Rejection sampling works mostly in 1D (truncation is a special case)

UPPSALA UNIVERSITET

- Monte Carlo recap
- Markov Chain Monte Carlo (MCMC)
  - Gibbs sampling
  - Metropolis-Hastings
- Diagnostics
  - Warm-up
  - Convergence
  - S_{eff}, MCSE, and autocorrelation
- Difficult geometries

Monte Carlo

- Monte Carlo methods we have discussed so far
  - Inverse CDF works for 1D
  - Analytic transformations work for only certain distributions
  - Grid methods 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

UPPSALA UNIVERSITET

Monte Carlo

- Monte Carlo methods we have discussed so far
  - Inverse CDF works for 1D
  - Analytic transformations work for only certain distributions
  - Grid methods 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?

UPPSALA UNIVERSITET

Monte Carlo

- Monte Carlo methods we have discussed so far
  - Inverse CDF works for 1D
  - Analytic transformations work for only certain distributions
  - Grid methods 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)

UPPSALA UNIVERSITET

Monte Carlo

- Monte Carlo methods we have discussed so far
  - Inverse CDF works for 1D
  - Analytic transformations work for only certain distributions
  - Grid methods 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*, next course)

Markov chains

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

Markov chains

- 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)

$$p(\theta_t|\theta_{t-1}, \theta_{t-2}, ...) = p(\theta_t|\theta_{t-1})$$

Markov chains

- 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)

$$p(\theta_t|\theta_{t-1}, \theta_{t-2}, ...) = p(\theta_t|\theta_{t-1})$$

- $T_t(\theta_t \mid \theta_{t-1}) \equiv p(\theta_t|\theta_{t-1})$ is usually refered to as the transition distribution

Markov chains

- 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)

$$p(\theta_t|\theta_{t-1}, \theta_{t-2}, \ldots) = p(\theta_t|\theta_{t-1})$$

- $T_t(\theta_t \mid \theta_{t-1}) \equiv p(\theta_t|\theta_{t-1})$ is usually refered to as the transition distribution
- Under some assumptions $p(\theta_t|\theta_{t-1})$ will converge (in total variation) to *one* stationary distribution $p(\theta)$

Markov chains

- 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)

$$p(\theta_t|\theta_{t-1}, \theta_{t-2}, \ldots) = p(\theta_t|\theta_{t-1})$$

- $T_t(\theta_t \mid \theta_{t-1}) \equiv p(\theta_t|\theta_{t-1})$ is usually refered to as the transition distribution
- Under some assumptions $p(\theta_t|\theta_{t-1})$ will converge (in total variation) to *one* stationary distribution $p(\theta)$
- Goal in MCMC: Construct a transition distribution with $p(\theta|y)$ as the stationary distribution

Markov chain Monte Carlo (MCMC)

- Produce draws $\theta_{(t)}$ given $\theta_{(t-1)}$ from a Markov chain, with stationary distribution $p(\theta|y)$

Markov chain Monte Carlo (MCMC)

- Produce draws $\theta_{(t)}$ given $\theta_{(t-1)}$ from a Markov chain, with stationary distribution $p(\theta|y)$
  + 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, with stationary distribution $p(\theta|y)$
  - + 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 $\alpha\%$ of time where $\alpha\%$ posterior mass is

Markov chain Monte Carlo (MCMC)

- Produce draws $\theta_{(t)}$ given $\theta_{(t-1)}$ from a Markov chain, with stationary distribution $p(\theta|y)$
  - \+ 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 $\alpha\%$ of time where $\alpha\%$ 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, with stationary distribution $p(\theta|y)$
  - \+ 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 $\alpha\%$ of time where $\alpha\%$ posterior mass is
  - \+ central limit theorem holds for expectations
  - \- draws are dependent
  - \- construction of efficient Markov chains is not always easy

Markov chain Monte Carlo (MCMC)

- Random variables $\theta_1, \theta_2, \ldots$ where θ_t depends only on the previous $\theta_{(t-1)}$

$$p(\theta_t | \theta_1, \ldots, \theta_{(t-1)}) = p(\theta_t | \theta_{(t-1)})$$

Markov chain Monte Carlo (MCMC)

- Random variables $\theta_1, \theta_2, \ldots$ where 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

Markov chain Monte Carlo (MCMC)

- Random variables $\theta_1, \theta_2, \ldots$ where 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)

Markov chain Monte Carlo (MCMC)

- Random variables $\theta_1, \theta_2, \ldots$ where 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)
- Choose a transition distribution so the stationary distribution of the Markov chain is $p(\theta | y)$

Gibbs sampling

- Alternate sampling from conditional distributions
- Basic algorithm, for $j \in \{1, ..., J\}$

$$\text{sample } \theta_{j,t} \text{ from } \quad p(\theta_j | \theta_{-j,t-1}, y),$$
$$\text{where } \quad \theta_{j,t-1} = (\theta_{1,J}, \ldots, \theta_{j-1,t}, \theta_{j+1,t-1}, \ldots, \theta_{t-1,J})$$

- Will converge (in total variation) to $p(\theta|y)$ as $N \to \infty$

Gibbs sampling

UPPSALA UNIVERSITET

- Alternate sampling from conditional distributions
- Basic algorithm, for $j \in \{1, ..., J\}$

$$\text{sample } \theta_{j,t} \text{ from } \quad p(\theta_j | \theta_{-j,t-1}, y),$$
$$\text{where } \quad \theta_{j,t-1} = (\theta_{1,J}, \ldots, \theta_{j-1,t}, \theta_{j+1,t-1}, \ldots, \theta_{t-1,J})$$

- Will converge (in total variation) to $p(\theta|y)$ as $N \to \infty$
- j can be multiple (blocked) parameters
- 1D sampling ($|j| = 1$) is generally easy

Gibbs sampling

- Alternate sampling from conditional distributions
- Basic algorithm, for $j \in \{1, ..., J\}$

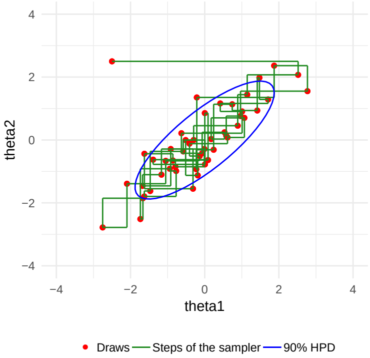
$$\begin{aligned} &\text{sample } \theta_{j,t} \text{ from } \quad p(\theta_j | \theta_{-j,t-1}, y), \\ &\quad \text{where } \quad \theta_{j,t-1} = (\theta_{1,J}, \ldots, \theta_{j-1,t}, \theta_{j+1,t-1}, \ldots, \theta_{t-1,J}) \end{aligned}$$

- Will converge (in total variation) to $p(\theta|y)$ as $N \to \infty$
- j can be multiple (blocked) parameters
- 1D sampling ($|j| = 1$) is generally easy
- Related to the (stochastic) EM algorithm

UPPSALA UNIVERSITET

- Monte Carlo recap
- Markov Chain Monte Carlo (MCMC)
  - Gibbs sampling
  - Metropolis-Hastings
- Diagnostics
  - Warm-up
  - Convergence
  - S_{eff}, MCSE, and autocorrelation
- Difficult geometries

Gibbs sampling



demo

UPPSALA UNIVERSITET

Gibbs sampling

- With *conditionally* conjugate priors, the sampling from the conditional distributions is easy for wide range of models

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

UPPSALA UNIVERSITET

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
- For not so easy conditionals, use e.g. inverse-CDF

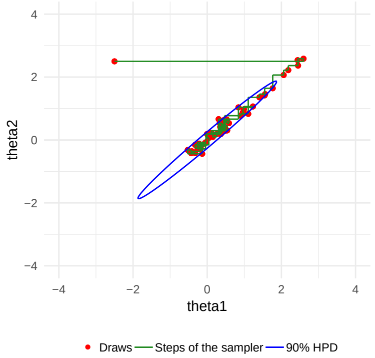
Gibbs sampling

- 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
- 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
- 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...

Gibbs sampling



demo

Sampling conditional vs joint

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

Sampling 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?

The Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \ldots$
     (a) pick a proposal θ^* from a 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

The Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \ldots$
     (a) pick a proposal θ^* from a 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

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

The Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \ldots$
     (a) pick a proposal θ^* from a 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
     $$\theta_t = \begin{cases} \theta^* & \text{with probability } \min(r, 1) \\ \theta_{t-1} & \text{otherwise} \end{cases}$$

The Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \ldots$
     (a) pick a proposal θ^* from a 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 $p(\theta^*|y) > p(\theta_{t-1}|y)$ accept the proposal always and otherwise accept the proposal with probability r

The Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \ldots$
     (a) pick a proposal θ^* from a 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

UPPSALA UNIVERSITET

The Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \ldots$
     (a) pick a proposal θ^* from a 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 $\mathcal{U}(0, 1)$

The Metropolis algorithm

- Algorithm
  1. starting point θ^0
  2. $t = 1, 2, \ldots$
     (a) pick a proposal θ^* from a 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 $\mathcal{U}(0, 1)$
  - $p(\theta^*|y)$ and $p(\theta_{t-1}|y)$ have the same normalization terms, and thus instead of $p(\cdot|y)$, unnormalized $q(\cdot|y)$ 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

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

  - proposal distribution $J_t(\theta^*|\theta_{t-1}) = \mathcal{N}(\theta^*|\theta_{t-1}, \sigma_p^2)$

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 $p(\theta|y)$

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

UPPSALA UNIVERSITET

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 as $T \to \infty$
      - 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 $p(\theta|y)$
   - 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 and 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 and 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 and 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 and 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 and 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

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

UPPSALA UNIVERSITET

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

UPPSALA UNIVERSITET

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 proposal distribution 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

demo

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 proposal distribution 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

demo

- Generic rule for rejection rate is 60-90%

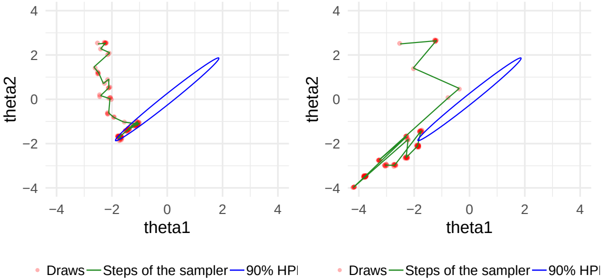
UPPSALA UNIVERSITET

Gibbs sampling as a special case

- 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

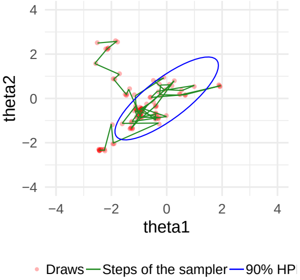


Warm-up

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

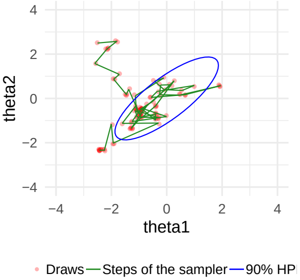
Warm-up

- Asymptotically chain spends the α% of time where α% posterior mass is
  - but in finite time the initial part of the Markov chain may be non-representative



Warm-up

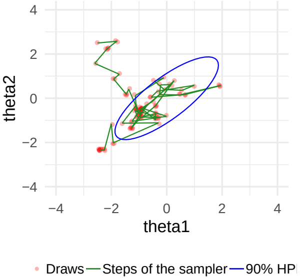
- Asymptotically chain spends the $\alpha\%$ of time where $\alpha\%$ posterior mass is
  - but in finite time the initial part of the Markov chain may be non-representative



- Warm-up period $=$ (non-representative) draws from the beginning of the Markov chain
  - remove warm-up before using samples for inference
  - warm-up may include also phase for adapting algorithm parameters

Warm-up

- Asymptotically chain spends the $\alpha\%$ of time where $\alpha\%$ posterior mass is
  - but in finite time the initial part of the Markov chain may be non-representative



- Warm-up period = (non-representative) draws from the beginning of the Markov chain
  - remove warm-up before using samples for inference
  - warm-up may include also phase for adapting algorithm parameters
- Also called burn-in

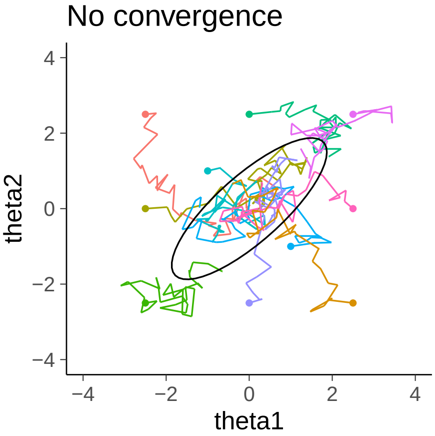
UPPSALA UNIVERSITET

Assesing convergence

- Several Markov chains make convergence diagnostics easier

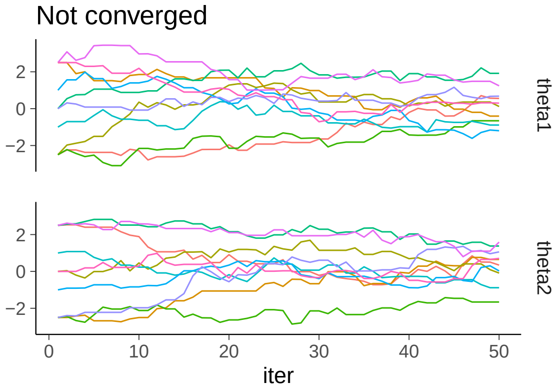
Assesing convergence

- Several Markov chains make convergence diagnostics easier
- Start chains from different starting points – preferably overdispersed



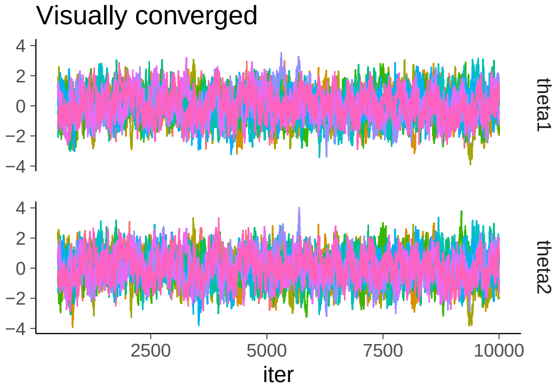
- Remove warm-up draws and run chains long enough

Several chains

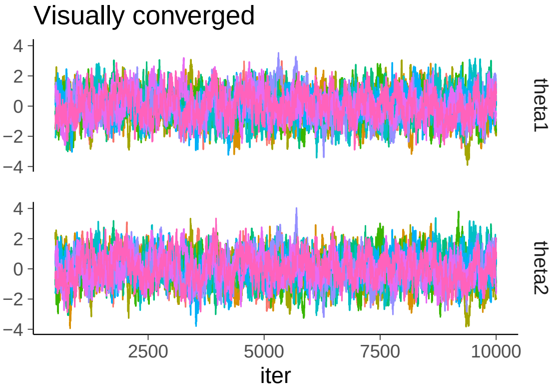


Several chains

- Monte Carlo recap
- Markov Chain Monte Carlo (MCMC)
  - Gibbs sampling
  - Metropolis-Hastings
- Diagnostics
  - Warm-up
  - Convergence
  - S_{eff}, MCSE, and autocorrelation
- Difficult geometries



Several chains



Visual convergence check is not sufficient

UPPSALA UNIVERSITET

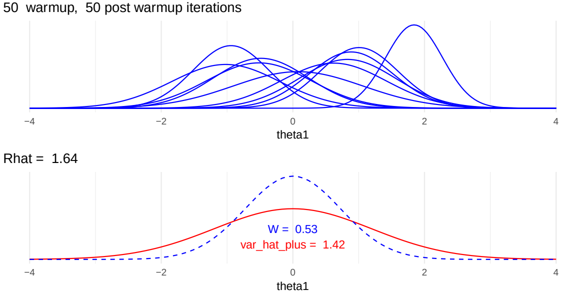
$\widehat{R}$: comparison of within and between variances of the chains

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

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

- Monte Carlo recap
- Markov Chain Monte Carlo (MCMC)
  - Gibbs sampling
  - Metropolis-Hastings
- Diagnostics
  - Warm-up
  - Convergence
  - S_{eff}, MCSE, and autocorrelation
- Difficult geometries

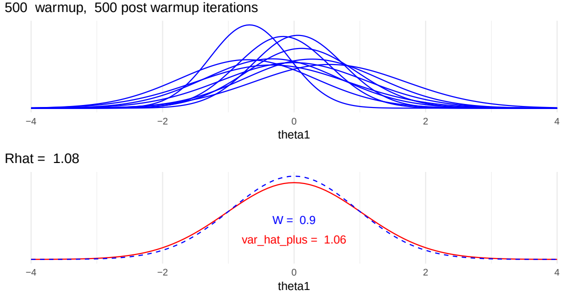
- $\widehat{R}$ or potential scale reduction factor (PSRF)
- Compare means and variances of the chains
  W = within chain variance estimate
  var_hat_plus = total variance estimate



UPPSALA UNIVERSITET

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

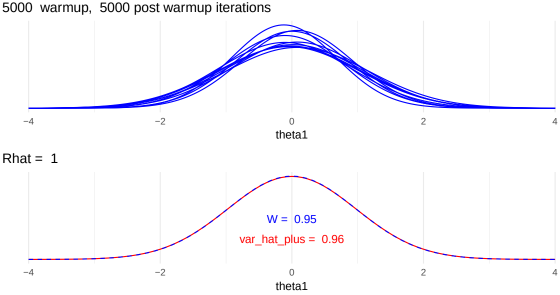
- $\widehat{R}$ or potential scale reduction factor (PSRF)
- Compare means and variances of the chains
  W = within chain variance estimate
  var_hat_plus = total variance estimate



UPPSALA UNIVERSITET

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

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



UPPSALA UNIVERSITET

$\widehat{R}$

- M chains, each having N draws

$\widehat{R}$

- M chains, each having N draws
- Within chains variance W

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

$\widehat{R}$

- M chains, each having N draws
- 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

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

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

$\widehat{R}$

- M chains, each having N draws
- 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

$\widehat{R}$

- M chains, each having N draws
- 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 $\mathrm{var}(\theta|y)$ as a weighted mean of W and B

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

$\widehat{R}$

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

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

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

$\widehat{R}$

- Estimate total variance $\mathrm{var}(\theta|y)$ as a weighted mean of W and B
$$\widehat{\mathrm{var}}^{+}(\theta|y) = \frac{N-1}{N} W + \frac{1}{N} B$$
  - 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 $N \to \infty, \quad \mathrm{E}(W) \to \mathrm{var}(\theta|y)$

$\widehat{R}$

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

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

  - 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 $N \to \infty, \quad \mathrm{E}(W) \to \mathrm{var}(\theta|y)$

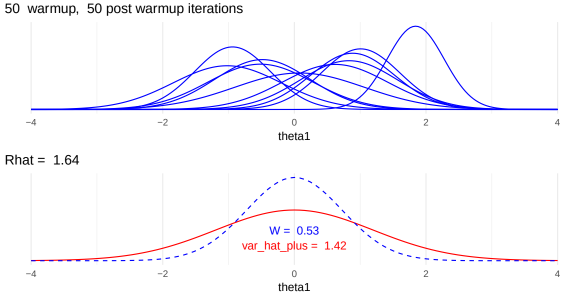
- As $\widehat{\mathrm{var}}^{+}(\theta|y)$ overestimates and W underestimates, compute

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

$\widehat{R}$

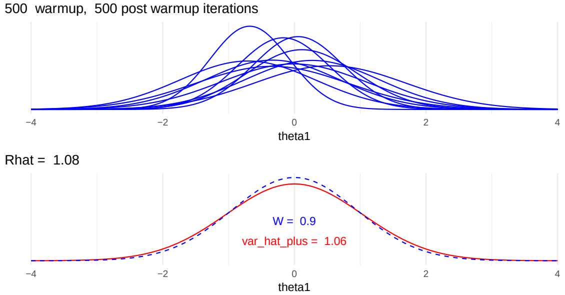
- Monte Carlo recap
- Markov Chain Monte Carlo (MCMC)
  - Gibbs sampling
  - Metropolis-Hastings
- Diagnostics
  - Warm-up
  - Convergence
  - S_{eff}, MCSE, and autocorrelation
- Difficult geometries

- 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



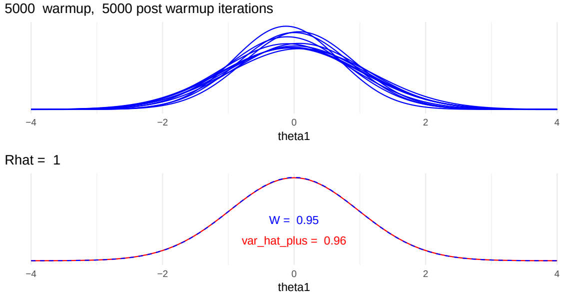
$\widehat{R}$

- 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



$\widehat{R}$

UPPSALA
UNIVERSITET

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

- Estimates how much the scale 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

$\widehat{R}$

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

- Estimates how much the scale 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

$\widehat{R}$

- Additional $\widehat{R}$ methods to assess convergence
- Split-$\widehat{R}$
  - Examines *mixing* and *stationarity* of chains
  - To examine stationarity chains are split to two parts: compare means and variances of the split chains

$\widehat{R}$

UPPSALA UNIVERSITET

- Additional $\widehat{R}$ methods to assess convergence
- Split-$\widehat{R}$
  - Examines *mixing* and *stationarity* of chains
  - To examine stationarity chains are split to two parts: compare means and variances of the split chains
- Rank normalized $\widehat{R}$
  - Does not requires that the target distribution has finite mean and variance

MCMC draws are dependent

- Monte Carlo estimates still valid (central limit theorem)

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

- Estimation of Monte Carlo error is more difficult
  - evaluation of *effective* sample size, S_{eff}.

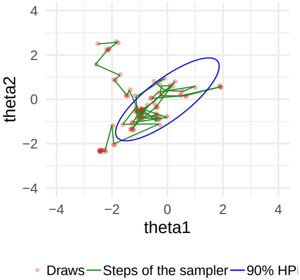
MCMC Autocorrelation

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

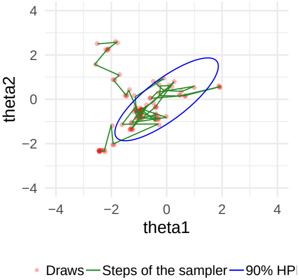
UPPSALA UNIVERSITET

- Monte Carlo recap
- Markov Chain Monte Carlo (MCMC)
  - Gibbs sampling
  - Metropolis-Hastings
- Diagnostics
  - Warm-up
  - Convergence
  - S_{eff}, MCSE, and autocorrelation
- Difficult geometries

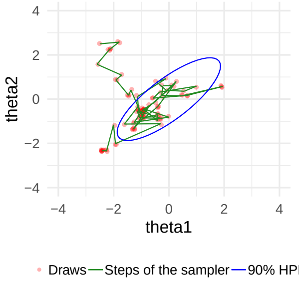
Autocorrelation

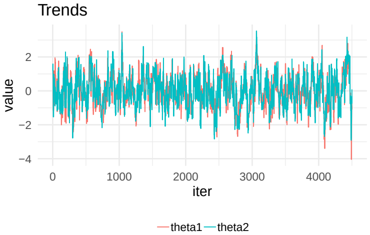


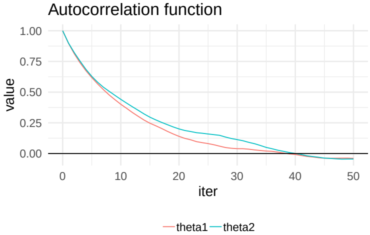
Autocorrelation



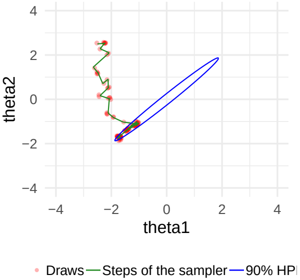
Autocorrelation

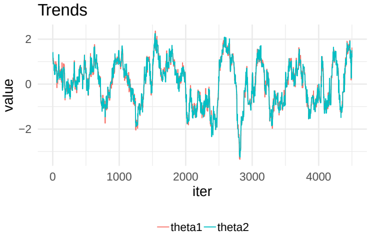


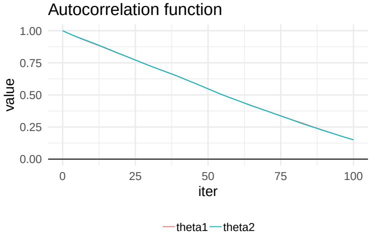




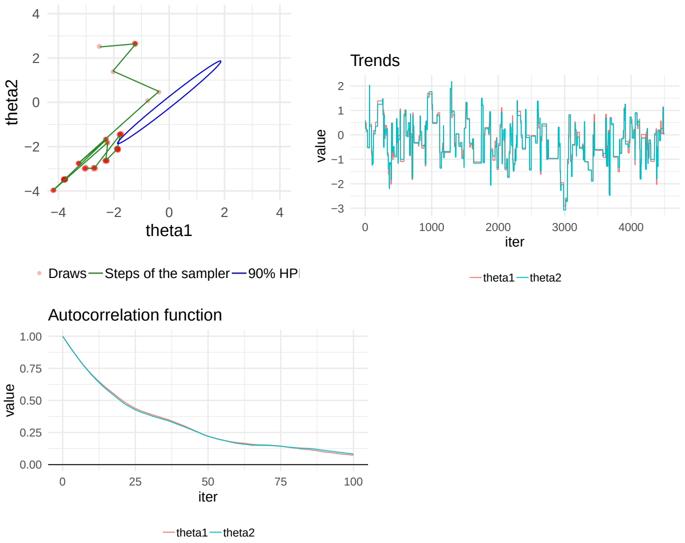
Autocorrelation







Autocorrelation



Autocorrelation

- The autocorrleation 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 $S_{\text{eff}} = S/\tau$, and τ is sum of autocorrelations

Autocorrelation

- The autocorrleation 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 $S_{\text{eff}} = S/\tau$, and τ is sum of autocorrelations
  - τ describes how many dependent draws correspond to one independent draw

Autocorrelation

- The autocorrleation 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 $S_{\text{eff}} = S/\tau$, and τ is sum of autocorrelations
  - τ describes how many dependent draws correspond to one independent draw
  - Here $S = NM$ (in BDA3 $N = nm$ and $n_{\text{eff}} = N/\tau$)

Autocorrelation

- The autocorrleation 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 $S_{\text{eff}} = S/\tau$, and τ is sum of autocorrelations
  - τ describes how many dependent draws correspond to one independent draw
  - Here $S = NM$ (in BDA3 $N = nm$ and $n_{\text{eff}} = N/\tau$)
  - BDA3 focuses on S_{eff} and not the Monte Carlo error directly

UPPSALA UNIVERSITET

Autocorrelation

- 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

Autocorrelation

- 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

- BDA3 has slightly different and less accurate equation. The above equation is used in Stan 2.18+

Autocorrelation

- Estimation of the autocorrelation using several chains

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

  where $\hat{\rho}_{n,m}$ is autocorrelation at lag n for chain m
- 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

UPPSALA UNIVERSITET

Estimating τ

- Estimation of τ

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

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

Estimating τ

- 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
- noise is larger for longer lags (less observations)

Estimating τ

- 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
- noise is larger for longer lags (less observations)
- less noisy estimate is obtained by truncating

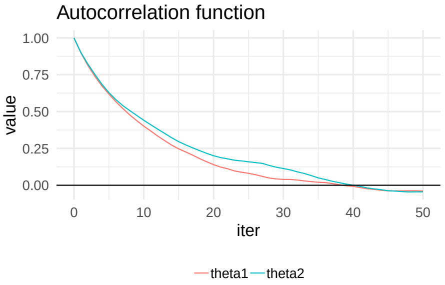
$$\hat{\tau} = 1 + 2\sum_{t=1}^{T} \hat{\rho}_t$$

Geyer's adaptive window estimator of τ

- Truncation T 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 of τ

- Truncation T 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, S_{eff}

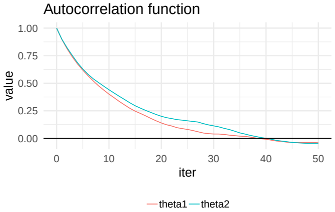
Effective sample size $\mathrm{ESS} = S_{\text{eff}} \approx S/\hat{\tau}$

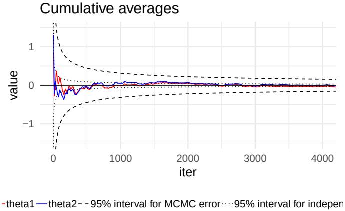
UPPSALA UNIVERSITET

- Monte Carlo recap
- Markov Chain Monte Carlo (MCMC)
  - Gibbs sampling
  - Metropolis-Hastings
- Diagnostics
  - Warm-up
  - Convergence
  - S_{eff}, MCSE, and autocorrelation
- Difficult geometries

Effective sample size, S_{eff}

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



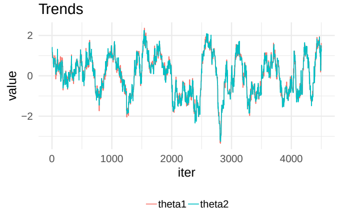


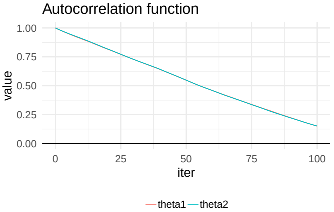
$$\hat{\tau} = 1 + 2\sum_{t=1}^{T} \hat{\rho}_t$$

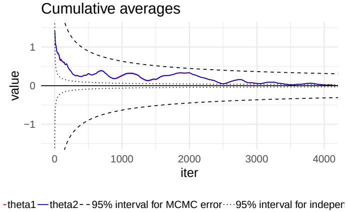
$$\approx 24$$

Effective sample size, S_{eff}

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





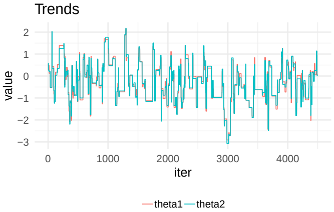


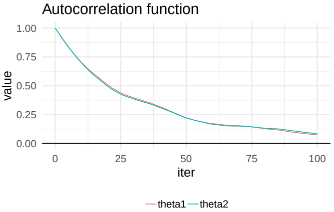
$$\hat{\tau} = 1 + 2\sum_{t=1}^{T} \hat{\rho}_t$$

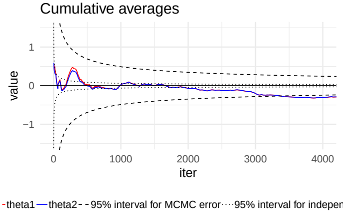
$$\approx 104$$

Effective sample size, S_{eff}

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







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

$$\approx 63$$

Problematic distributions

- Nonlinear dependencies
  - optimal proposal depends on location

UPPSALA UNIVERSITET

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

demo