



UPPSALA  
UNIVERSITET

# Bayesian Statistics and Data Analysis

## Lecture 6

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

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan



# Recap: MCMC, Gibbs and Metropolis

---

- Markov Chain Monte Carlo
  - A **transition distribution**  $T(\theta_0 \rightarrow \theta_1)$  with a unique **stationary distribution**
  - Target: setup  $T$  so that  $p(\theta|y)$  is the stationary distribution

- **MCMC recap**
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan



# Recap: MCMC, Gibbs and Metropolis

---

- Markov Chain Monte Carlo
  - A **transition distribution**  $T(\theta_0 \rightarrow \theta_1)$  with a unique **stationary distribution**
  - Target: setup  $T$  so that  $p(\theta|y)$  is the stationary distribution
  - + generic

- **MCMC recap**
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan



# Recap: MCMC, Gibbs and Metropolis

---

- Markov Chain Monte Carlo
  - A **transition distribution**  $T(\theta_0 \rightarrow \theta_1)$  with a unique **stationary distribution**
  - Target: setup  $T$  so that  $p(\theta|y)$  is the stationary distribution
- + generic
  - generates dependent draws (inefficiencies/low  $S_{\text{eff}}$ )
  - need to assess convergence to  $p(\theta|y)$

- **MCMC recap**
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Recap: MCMC, Gibbs and Metropolis

- Markov Chain Monte Carlo
  - A **transition distribution**  $T(\theta_0 \rightarrow \theta_1)$  with a unique **stationary distribution**
  - Target: setup  $T$  so that  $p(\theta|y)$  is the stationary distribution
  - + generic
    - generates dependent draws (inefficiencies/low  $S_{\text{eff}}$ )
    - need to assess convergence to  $p(\theta|y)$
- Gibbs sampling
  - Conditional (or block) sampling of  $\theta$

$$\theta_j \sim p(\theta_j | \theta_{-j}, y)$$



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Recap: MCMC, Gibbs and Metropolis

- Markov Chain Monte Carlo
  - A **transition distribution**  $T(\theta_0 \rightarrow \theta_1)$  with a unique **stationary distribution**
  - Target: setup  $T$  so that  $p(\theta|y)$  is the stationary distribution
  - + generic
    - generates dependent draws (inefficiencies/low  $S_{\text{eff}}$ )
    - need to assess convergence to  $p(\theta|y)$
- Gibbs sampling
  - Conditional (or block) sampling of  $\theta$

$$\theta_j \sim p(\theta_j | \theta_{-j}, y)$$

+ Often easy to construct



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Recap: MCMC, Gibbs and Metropolis

- Markov Chain Monte Carlo
  - A **transition distribution**  $T(\theta_0 \rightarrow \theta_1)$  with a unique **stationary distribution**
  - Target: setup  $T$  so that  $p(\theta|y)$  is the stationary distribution
  - + generic
    - generates dependent draws (inefficiencies/low  $S_{\text{eff}}$ )
    - need to assess convergence to  $p(\theta|y)$
- Gibbs sampling
  - Conditional (or block) sampling of  $\theta$ 
$$\theta_j \sim p(\theta_j | \theta_{-j}, y)$$
  - + Often easy to construct
    - Inefficient if posterior has correlated parameters



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Recap: MCMC, Gibbs and Metropolis

- Markov Chain Monte Carlo
  - A **transition distribution**  $T(\theta_0 \rightarrow \theta_1)$  with a unique **stationary distribution**
  - Target: setup  $T$  so that  $p(\theta|y)$  is the stationary distribution
  - + generic
    - generates dependent draws (inefficiencies/low  $S_{\text{eff}}$ )
    - need to assess convergence to  $p(\theta|y)$
- Gibbs sampling
  - Conditional (or block) sampling of  $\theta$ 
$$\theta_j \sim p(\theta_j | \theta_{-j}, y)$$
  - + Often easy to construct
    - Inefficient if posterior has correlated parameters
- Metropolis(-Hastings) sampling
  - Joint (or block) sampling of  $\theta$
  - Proposal distribution  $J$  (i.e.  $T$ )





- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Recap: MCMC, Gibbs and Metropolis

- Markov Chain Monte Carlo
  - A **transition distribution**  $T(\theta_0 \rightarrow \theta_1)$  with a unique **stationary distribution**
  - Target: setup  $T$  so that  $p(\theta|y)$  is the stationary distribution
    - + generic
      - generates dependent draws (inefficiencies/low  $S_{\text{eff}}$ )
      - need to assess convergence to  $p(\theta|y)$
- Gibbs sampling
  - Conditional (or block) sampling of  $\theta$ 
$$\theta_j \sim p(\theta_j | \theta_{-j}, y)$$
    - + Often easy to construct
      - Inefficient if posterior has correlated parameters
- Metropolis(-Hastings) sampling
  - Joint (or block) sampling of  $\theta$
  - Proposal distribution  $J$  (i.e.  $T$ )
  - + better for correlated posteriors



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Recap: MCMC, Gibbs and Metropolis

- Markov Chain Monte Carlo
  - A **transition distribution**  $T(\theta_0 \rightarrow \theta_1)$  with a unique **stationary distribution**
  - Target: setup  $T$  so that  $p(\theta|y)$  is the stationary distribution
  - + generic
    - generates dependent draws (inefficiencies/low  $S_{\text{eff}}$ )
    - need to assess convergence to  $p(\theta|y)$
- Gibbs sampling
  - Conditional (or block) sampling of  $\theta$ 
$$\theta_j \sim p(\theta_j | \theta_{-j}, y)$$
  - + Often easy to construct
    - Inefficient if posterior has correlated parameters
- Metropolis(-Hastings) sampling
  - Joint (or block) sampling of  $\theta$
  - Proposal distribution  $J$  (i.e.  $T$ )
  - + better for correlated posteriors
    - scale need to be tuned for efficient sampling
    - hard to propose in high dimensions (many small steps or many rejections)



UPPSALA  
UNIVERSITET

- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Section 2

# Hamiltonian Monte Carlo



# Why Hamiltonian Monte Carlo?

---

- MCMC recap
  - **Hamiltonian Monte Carlo**
    - Dynamic HMC and NUTS
  - HMC diagnostics
  - Probabilistic Programming
  - Stan
- Want to build an **efficient** Markov Chain
    - We want to sample jointly all  $\theta$
    - We know the **unnormalized** posterior  $q(\theta|y) = Z \cdot p(\theta|y)$ , where  $Z$  is the normalization constant.
    - Can we use this to create a good **proposal** distribution  $J$ ?



# Why Hamiltonian Monte Carlo?

---

- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- Want to build an **efficient** Markov Chain
  - We want to sample jointly all  $\theta$
  - We know the **unnormalized** posterior  $q(\theta|y) = Z \cdot p(\theta|y)$ , where  $Z$  is the normalization constant.
  - Can we use this to create a good **proposal** distribution  $J$ ?
  - **Hamiltonian Monte Carlo!**



# What is Hamiltonian Monte Carlo?

---

- Add momentum variables to our posterior (**canonical** distribution)

$$p(\psi, \theta|y) = p(\psi|\theta, y) \cdot p(\theta|y),$$

in practice we let  $p(\psi|\theta, y) = p(\psi)$

- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# What is Hamiltonian Monte Carlo?

---

- Add momentum variables to our posterior (**canonical** distribution)

$$p(\psi, \theta | y) = p(\psi | \theta, y) \cdot p(\theta | y),$$

in practice we let  $p(\psi | \theta, y) = p(\psi)$

- Idea from Physics (Mechanics):
  - $\theta$ : position
  - $\psi$ : momentum



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# What is Hamiltonian Monte Carlo?

- Add momentum variables to our posterior (**canonical** distribution)

$$p(\psi, \theta|y) = p(\psi|\theta, y) \cdot p(\theta|y),$$

in practice we let  $p(\psi|\theta, y) = p(\psi)$

- Idea from Physics (Mechanics):
  - $\theta$ : position
  - $\psi$ : momentum
- Define the Hamiltonian as

$$H(\psi, \theta) = -\log(p(\psi)) - \log(p(\theta|y)) \quad (1)$$

$$= K(\psi) + V(\theta), \quad (2)$$

where  $K(\psi)$  is the **kinetic** energy and  $V(\theta)$  is the **potential** energy





# What is Hamiltonian Monte Carlo?

- Hamiltonian Dynamics (**preserve energy**)

$$\frac{d\theta}{dt} = \frac{\partial H}{\partial \psi} = \frac{\partial K}{\partial \psi} \quad (3)$$

$$\frac{d\psi}{dt} = -\frac{\partial H}{\partial \theta} = -\frac{\partial V}{\partial \theta} \quad (4)$$

- Let  $V(\theta) = -\log(q(\theta|y)) = -\log p(\theta) - \log p(y|\theta)$
- Let  $\psi \sim N(0, M)$  where  $M$  is the **mass matrix**
- Hence,  $K(\psi) = -\log p(\psi) \propto 0.5\psi^T M^{-1}\psi + C$

- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan



# What is Hamiltonian Monte Carlo?

- Hamiltonian Dynamics (**preserve energy**)

$$\frac{d\theta}{dt} = \frac{\partial H}{\partial \psi} = \frac{\partial K}{\partial \psi} \quad (3)$$

$$\frac{d\psi}{dt} = -\frac{\partial H}{\partial \theta} = -\frac{\partial V}{\partial \theta} \quad (4)$$

- Let  $V(\theta) = -\log(q(\theta|y)) = -\log p(\theta) - \log p(y|\theta)$
- Let  $\psi \sim N(0, M)$  where  $M$  is the **mass matrix**
- Hence,  $K(\psi) = -\log p(\psi) \propto 0.5\psi^T M^{-1}\psi + C$
- We need to choose  $M$  in a smart way.
  1. Ideally,  $M^{-1} = \text{Cov}(\theta|y)$
  2. In practice,  $M^{-1} = V(\theta|y)$



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# The leapfrog integrator

---

- We want to simulate Hamiltonian dynamics

$$\frac{d\theta}{dt} = M^{-1}\psi \quad (5)$$

$$\frac{d\psi}{dt} = \frac{\partial \log q(\theta|y)}{\partial \theta} \quad (6)$$

- A **discrete** approximation: **the leapfrog integrator**



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# The leapfrog integrator

- We want to simulate Hamiltonian dynamics

$$\frac{d\theta}{dt} = M^{-1}\psi \quad (5)$$

$$\frac{d\psi}{dt} = \frac{\partial \log q(\theta|y)}{\partial \theta} \quad (6)$$

- A **discrete** approximation: **the leapfrog integrator**
- We take  $L$  leapfrog steps with step size  $\epsilon$  as

$$\psi \leftarrow \psi + \frac{1}{2}\epsilon \frac{d \log q(\theta|y)}{d\theta} \quad (7)$$

$$\theta \leftarrow \theta + \epsilon M^{-1}\psi \quad (8)$$

$$\psi \leftarrow \psi + \frac{1}{2}\epsilon \frac{d \log q(\theta|y)}{d\theta} \quad (9)$$



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# The leapfrog integrator

- We want to simulate Hamiltonian dynamics

$$\frac{d\theta}{dt} = M^{-1}\psi \quad (5)$$

$$\frac{d\psi}{dt} = \frac{\partial \log q(\theta|y)}{\partial \theta} \quad (6)$$

- A **discrete** approximation: **the leapfrog integrator**
- We take  $L$  leapfrog steps with step size  $\epsilon$  as

$$\psi \leftarrow \psi + \frac{1}{2}\epsilon \frac{d \log q(\theta|y)}{d\theta} \quad (7)$$

$$\theta \leftarrow \theta + \epsilon M^{-1}\psi \quad (8)$$

$$\psi \leftarrow \psi + \frac{1}{2}\epsilon \frac{d \log q(\theta|y)}{d\theta} \quad (9)$$

- Discretization introduce a error depending on  $\epsilon$  (not  $L\epsilon$ )



UPPSALA  
UNIVERSITET

# Hamiltonian Monte Carlo Algorithm

---

## 1. Sample momentum

$$\psi_0 \sim N(0, M)$$

- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan



UPPSALA  
UNIVERSITET

# Hamiltonian Monte Carlo Algorithm

---

## 1. Sample momentum

$$\psi_0 \sim N(0, M)$$

## 2. Simulate values $(\theta^*, \psi^*)$ using the leapfrog integrator $L$ steps with stepsize $\epsilon$ , starting from $(\theta_{t-1}, \psi_0)$

- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## 1. Sample momentum

$$\psi_0 \sim N(0, M)$$

- ## 2. Simulate values $(\theta^*, \psi^*)$ using the leapfrog integrator $L$ steps with stepsize $\epsilon$ , starting from $(\theta_{t-1}, \psi_0)$
- ## 3. Accept the proposed values $(\theta^*, \psi^*)$ with probability

$$r = \min \left( 1, \frac{q(\theta^*|y)}{q(\theta_{t-1}|y)} \frac{p(\psi^*)}{p(\psi_0)} \right)$$





UPPSALA  
UNIVERSITET

# Hamiltonian Monte Carlo

---

- Bivariate Normal HMC example

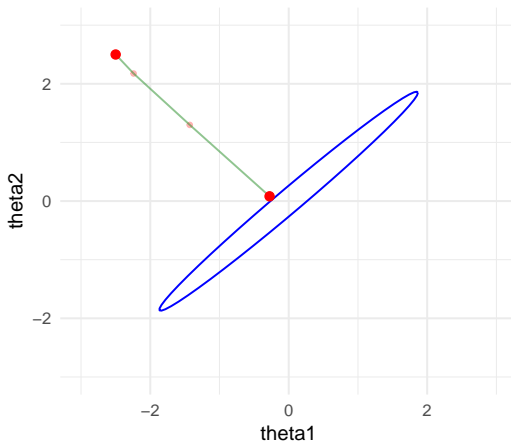
- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Hamiltonian Monte Carlo

- Bivariate Normal HMC example



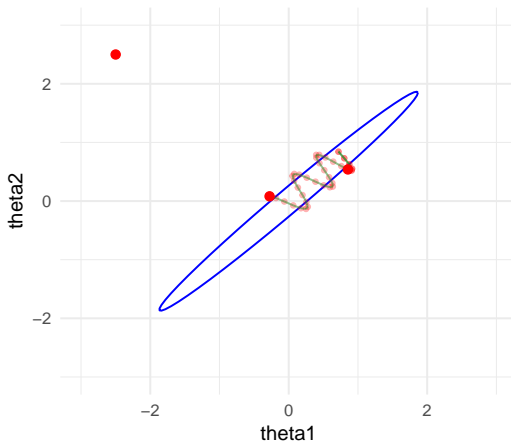
• Samples — Steps of the sampler — 90% HPD



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Hamiltonian Monte Carlo

- Bivariate Normal HMC example



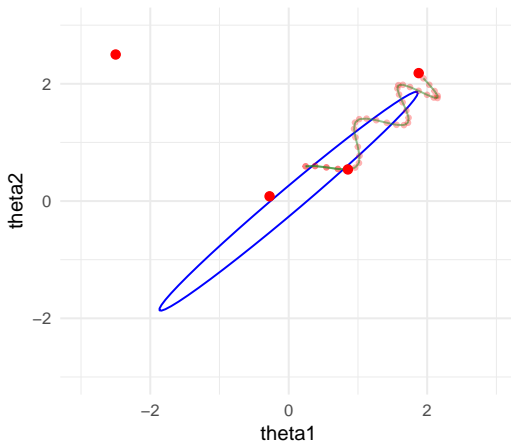
• Samples — Steps of the sampler — 90% HPD



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Hamiltonian Monte Carlo

- Bivariate Normal HMC example



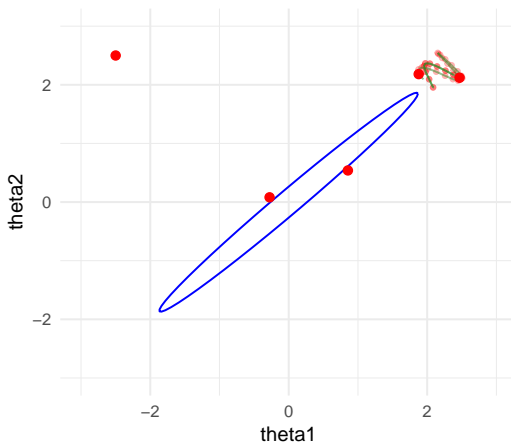
• Samples — Steps of the sampler — 90% HPD



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Hamiltonian Monte Carlo

- Bivariate Normal HMC example



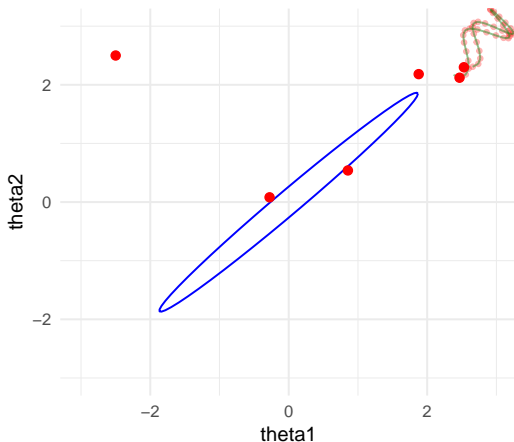
• Samples — Steps of the sampler — 90% HPD



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Hamiltonian Monte Carlo

- Bivariate Normal HMC example



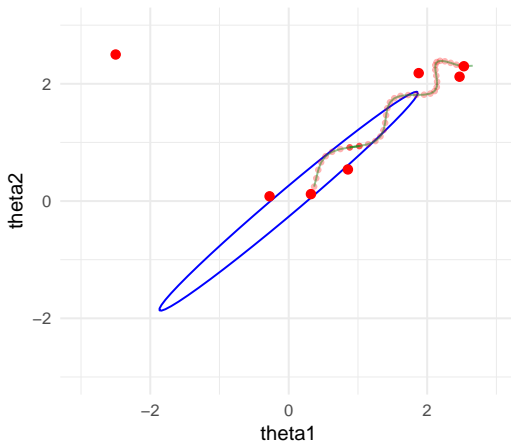
• Samples — Steps of the sampler — 90% HPD



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Hamiltonian Monte Carlo

- Bivariate Normal HMC example



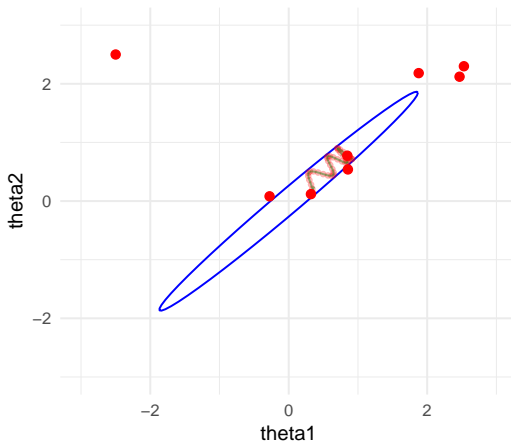
• Samples — Steps of the sampler — 90% HPD



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Hamiltonian Monte Carlo

- Bivariate Normal HMC example



• Samples — Steps of the sampler — 90% HPD

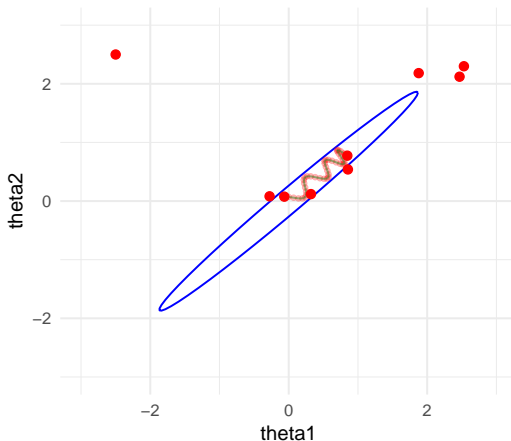




- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Hamiltonian Monte Carlo

- Bivariate Normal HMC example



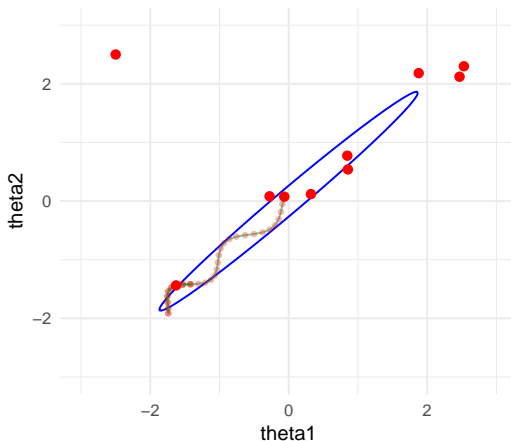
• Samples — Steps of the sampler — 90% HPD



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Hamiltonian Monte Carlo

- Bivariate Normal HMC example



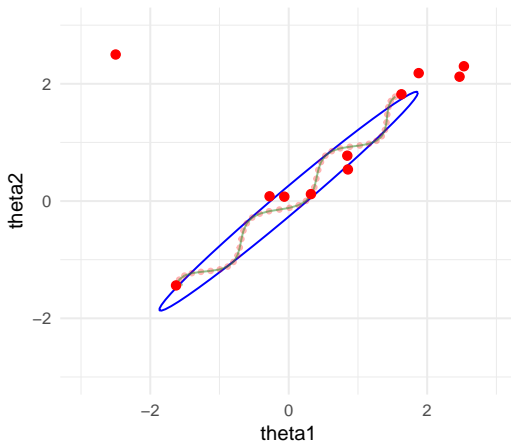
• Samples — Steps of the sampler — 90% HPD



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Hamiltonian Monte Carlo

- Bivariate Normal HMC example



• Samples — Steps of the sampler — 90% HPD

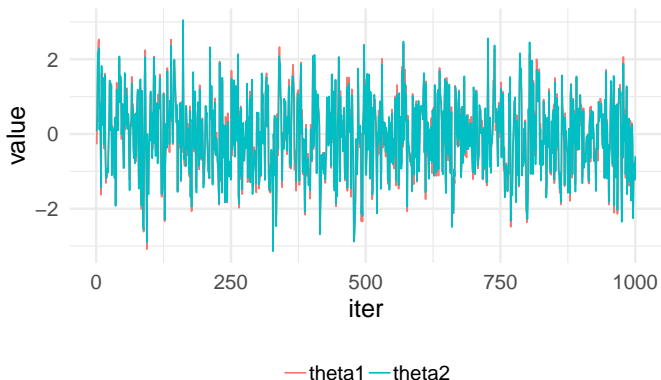


- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Hamiltonian Monte Carlo

- Bivariate Normal HMC example

## Trends



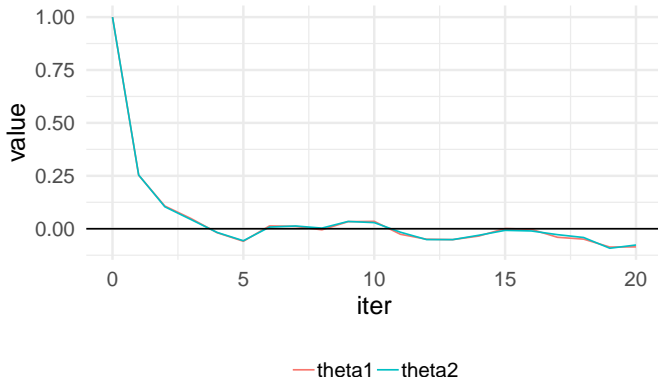


- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Hamiltonian Monte Carlo

- Bivariate Normal HMC example

## Autocorrelation function





UPPSALA  
UNIVERSITET

# Hamiltonian Monte Carlo Summary

---

- Parameters:

- $\epsilon$  step size

- MCMC recap

- Hamiltonian Monte Carlo

- Dynamic HMC and NUTS

- HMC diagnostics

- Probabilistic Programming

- Stan



UPPSALA  
UNIVERSITET

# Hamiltonian Monte Carlo Summary

---

- Parameters:

- $\epsilon$  step size

- $L$  leapfrog steps

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan



UPPSALA  
UNIVERSITET

# Hamiltonian Monte Carlo Summary

---

- Parameters:

- $\epsilon$  step size

- $L$  leapfrog steps

- $M$  mass matrix

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan





UPPSALA  
UNIVERSITET

# Hamiltonian Monte Carlo Summary

---

- Parameters:

- $\epsilon$  step size

- $L$  leapfrog steps

- $M$  mass matrix

- + Can be very efficient ( $S_{\text{eff}}$ )

- + Additional diagnostics

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- Parameters:

- $\epsilon$  step size

- $L$  leapfrog steps

- $M$  mass matrix

- + Can be very efficient ( $S_{\text{eff}}$ )

- + Additional diagnostics

- Can be difficult to tune (U-turns)
  - Bounded parameters needs handling
  - Ideally, we should adapt  $\epsilon L$
  - Costly to run each iteration ( $L \log$  density gradient evaluations)



- MCMC recap
- **Hamiltonian Monte Carlo**
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Hamiltonian Monte Carlo Summary

- Parameters:

- $\epsilon$  step size

- $L$  leapfrog steps

- $M$  mass matrix

- + Can be very efficient ( $S_{\text{eff}}$ )

- + Additional diagnostics

- Can be difficult to tune (U-turns)

- Bounded parameters needs handling

- Ideally, we should adapt  $\epsilon L$

- Costly to run each iteration ( $L \log$  density gradient evaluations)

demo



- **Goal:** Simplify/adapt the tuning of HMC
- **Dynamic** HMC refers to **dynamic** trajectory length of the leapfrog integrator (i.e.  $L$  is chosen on the fly)

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Dynamic HMC

- **Goal:** Simplify/adapt the tuning of HMC
- **Dynamic** HMC refers to **dynamic** trajectory length of the leapfrog integrator (i.e.  $L$  is chosen on the fly)
- The NUTS/dynamic algorithm:
  1. Grow a binary tree of leapfrog steps  $L$
  2. Grow (randomly) in two directions (to keep reversibility of Markov chain)
  3. Stop to grow tree when encounter a U-turn  
( $\theta_{t-1} - \theta_L$ )  $\cdot \psi$ )
  4. Sample one of the step at the trajectory (higher probability further away)



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Dynamic HMC

- **Goal:** Simplify/adapt the tuning of HMC
- **Dynamic** HMC refers to **dynamic** trajectory length of the leapfrog integrator (i.e.  $L$  is chosen on the fly)
- The NUTS/dynamic algorithm:
  1. Grow a binary tree of leapfrog steps  $L$
  2. Grow (randomly) in two directions (to keep reversibility of Markov chain)
  3. Stop to grow tree when encounter a U-turn  
( $\theta_{t-1} - \theta_L$ )  $\cdot \psi$
  4. Sample one of the step at the trajectory (higher probability further away)
- Dynamic simulation is discretized
  - small  $\epsilon$  gives accurate simulation, but requires more log density evaluations
  - large  $\epsilon$  reduces computation, but increases simulation error



UPPSALA  
UNIVERSITET

# Dynamic Hamiltonian Monte Carlo Summary

---

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- Parameters:
  - € step size



# Dynamic Hamiltonian Monte Carlo Summary

---

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- Parameters:
  - $\epsilon$  step size
  - $M$  mass matrix





# Dynamic Hamiltonian Monte Carlo Summary

---

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- Parameters:
  - $\epsilon$  step size
  - $M$  mass matrix
- + Can be very efficient ( $S_{\text{eff}}$ )
- + Additional diagnostics



# Dynamic Hamiltonian Monte Carlo Summary

---

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- Parameters:

- $\epsilon$  step size

- $M$  mass matrix

- + Can be very efficient ( $S_{\text{eff}}$ )

- + Additional diagnostics

- Bounded parameters needs handling

- Costly to run each iteration ( $L$  log density gradient evaluations)



# Dynamic Hamiltonian Monte Carlo Summary

---

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- Parameters:

- $\epsilon$  step size

- $M$  mass matrix

- + Can be very efficient ( $S_{\text{eff}}$ )

- + Additional diagnostics

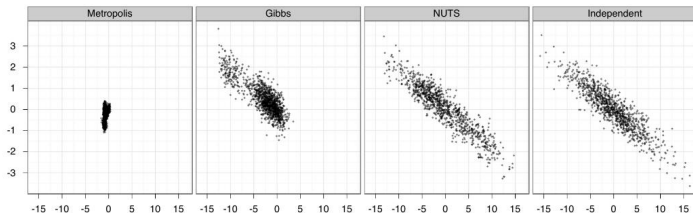
- Bounded parameters needs handling
  - Costly to run each iteration ( $L$  log density gradient evaluations)

demo



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

- MCMC recap
  - Hamiltonian Monte Carlo
    - Dynamic HMC and NUTS
  - HMC diagnostics
  - Probabilistic Programming
  - Stan
- 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



UPPSALA  
UNIVERSITET

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Section 3

# HMC diagnostics



UPPSALA  
UNIVERSITET

# Max tree depth

---

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- Dynamic HMC specific diagnostic
- Indicates inefficiency in sampling leading to higher autocorrelations and lower ESS ( $n_{\text{eff}}$ )
- Different parameterizations can help/matter



UPPSALA  
UNIVERSITET

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Divergences

---

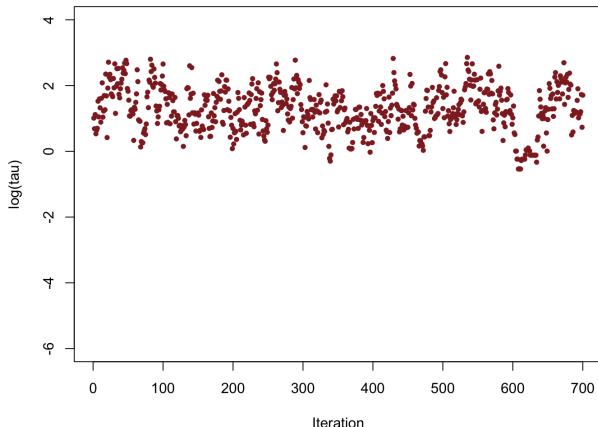
- HMC specific diagnostic
- indicates that Hamiltonian dynamic simulation has problems with unexpected fast changes in log-density
  - indicates possibility of biased estimates
- Different parameterizations matter
- See Betancourt (2017)



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Divergences

- HMC specific diagnostic
- indicates that Hamiltonian dynamic simulation has problems with unexpected fast changes in log-density
  - indicates possibility of biased estimates
- Different parameterizations matter
- See Betancourt (2017)



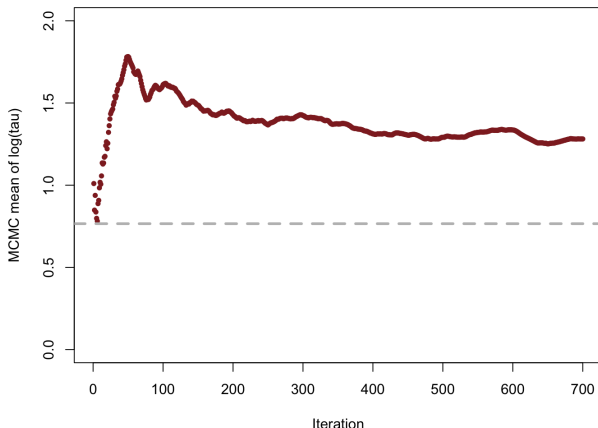




- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Divergences

- HMC specific diagnostic
- indicates that Hamiltonian dynamic simulation has problems with unexpected fast changes in log-density
  - indicates possibility of biased estimates
- Different parameterizations matter
- See Betancourt (2017)

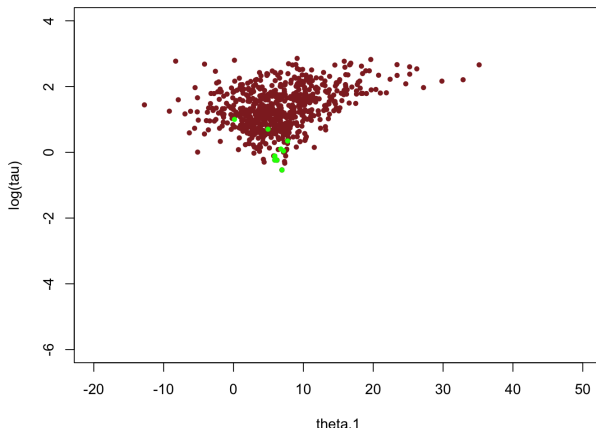




- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Divergences

- HMC specific diagnostic
- indicates that Hamiltonian dynamic simulation has problems with unexpected fast changes in log-density
  - indicates possibility of biased estimates
- Different parameterizations matter
- See Betancourt (2017)

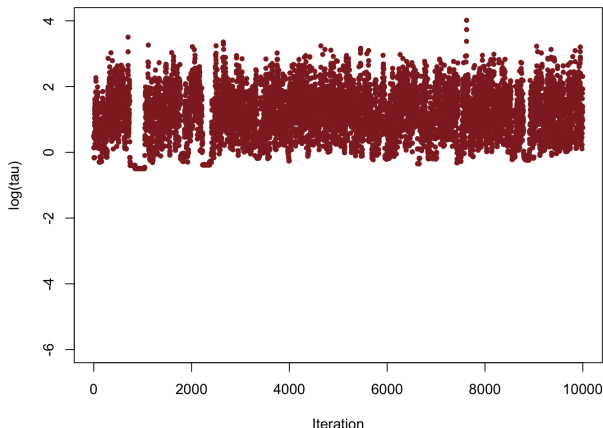




- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Divergences

- HMC specific diagnostic
- indicates that Hamiltonian dynamic simulation has problems with unexpected fast changes in log-density
  - indicates possibility of biased estimates
- Different parameterizations matter
- See Betancourt (2017)

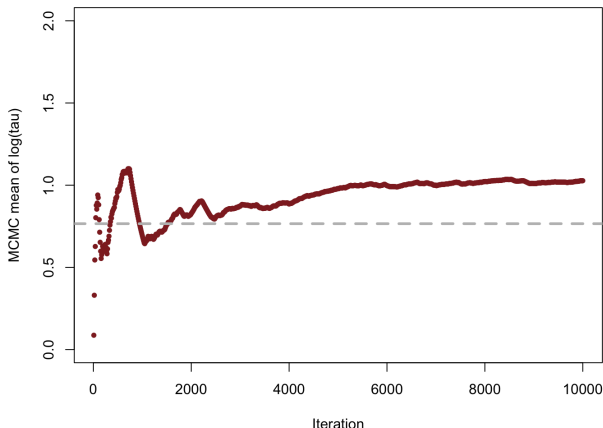




- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Divergences

- HMC specific diagnostic
- indicates that Hamiltonian dynamic simulation has problems with unexpected fast changes in log-density
  - indicates possibility of biased estimates
- Different parameterizations matter
- See Betancourt (2017)

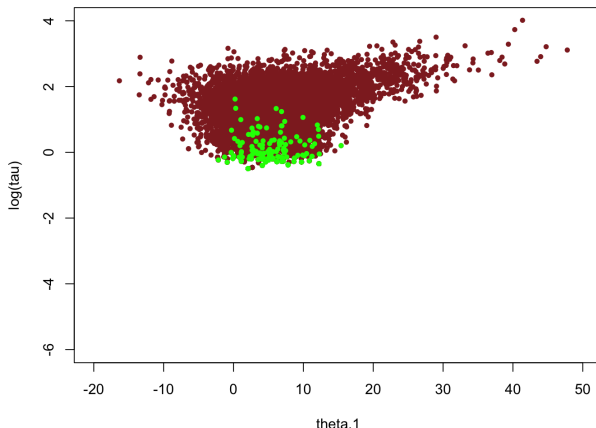




- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Divergences

- HMC specific diagnostic
- indicates that Hamiltonian dynamic simulation has problems with unexpected fast changes in log-density
  - indicates possibility of biased estimates
- Different parameterizations matter
- See Betancourt (2017)

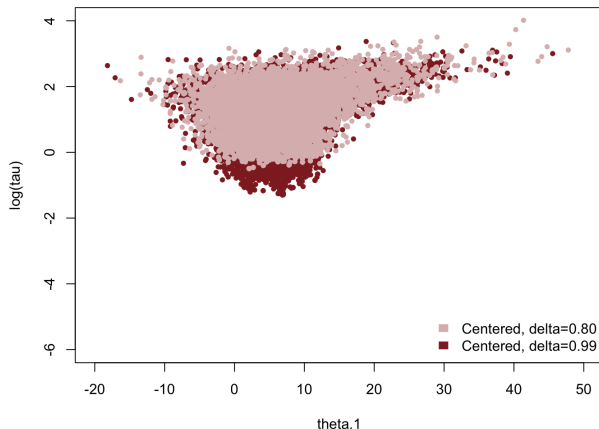




- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Divergences

- HMC specific diagnostic
- indicates that Hamiltonian dynamic simulation has problems with unexpected fast changes in log-density
  - indicates possibility of biased estimates
- Different parameterizations matter
- See Betancourt (2017)

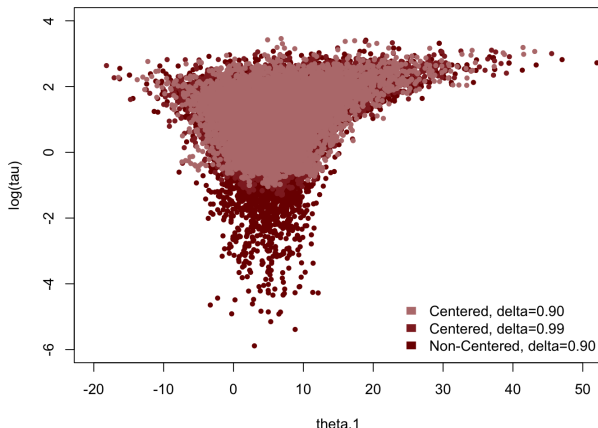




- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Divergences

- HMC specific diagnostic
- indicates that Hamiltonian dynamic simulation has problems with unexpected fast changes in log-density
  - indicates possibility of biased estimates
- Different parameterizations matter
- See Betancourt (2017)





UPPSALA  
UNIVERSITET

# Problematic distributions

---

- MCMC recap
  - Hamiltonian Monte Carlo
    - Dynamic HMC and NUTS
  - HMC diagnostics
  - Probabilistic Programming
  - Stan
- Nonlinear dependencies
    - simple mass matrix scaling doesn't help





- MCMC recap
  - Hamiltonian Monte Carlo
    - Dynamic HMC and NUTS
  - HMC diagnostics
  - Probabilistic Programming
  - Stan
- Nonlinear dependencies
    - simple mass matrix scaling doesn't help
  - Funnels
    - optimal step size depends on location



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- Nonlinear dependencies
  - simple mass matrix scaling doesn't help
- Funnels
  - optimal step size depends on location
- Multimodal
  - difficult to move from one mode to another



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- Nonlinear dependencies
  - simple mass matrix scaling doesn't help
- Funnels
  - optimal step size depends on location
- Multimodal
  - difficult to move from one mode to another
- Long-tailed with non-finite variance and mean
  - efficiency of exploration is reduced
  - central limit theorem doesn't hold for mean and variance



UPPSALA  
UNIVERSITET

## Extra (optional) material for HMC

---

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- Michael Betancourt (2018). A Conceptual Introduction to Hamiltonian Monte Carlo.  
<https://arxiv.org/abs/1701.02434>
- Michael Betancourt (2017). Diagnosing Biased Inference with Divergences.  
[https://mc-stan.org/users/documentation/case-studies/divergences\\_and\\_bias.html](https://mc-stan.org/users/documentation/case-studies/divergences_and_bias.html)



UPPSALA  
UNIVERSITET

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Section 4

# Probabilistic Programming



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# The Box process: Probabilistic modeling

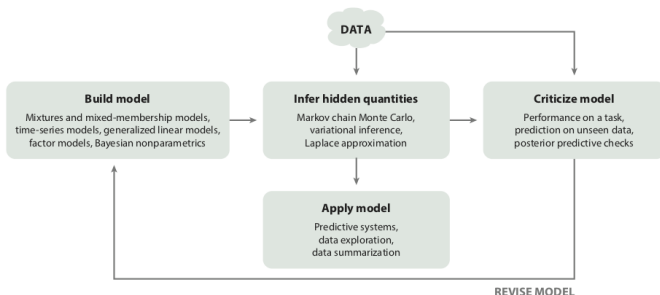


Figure: The Box approach (Box, 1976, Blei, 2014)



UPPSALA  
UNIVERSITET

# Probabilistic programming languages

---

- MCMC recap
  - Hamiltonian Monte Carlo
    - Dynamic HMC and NUTS
  - HMC diagnostics
  - Probabilistic Programming
  - Stan
- Wikipedia “A probabilistic programming language (PPL) is a programming language designed to describe probabilistic models and then perform inference in those models”



- MCMC recap
  - Hamiltonian Monte Carlo
    - Dynamic HMC and NUTS
  - HMC diagnostics
  - Probabilistic Programming
  - Stan
- Wikipedia “A probabilistic programming language (PPL) is a programming language designed to describe probabilistic models and then perform inference in those models”
  - To make probabilistic programming useful
    - easy workflow to build and revise models
    - inference has to be as automatic as possible
    - diagnostics for telling if the automatic inference doesn't work





- MCMC recap
  - Hamiltonian Monte Carlo
    - Dynamic HMC and NUTS
  - HMC diagnostics
  - Probabilistic Programming
  - Stan
- Enables agile (incremental) workflow for developing probabilistic models
    - language
    - automated inference
    - diagnostics
  - Many frameworks Stan, PyMC3, Pyro (Uber), Edward (Google), Birch (Uppsala), ...



UPPSALA  
UNIVERSITET

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Section 5

Stan



UPPSALA  
UNIVERSITET

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Stan - probabilistic programming framework

---

- Language, inference engine, user interfaces, documentation, case studies, diagnostics, packages, ...
  - autodiff to compute gradients of the log density



[mc-stan.org](https://mc-stan.org)



UPPSALA  
UNIVERSITET

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Stan - probabilistic programming framework

---

- Language, inference engine, user interfaces, documentation, case studies, diagnostics, packages, ...
  - autodiff to compute gradients of the log density
- More than ten thousand users in social, biological, and physical sciences, medicine, engineering, and business



[mc-stan.org](https://mc-stan.org)



UPPSALA  
UNIVERSITET

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Stan - probabilistic programming framework

---

- Language, inference engine, user interfaces, documentation, case studies, diagnostics, packages, ...
  - autodiff to compute gradients of the log density
- More than ten thousand users in social, biological, and physical sciences, medicine, engineering, and business
- Several full time developers, 40+ developers, more than 100 contributors



[mc-stan.org](https://mc-stan.org)



UPPSALA  
UNIVERSITET

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Stan - probabilistic programming framework

---

- Language, inference engine, user interfaces, documentation, case studies, diagnostics, packages, ...
  - autodiff to compute gradients of the log density
- More than ten thousand users in social, biological, and physical sciences, medicine, engineering, and business
- Several full time developers, 40+ developers, more than 100 contributors
- R, Python, Julia, Scala, Stata, Matlab, command line interfaces
- More than 100 R packages using Stan



[mc-stan.org](https://mc-stan.org)



UPPSALA  
UNIVERSITET

# Stan

---

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- Stanislaw Ulam (1909-1984)
  - Monte Carlo method
  - H-Bomb



UPPSALA  
UNIVERSITET

# Adaptive dynamic HMC in Stan

---

- Dynamic HMC using growing tree to increase simulation trajectory until no-U-turn criterion stopping
  - max treedepth to keep computation in control

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan





UPPSALA  
UNIVERSITET

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Adaptive dynamic HMC in Stan

---

- Dynamic HMC using growing tree to increase simulation trajectory until no-U-turn criterion stopping
  - max treedepth to keep computation in control
  - pick a draw along the trajectory with probabilities adjusted to take into account the error in the discretized dynamic simulation



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Adaptive dynamic HMC in Stan

---

- Dynamic HMC using growing tree to increase simulation trajectory until no-U-turn criterion stopping
  - max treedepth to keep computation in control
  - pick a draw along the trajectory with probabilities adjusted to take into account the error in the discretized dynamic simulation
  - give bigger weight for tree parts further away to increase probability of jumping further away



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Adaptive dynamic HMC in Stan

---

- Dynamic HMC using growing tree to increase simulation trajectory until no-U-turn criterion stopping
  - max treedepth to keep computation in control
  - pick a draw along the trajectory with probabilities adjusted to take into account the error in the discretized dynamic simulation
  - give bigger weight for tree parts further away to increase probability of jumping further away
- Mass matrix and step size adaptation in Stan
  - mass matrix refers to having different scaling for different parameters and optionally also rotation to reduce correlations



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Adaptive dynamic HMC in Stan

---

- Dynamic HMC using growing tree to increase simulation trajectory until no-U-turn criterion stopping
  - max treedepth to keep computation in control
  - pick a draw along the trajectory with probabilities adjusted to take into account the error in the discretized dynamic simulation
  - give bigger weight for tree parts further away to increase probability of jumping further away
- Mass matrix and step size adaptation in Stan
  - mass matrix refers to having different scaling for different parameters and optionally also rotation to reduce correlations
  - mass matrix and step size adjustment and are estimated during initial adaptation phase



UPPSALA  
UNIVERSITET

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Adaptive dynamic HMC in Stan

---

- Dynamic HMC using growing tree to increase simulation trajectory until no-U-turn criterion stopping
  - max treedepth to keep computation in control
  - pick a draw along the trajectory with probabilities adjusted to take into account the error in the discretized dynamic simulation
  - give bigger weight for tree parts further away to increase probability of jumping further away
- Mass matrix and step size adaptation in Stan
  - mass matrix refers to having different scaling for different parameters and optionally also rotation to reduce correlations
  - mass matrix and step size adjustment and are estimated during initial adaptation phase
  - step size is adjusted to be as big as possible while keeping discretization error in control (adapt\_delta)



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Adaptive dynamic HMC in Stan

- Dynamic HMC using growing tree to increase simulation trajectory until no-U-turn criterion stopping
  - max treedepth to keep computation in control
  - pick a draw along the trajectory with probabilities adjusted to take into account the error in the discretized dynamic simulation
  - give bigger weight for tree parts further away to increase probability of jumping further away
- Mass matrix and step size adaptation in Stan
  - mass matrix refers to having different scaling for different parameters and optionally also rotation to reduce correlations
  - mass matrix and step size adjustment and are estimated during initial adaptation phase
  - step size is adjusted to be as big as possible while keeping discretization error in control (`adapt_delta`)
- After adaptation the algorithm parameters are fixed



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Adaptive dynamic HMC in Stan

- Dynamic HMC using growing tree to increase simulation trajectory until no-U-turn criterion stopping
  - max treedepth to keep computation in control
  - pick a draw along the trajectory with probabilities adjusted to take into account the error in the discretized dynamic simulation
  - give bigger weight for tree parts further away to increase probability of jumping further away
- Mass matrix and step size adaptation in Stan
  - mass matrix refers to having different scaling for different parameters and optionally also rotation to reduce correlations
  - mass matrix and step size adjustment and are estimated during initial adaptation phase
  - step size is adjusted to be as big as possible while keeping discretization error in control (`adapt_delta`)
- After adaptation the algorithm parameters are fixed
- After warmup store iterations for inference



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Adaptive dynamic HMC in Stan

---

- Dynamic HMC using growing tree to increase simulation trajectory until no-U-turn criterion stopping
  - max treedepth to keep computation in control
  - pick a draw along the trajectory with probabilities adjusted to take into account the error in the discretized dynamic simulation
  - give bigger weight for tree parts further away to increase probability of jumping further away
- Mass matrix and step size adaptation in Stan
  - mass matrix refers to having different scaling for different parameters and optionally also rotation to reduce correlations
  - mass matrix and step size adjustment and are estimated during initial adaptation phase
  - step size is adjusted to be as big as possible while keeping discretization error in control (`adapt_delta`)
- After adaptation the algorithm parameters are fixed
- After warmup store iterations for inference
- See more details in Stan reference manual





- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Binomial model - Stan code

---

```
data {  
  int<lower=0> N;          // number of experiments  
  int<lower=0,upper=N> y; // number of successes  
}  
  
parameters {  
  real<lower=0,upper=1> theta; // parameter of the  
}  
  
model {  
  theta ~ beta(1,1); //prior  
  y ~ binomial(N,theta); // observation model  
}
```



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Binomial model - Stan code

---

```
data {  
  int<lower=0> N;          // number of experiments  
  int<lower=0,upper=N> y;  // number of successes  
}  
  
parameters {  
  real<lower=0,upper=1> theta; // parameter of the  
}  
  
model {  
  theta ~ beta(1,1);        //prior  
  y ~ binomial(N,theta);    // observation model  
}
```



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Binomial model - Stan code

---

```
data {  
  int<lower=0> N;          // number of experiments  
  int<lower=0,upper=N> y;  // number of successes  
}  
  
parameters {  
  real<lower=0,upper=1> theta; // parameter of the  
}  
  
model {  
  theta ~ beta(1,1);        //prior  
  y ~ binomial(N,theta);    // observation model  
}
```



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Binomial model - Stan code

---

```
data {  
  int<lower=0> N;           // number of experiments  
  int<lower=0,upper=N> y;   // number of successes  
}
```

- Data type and size are declared
- Stan checks that given data matches type and constraints



```
data {  
  int<lower=0> N;      // number of experiments  
  int<lower=0,upper=N> y; // number of successes  
}
```

- MCMC recap
  - Hamiltonian Monte Carlo
    - Dynamic HMC and NUTS
  - HMC diagnostics
  - Probabilistic Programming
  - Stan
- Data type and size are declared
  - Stan checks that given data matches type and constraints
    - If you are not used to strong typing, this may feel annoying, but it will reduce the probability of coding errors, which will reduce probability of data analysis errors



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- **Stan**

## Binomial model - Stan code

---

```
parameters {  
  real<lower=0,upper=1> theta;  
}
```

- Parameters may have constraints
- Stan makes transformation to unconstrained space and samples in unconstrained space
  - e.g. log transformation for `<lower=a>`
  - e.g. logit transformation for `<lower=a,upper=b>`



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Binomial model - Stan code

```
parameters {  
  real<lower=0,upper=1> theta;  
}
```

- Parameters may have constraints
- Stan makes transformation to unconstrained space and samples in unconstrained space
  - e.g. log transformation for `<lower=a>`
  - e.g. logit transformation for `<lower=a,upper=b>`
- For these declared transformation Stan automatically takes into account the Jacobian of the transformation (see BDA3 p. 21)



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

```
model {  
  theta ~ beta(1,1);    // prior  
  y ~ binomial(N,theta); // likelihood  
}
```





- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Binomial model - Stan code

```
model {  
  theta ~ beta(1,1);      // prior  
  y ~ binomial(N,theta);  // likelihood  
}
```

- $\sim$  is syntactic sugar and this is equivalent to

```
model {  
  target += beta_lpdf(theta | 1, 1);  
  target += binomial_lpmf(y | N, theta);  
}
```



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Binomial model - Stan code

```
model {  
  theta ~ beta(1,1);      // prior  
  y ~ binomial(N,theta);  // likelihood  
}
```

- $\sim$  is syntactic sugar and this is equivalent to

```
model {  
  target += beta_lpdf(theta | 1, 1);  
  target += binomial_lpmf(y | N, theta);  
}
```

- target is the log posterior density



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Binomial model - Stan code

```
model {  
  theta ~ beta(1,1);      // prior  
  y ~ binomial(N,theta);  // likelihood  
}
```

- $\sim$  is syntactic sugar and this is equivalent to

```
model {  
  target += beta_lpdf(theta | 1, 1);  
  target += binomial_lpmf(y | N, theta);  
}
```

- target is the log posterior density
- `_lpdf` for continuous, `_lpmf` for discrete distributions (discrete for the left hand side of `|`)



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Binomial model - Stan code

```
model {  
  theta ~ beta(1,1);      // prior  
  y ~ binomial(N,theta);  // likelihood  
}
```

- $\sim$  is syntactic sugar and this is equivalent to

```
model {  
  target += beta_lpdf(theta | 1, 1);  
  target += binomial_lpmf(y | N, theta);  
}
```

- target is the log posterior density
- `_lpdf` for continuous, `_lpmf` for discrete distributions (discrete for the left hand side of `|`)
- for Stan sampler there is no difference between prior and likelihood, all that matters is the final target



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Binomial model - Stan code

```
model {  
  theta ~ beta(1,1);      // prior  
  y ~ binomial(N,theta);  // likelihood  
}
```

- $\sim$  is syntactic sugar and this is equivalent to

```
model {  
  target += beta_lpdf(theta | 1, 1);  
  target += binomial_lpmf(y | N, theta);  
}
```

- target is the log posterior density
- `_lpdf` for continuous, `_lpmf` for discrete distributions (discrete for the left hand side of `|`)
- for Stan sampler there is no difference between prior and likelihood, all that matters is the final target
- you can write in Stan language any program to compute the log density (Stan language is Turing complete)



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- Stan compiles (transpiles) the model written in Stan language to C++
  - this makes the sampling for complex models and bigger data faster
  - also makes Stan models easily portable, you can use your own favorite interface



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# RStan

---

## RStan

```
library(rstan)
rstan_**options**(**auto_write** = TRUE)
options(mc.cores = parallel::detectCores())

d_bin <- list(N = 10, y = 7)
fit_bin <- stan(file = 'binom.stan', data = d_bin)
```



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# RStan

---

## RStan

```
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())

d_bin <- list(N = 10, y = 7)
fit_bin <- stan(file = 'binom.stan', data = d_bin)
```





UPPSALA  
UNIVERSITET

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- **Stan**

# PyStan

---

## PyStan

```
import pystan
import stan_utility
```

```
data = dict(N=10, y=8)
model = stan_utility.compile_model('binom.stan')
fit = model.sampling(data=data)
```



UPPSALA  
UNIVERSITET

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- **Stan**

# PyStan

---

## PyStan

```
import pystan
import stan_utility
```

```
data = dict(N=10, y=8)
model = stan_utility.compile_model('binom.stan')
fit = model.sampling(data=data)
```



UPPSALA  
UNIVERSITET

# Stan

---

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- Compilation (unless previously compiled model available)
- Warm-up including adaptation
- Sampling
- Generated quantities
- Save posterior draws
- Report divergences,  $n_{E_{\max}}$ ,  $\hat{R}$



# Difference between proportions

---

- MCMC recap
  - Hamiltonian Monte Carlo
    - Dynamic HMC and NUTS
  - HMC diagnostics
  - Probabilistic Programming
  - Stan
- An experiment was performed to estimate the effect of beta-blockers on mortality of cardiac patients
  - A group of patients were randomly assigned to treatment and control groups:
    - out of 674 patients receiving the control, 39 died
    - out of 680 receiving the treatment, 22 died



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- **Stan**

## Difference between proportions

```
data {  
  int<lower=0> N1;  
  int<lower=0> y1;  
  int<lower=0> N2;  
  int<lower=0> y2;  
}  
parameters {  
  real<lower=0, upper=1> theta1;  
  real<lower=0, upper=1> theta2;  
}  
model {  
  theta1 ~ beta(1,1);  
  theta2 ~ beta(1,1);  
  y1 ~ binomial(N1, theta1);  
  y2 ~ binomial(N2, theta2);  
}  
  
generated quantities {  
  real oddsratio;  
  oddsratio = (theta2/(1-theta2))/(theta1/(1-theta1));  
}
```



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- **Stan**

## Difference between proportions

```
data {  
  int<lower=0> N1;  
  int<lower=0> y1;  
  int<lower=0> N2;  
  int<lower=0> y2;  
}  
parameters {  
  real<lower=0, upper=1> theta1;  
  real<lower=0, upper=1> theta2;  
}  
model {  
  theta1 ~ beta(1,1);  
  theta2 ~ beta(1,1);  
  y1 ~ binomial(N1, theta1);  
  y2 ~ binomial(N2, theta2);  
}  
  
generated quantities {  
  real oddsratio;  
  oddsratio = (theta2/(1-theta2))/(theta1/(1-theta1));  
}
```



# Difference between proportions

---

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

```
generated quantities {  
  real oddsratio;  
  oddsratio = (theta2/(1-theta2))/(theta1/(1-theta1))  
}
```

- generated quantities is run after the sampling



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Difference between proportions

```
d_bin2 <- list(N1 = 674, y1 = 39, N2 = 680, y2 = 20)
fit_bin2 <- stan(file = 'binom2.stan', data = d_bin2)
```

```
starting worker pid=10151 on localhost:11783 at 10:03:27.872
starting worker pid=10164 on localhost:11783 at 10:03:28.087
starting worker pid=10176 on localhost:11783 at 10:03:28.295
starting worker pid=10185 on localhost:11783 at 10:03:28.461
```

SAMPLING FOR MODEL 'binom2' NOW (CHAIN 1).

Gradient evaluation took 6e-06 seconds  
1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.  
Adjust your expectations accordingly!

```
Iteration:   1 / 2000 [  0%] (Warmup)
Iteration:  200 / 2000 [ 10%] (Warmup)
...
Iteration: 1000 / 2000 [ 50%] (Warmup)
Iteration: 1001 / 2000 [ 50%] (Sampling)
...
Iteration: 2000 / 2000 [100%] (Sampling)
```

```
Elapsed Time: 0.012908 seconds (Warm-up)
              0.017027 seconds (Sampling)
              0.029935 seconds (Total)
```

SAMPLING FOR MODEL 'binom2' NOW (CHAIN 2).

...





- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Difference between proportions

```
monitor(fit_bin2, probs = c(0.1, 0.5, 0.9))
```

Inference for the input samples  
(4 chains: each with iter=1000; warmup=0):

	mean	se_mean	sd	10%	50%	90%	n_eff	Rhat
theta1	0.1	0	0.0	0.0	0.1	0.1	3280	1
theta2	0.0	0	0.0	0.0	0.0	0.0	3171	1
oddsratio	0.6	0	0.2	0.4	0.6	0.8	3108	1
lp__	-253.5	0	1.0	-254.8	-253.2	-252.6	1922	1

For each parameter, `n_eff` is a crude measure of effective sample size and `Rhat` is the potential scale reduction factor on split chain convergence, `Rhat=1`).



## Difference between proportions

```
monitor(fit_bin2, probs = c(0.1, 0.5, 0.9))
```

Inference for the input samples  
(4 chains: each with iter=1000; warmup=0):

	mean	se_mean	sd	10%	50%	90%	n_eff	Rhat
theta1	0.1	0	0.0	0.0	0.1	0.1	3280	1
theta2	0.0	0	0.0	0.0	0.0	0.0	3171	1
oddsratio	0.6	0	0.2	0.4	0.6	0.8	3108	1
lp__	-253.5	0	1.0	-254.8	-253.2	-252.6	1922	1

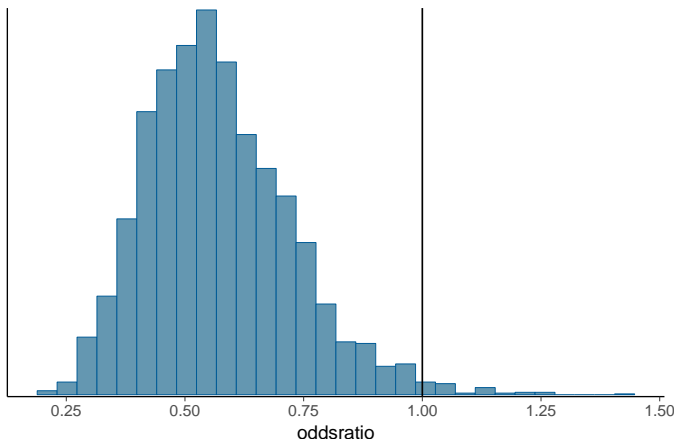
For each parameter, `n_eff` is a crude measure of effective sample size and `Rhat` is the potential scale reduction factor on split chain convergence, `Rhat=1`).

- `lp__` is the log density, ie, same as target



## Difference between proportions

```
draws <- as.data.frame(fit_bin2)
mcmc_hist(draws, pars = 'oddsratio') +
  geom_vline(xintercept = 1) +
  scale_x_continuous(breaks = c(seq(0.25, 1.5, by=0.
```



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

```
check_treedepth(fit_bin2)  
check_div(fit_bin2)
```

```
[1] "0 of 4000 iterations saturated the maximum tree depth of 1  
[1] "0 of 4000 iterations ended with a divergence (0%)"
```

```
get_num_leapfrog_per_iteration(fit_bin2)
```



UPPSALA  
UNIVERSITET

# Shinystan

---

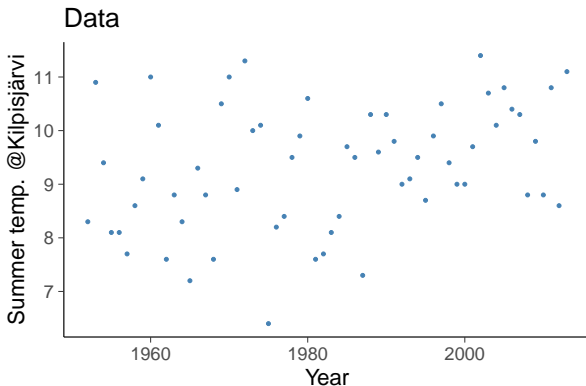
- MCMC recap
  - Hamiltonian Monte Carlo
    - Dynamic HMC and NUTS
  - HMC diagnostics
  - Probabilistic Programming
  - Stan
- Graphical user interface for analysing MCMC results



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Kilpisjärvi summer temperature

- Temperature at Kilpisjärvi in June, July and August from 1952 to 2013
- Is there change in the temperature?





- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Gaussian linear model

```
data {  
  int<lower=0> N; // number of data points  
  vector[N] x; //  
  vector[N] y; //  
}  
parameters {  
  real alpha;  
  real beta;  
  real<lower=0> sigma;  
}  
transformed parameters {  
  vector[N] mu;  
  mu <- alpha + beta*x;  
}  
model {  
  y ~ normal(mu, sigma);  
}
```



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

```
data {  
  int<lower=0> N; // number of data points  
  vector[N] x; //  
  vector[N] y; //  
}
```

- difference between `vector[N] x` and `real x[N]`





- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Gaussian linear model

---

```
parameters {  
  real alpha;  
  real beta;  
  real<lower=0> sigma;  
}  
transformed parameters {  
  vector[N] mu;  
  mu <- alpha + beta*x;  
}
```

- transformed parameters are deterministic transformations of parameters and data



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- **Stan**

## Priors for Gaussian linear model

```
data {  
  int <lower=0> N; // number of data points  
  vector[N] x; //  
  vector[N] y; //  
  real pmualpha; // prior mean for alpha  
  real psalpha; // prior std for alpha  
  real pmubeta; // prior mean for beta  
  real psbeta; // prior std for beta  
}  
...  
transformed parameters {  
  vector[N] mu;  
  mu <- alpha + beta*x;  
}  
model {  
  alpha ~ normal(pmualpha, psalpha);  
  beta ~ normal(pmubeta, psbeta);  
  y ~ normal(mu, sigma);  
}
```



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Student-t linear model

```
...
parameters {
  real alpha;
  real beta;
  real<lower=0> sigma;
  real<lower=1,upper=80> nu;
}
transformed parameters {
  vector[N] mu;
  mu <- alpha + beta*x;
}
model {
  nu ~ gamma(2,0.1);
  y ~ student_t(nu, mu, sigma);
}
```



UPPSALA  
UNIVERSITET

# Priors

---

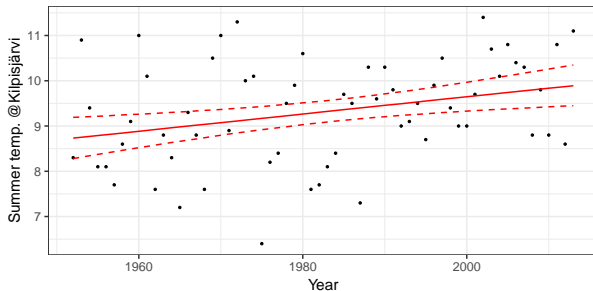
- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- Prior for temperature increase?



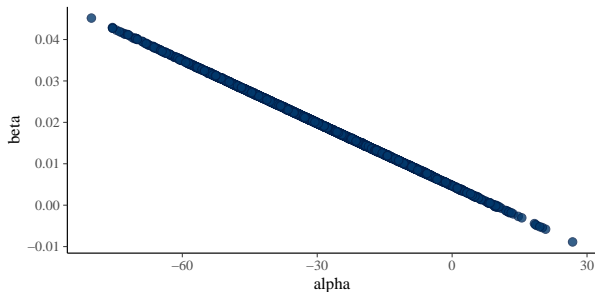
## Posterior fit

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan





## Posterior draws of alpha and beta

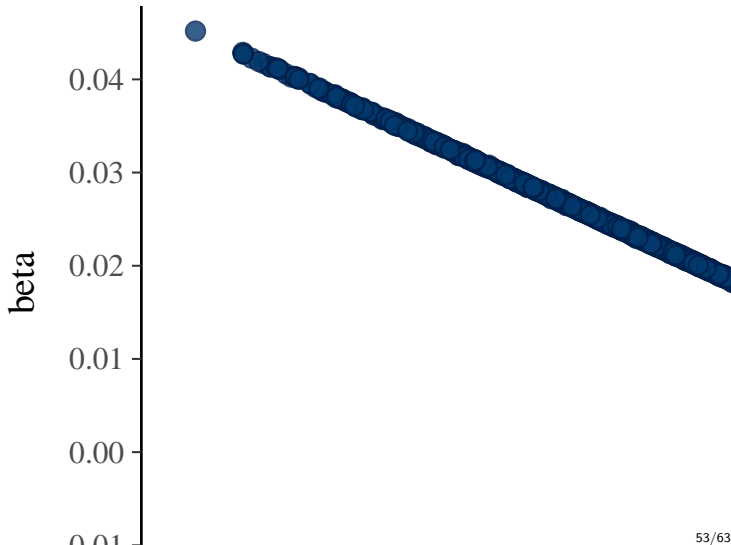


- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan



# Kilpisjärvi summer temperature

Posterior draws of alpha and beta



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Linear regression model in Stan

```
data {  
  int<lower=0> N; // number of data points  
  vector[N] x; //  
  vector[N] y; //  
  real xpred; // input location for prediction  
}  
  
transformed data {  
  vector[N] x_std;  
  vector[N] y_std;  
  real xpred_std;  
  x_std = (x - mean(x)) / sd(x);  
  y_std = (y - mean(y)) / sd(y);  
  xpred_std = (xpred - mean(x)) / sd(x);  
}
```





- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- RStanARM provides simplified model description with pre-compiled models
  - no need to wait for compilation
  - a restricted set of models

Two group Binomial model:

```
d_bin2 <- data.frame(N = c(674, 680), y = c(39, 22), grp2 = c(0, 1))
fit_bin2 <- stan_glm(y/N ~ grp2, family = binomial(), data = d_bin2, weights = N)
```



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- **Stan**

- RStanARM provides simplified model description with pre-compiled models
  - no need to wait for compilation
  - a restricted set of models

## Two group Binomial model:

```
d_bin2 <- data.frame(N = c(674, 680), y = c(39,22), grp2 = c(0,1))
fit_bin2 <- stan_glm(y/N ~ grp2, family = binomial(), data = d_bin2, weights = N)
```

## Gaussian linear model

```
fit_lin <- stan_glm(temp ~ year, data = d_lin)
```



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- BRMS provides simplified model description
  - a larger set of models than RStanARM, but still restricted
  - need to wait for the compilation

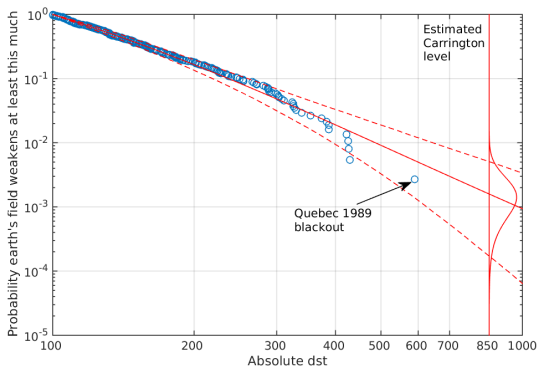
```
fit_bin2 <- brm(y/N ~ grp2, family = binomial(), data = d_bin2,  
               weights = N)
```

```
fit_lin_t <- brm(temp ~ year, data = d_lin, family = student())
```



## Geomagnetic storms

- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan





- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Extreme value analysis

```
data {  
  int<lower=0> N;  
  vector<lower=0>[N] y;  
  int<lower=0> Nt;  
  vector<lower=0>[Nt] yt;  
}  
transformed data {  
  real ymax;  
  ymax <- max(y);  
}  
parameters {  
  real<lower=0> sigma;  
  real<lower=-sigma/ymax> k;  
}  
model {  
  y ~ gpareto(k, sigma);  
}  
generated quantities {  
  vector[Nt] predccdf;  
  predccdf<-gpareto_ccdf(yt, k, sigma);  
}
```



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

# Functions

```
functions {  
  real gpareto_lpdf(vector y, real k, real sigma) {  
    // generalised Pareto log pdf with  $\mu=0$   
    // should check and give error if  $k < 0$   
    // and  $\max(y)/\sigma > -1/k$   
    int N;  
    N <- dims(y)[1];  
    if (fabs(k) > 1e-15)  
      return  $-(1+1/k) * \text{sum}(\log 1\text{pv}(y*k/\sigma)) - N * \log(\sigma)$   
    else  
      return  $-\text{sum}(y/\sigma) - N * \log(\sigma)$ ; // limit  $k \rightarrow 0$   
  }  
  vector gpareto_ccdf(vector y, real k, real sigma) {  
    // generalised Pareto log ccdf with  $\mu=0$   
    // should check and give error if  $k < 0$   
    // and  $\max(y)/\sigma < -1/k$   
    if (fabs(k) > 1e-15)  
      return  $\exp((-1/k) * \log 1\text{pv}(y/\sigma * k))$ ;  
    else  
      return  $\exp(-y/\sigma)$ ; // limit  $k \rightarrow 0$   
  }  
}
```



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

- R
  - shinystan — interactive diagnostics
  - bayesplot — visualization and model checking (see model checking in Ch 6)
  - loo — cross-validation model assessment, comparison and averaging (see Ch 7)
  - projpred — projection predictive variable selection
- Python
  - ArviZ — visualization, and model checking and assessment (see Ch 6 and 7)



- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Different interfaces

---

- RStan / PyStan
  - C++ functions of Stan are called directly from R / Python
  - Higher integration between R/Python and Stan, but maybe more difficult to install due to more requirements of compatible C++ compilers and libraries
- CmdStanR / CmdStanPy
  - Lightweight interface on top of commandline program CmdStan
  - Lacks some features that are not needed in this course, but is usually easier to install
- More recent useful R packages
  - posterior: for handling posterior draws, convergence diagnostics, and summaries
  - tidybayes + ggdist: pretty plots





- MCMC recap
- Hamiltonian Monte Carlo
  - Dynamic HMC and NUTS
- HMC diagnostics
- Probabilistic Programming
- Stan

## Extra material for Stan

- Andrew Gelman, Daniel Lee, and Jiqiang Guo (2015)  
Stan: A probabilistic programming language for Bayesian inference and optimization.  
[http://www.stat.columbia.edu/~gelman/research/published/stan\\_jebbs\\_2.pdf](http://www.stat.columbia.edu/~gelman/research/published/stan_jebbs_2.pdf)
- Carpenter et al (2017). Stan: A probabilistic programming language. Journal of Statistical Software 76(1).  
<https://doi.org/10.18637/jss.v076.i01>
- Stan User's Guide, Language Reference Manual, and Language Function Reference (in html and pdf)  
<https://mc-stan.org/users/documentation/>
  - easiest to start from Example Models in User's guide
- Basics of Bayesian inference and Stan, part 1 Jonah Gabry & Lauren Kennedy (StanCon 2019 Helsinki tutorial)
  - <https://www.youtube.com/watch?v=ZRpo41102KQ&index=6&list=PLuwyh42iHquU4hUBQs20hkBsKSMrp6H0J>
  - <https://www.youtube.com/watch?v=6cc4N1vT8pk&index=7&list=PLuwyh42iHquU4hUBQs20hkBsKSMrp6H0J>