COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

STAT 4224/5224

Bayesian Statistics

Dobrin Marchev

Recall: Metropolis-Hastings Algorithm

- Very general method for constructing a Markov chain with a specific invariant density $f(x)$, which is the target for the simulation.
- Suppose that $q(y|x)$ is any conditional density, called proposal density.
- Algorithm:
  - Step 0: Start at any $X = x_0$
  - Step 1: Generate $Y_i \sim q(y|x_{i\text{-}1})$
  - Step 2: Set $X_i = \begin{cases} Y_i \text{ with probability } \alpha(x_{i-1}, y_i) \\ \quad x_{i-1}, \quad \text{o/w} \end{cases}$

  where $\alpha(x, y) = \min\left\{\frac{f(y)}{f(x)} \frac{q(x|y)}{q(y|x)}, 1\right\}$ is called the acceptance probability

  - Go to Step 1 and repeat

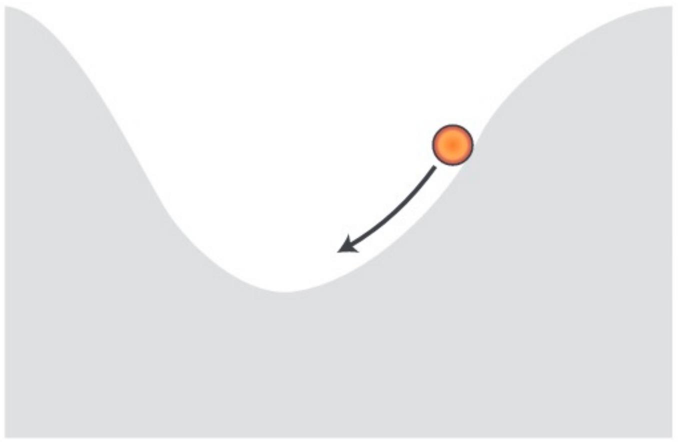
  Notes:

  - When $q(y|x)$ is symmetric, the algorithm is known as Metropolis.
  - When $q(y|x)$ does not depend on x, the algorithm is known as Independent Metropolis-Hastings.
  - When $q(y|x) = g(y - x)$, the algorithm is known as Random Walk Metropolis-Hastings (see next slide)

Hamiltonian Monte Carlo

- Hamiltonian Monte Carlo improves the efficiency of MH by employing a guided proposal generation scheme.
- More specifically, HMC uses the gradient of the log posterior to direct the Markov chain towards regions of higher posterior density.
- As a result, a well-tuned HMC chain will accept proposals at a much higher rate than the traditional MH algorithm.
- While both MH and HMC produce ergodic Markov chains, the mathematics of HMC is substantially more complex than that of MH.

Main Idea



- Suppose $f(\theta)$ is a one-dimensional posterior density function, and $-\log[f(\theta)]$ assumes the shape of an inverse bell-shaped curve.
- To generate θ in a region of high posterior density, one needs to sample θ in the region corresponding to the lower values of $-\log f(\theta)$; the region can be reached with the guidance of the gradient of $-\log f(\theta)$.

Hamiltonian Dynamics

- In a sense, the approach is analogous to the movement of a hypothetical object on a frictionless curve, where the object traverses and lingers at the bottom of the valley while occasionally visiting the higher grounds on both sides.
- In classical mechanics, such movements are described by the Hamiltonian equations, where the exchanges of kinetic and potential energy dictate the object's location at any given moment.

The Hamiltonian Equations

In a Hamiltonian system, the horizontal and vertical positions are given by (θ, p). In MCMC, we are interested in θ ~ *f*(θ). The parameter *p*, which is often referred to as the momentum, is an auxiliary quantity that we use to simulate θ under the Hamiltonian equations. The momentum matches the dimensionality of **θ** as a vector of length *k*.

We write the Hamiltonian function as *H*(**θ**, **p**), which consists of *potential* energy *U*(**θ**) and *kinetic* energy *K*(**p**):

$$H(\boldsymbol{\theta}, \boldsymbol{p}) = U(\boldsymbol{\theta}) + K(\boldsymbol{p}), \boldsymbol{p}, \boldsymbol{\theta} \in \mathbb{R}^k$$

In statistical applications of MCMC, we are primarily interested in generating **θ** from a given distribution *f* (**θ**). To do so, we let *U* (**θ**) = − log *f* (**θ**). For momentum, we typically assume **p** ~ $N_k(0, \boldsymbol{M})$, where ***M*** is a user-specified covariance matrix.

Hamiltonian Equations

Under the above formulation, we have that:

$$H(\boldsymbol{\theta}, \boldsymbol{p}) = -\log f(\theta) + \frac{1}{2} p' M p$$

Over time, HMC travels on trajectories that are governed by the following first-order differential equations, known as the Hamiltonian equations:

$$\frac{\partial p}{\partial t} = -\frac{\partial H(\boldsymbol{\theta}, \boldsymbol{p})}{\partial \theta} = -\frac{\partial U(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \nabla_{\boldsymbol{\theta}} \log f(\boldsymbol{\theta})$$

$$\frac{\partial \boldsymbol{\theta}}{\partial t} = -\frac{\partial H(\boldsymbol{\theta}, \boldsymbol{p})}{\partial p} = -\frac{\partial K(\boldsymbol{p})}{\partial \boldsymbol{p}} = \boldsymbol{M^{-1} p}$$

where $\nabla_{\boldsymbol{\theta}} \log f(\boldsymbol{\theta})$ is the gradient of the log posterior. A solution to the Hamiltonian equations is a function that defines the path of (θ, p) from which specific values of θ could be sampled.

Solving the Hamiltonian Differential Equations

- Solving the Hamiltonian equations is a critical step in HMC simulation.
- A standard approach for solving differential equations is Euler's method, which produces a discrete function that approximates the solution at each time t.
- Errors tend to accumulate in Euler's method, especially after a larger number of steps.
- In HMC, one often has to take a larger number of steps to ensure the new proposal is sufficiently far from the location of the previous sample.

Leapfrog method

The leapfrog method is a good alternative to the standard Euler's method for approximating the solutions to Hamiltonian equations (Ruth 1983).

The leapfrog algorithm modifies Euler's method by using a discrete step size ϵ individually for p and θ, with a full step ϵ in θ sandwiched between two half-steps ϵ /2 for p:

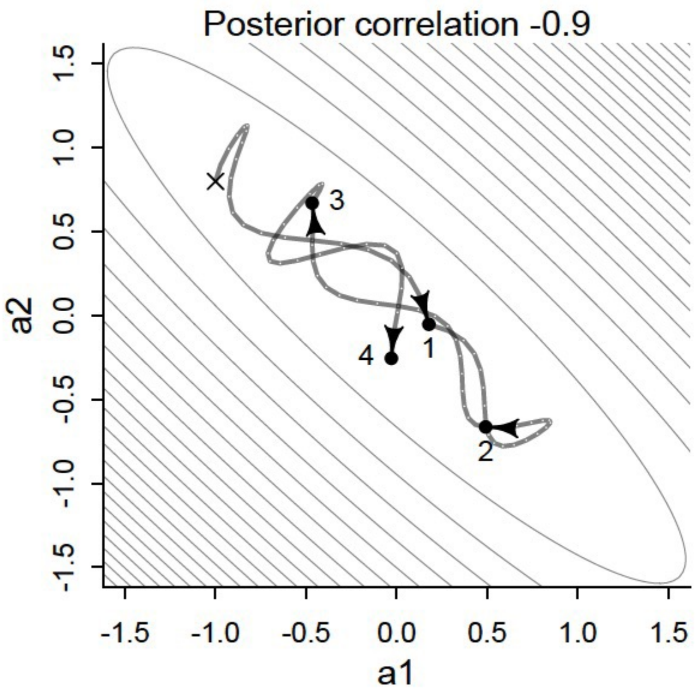
$$p\left(t+\frac{\epsilon}{2}\right) = p(t) + \left(\frac{\epsilon}{2}\right)\nabla_{\boldsymbol{\theta}} \log f(\boldsymbol{\theta}(t))$$

$$\theta(t+\epsilon) = \theta(t) + \epsilon M^{-1} p\left(t+\frac{\epsilon}{2}\right)$$

$$p(t+\epsilon) = p\left(t+\frac{\epsilon}{2}\right) + \left(\frac{\epsilon}{2}\right)\nabla_{\boldsymbol{\theta}} \log f(\boldsymbol{\theta}(t+\epsilon))$$

Hamiltonian Monte Carlo

- Why does HMC work much better?
- Doesn't get stuck — follows gradient
- Extra variables (momentum, energy) provide diagnostics
- But also requires more
  - Gradients — curvature of log-posterior
  - "Mass" of particle
  - Number of leaps in a single trajectory
  - Size of individual leaps
- These need to be tuned right
- Gradients are unique to each model



The U-Turn Problem

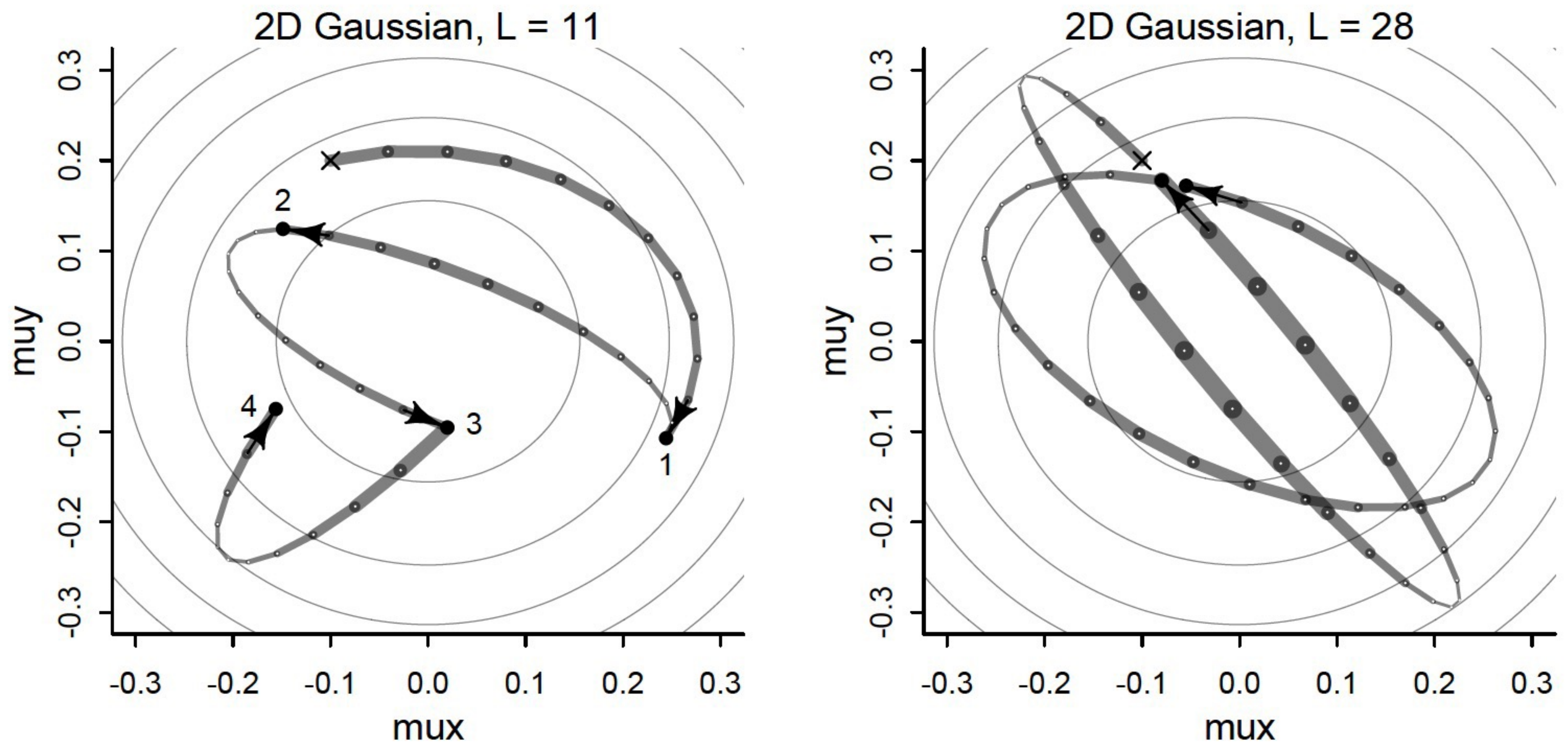
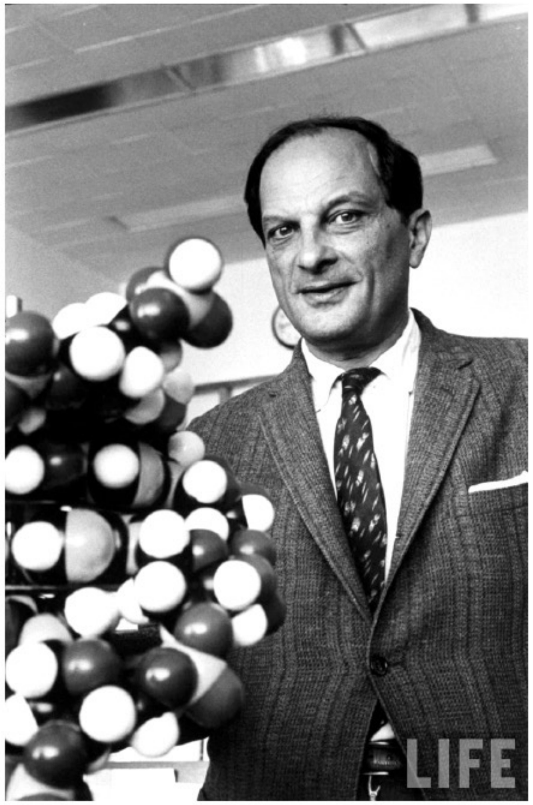
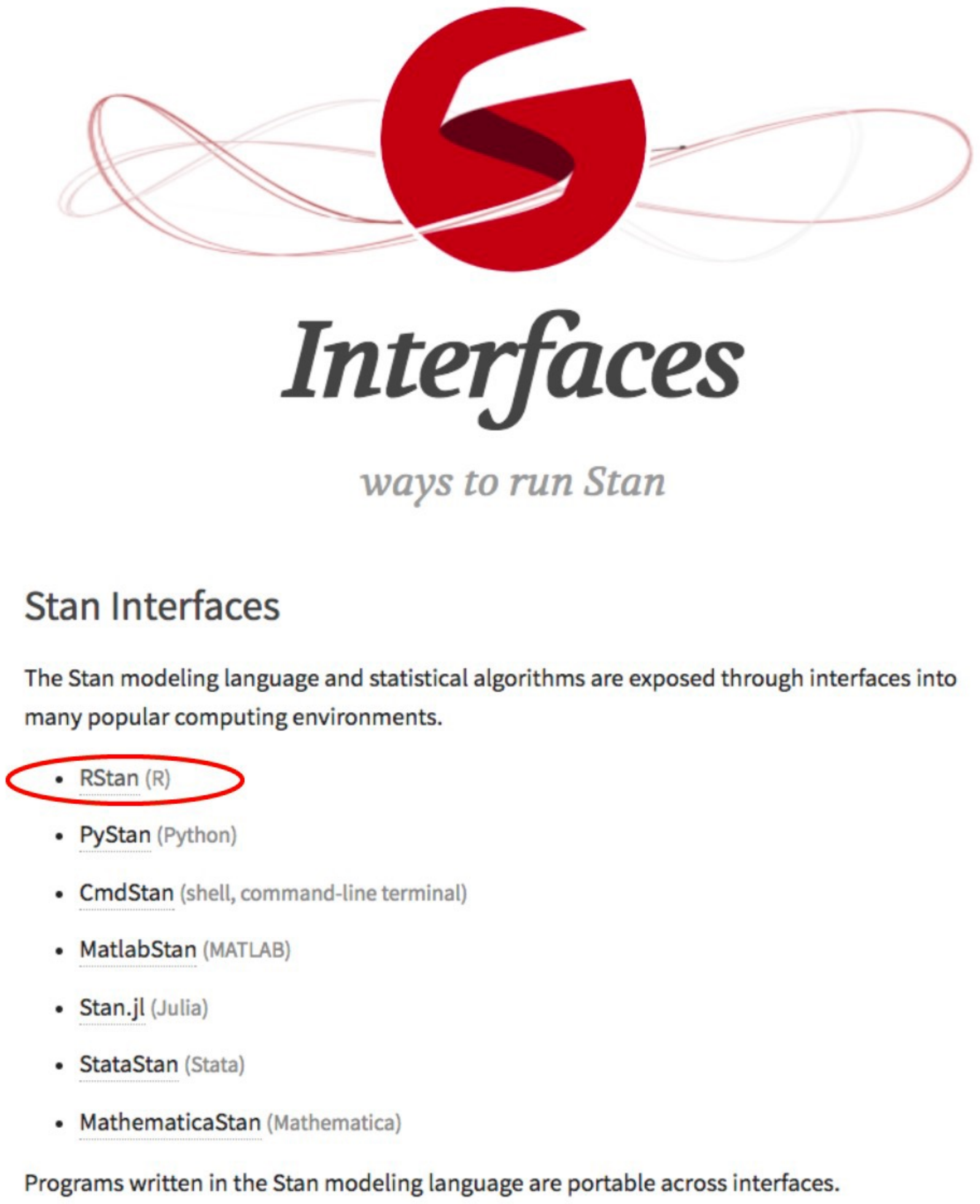


Figure 9.6

- mc-stan.org
- Install RStan
  1. Get C++ compiler
  2. ???
  3. Profit



Stanislaw Ulam (1909–1984)



Interfaces

ways to run Stan

Stan Interfaces

The Stan modeling language and statistical algorithms are exposed through interfaces into many popular computing environments.

- [✓ RStan (R)]
- PyStan (Python)
- CmdStan (shell, command-line terminal)
- MatlabStan (MATLAB)
- Stan.jl (Julia)
- StataStan (Stata)
- MathematicaStan (Mathematica)

Programs written in the Stan modeling language are portable across interfaces.

v Ulam and his daughter Claire with MANIAC

Check out the movie: "Adventures of a Mathematician"

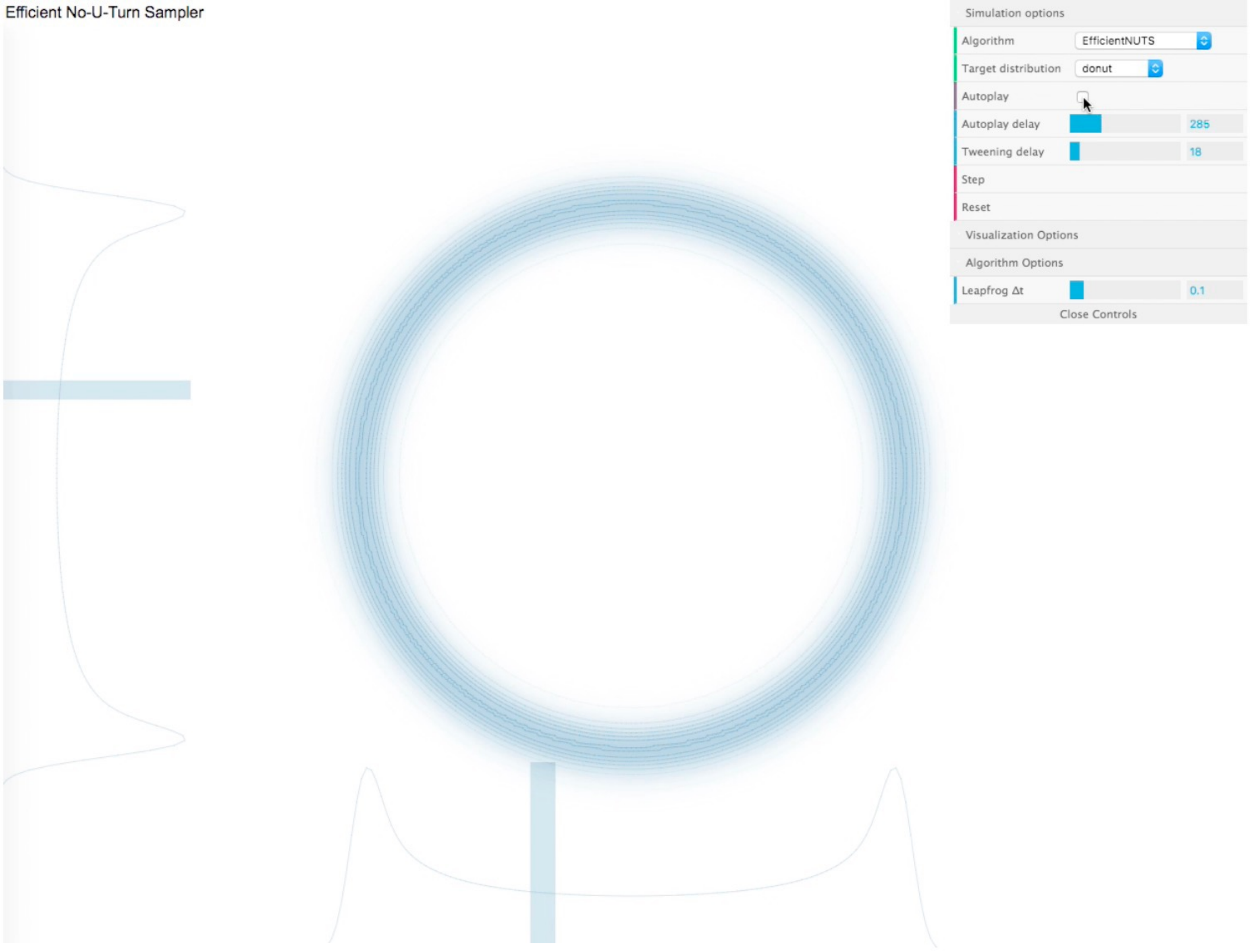
Stan is NUTS

No U-Turn Sampler

Automatic Step Size and Number Adaptation

- No U-Turn Sampler (NUTS2): Adaptive Hamiltonian Monte Carlo
- Implemented in Stan (rstan: mc-stan.org)

$$\begin{tikzcd}
\boxed{\text{Formula}} \arrow[r] & \boxed{\text{Stan model}} \arrow[r] & \boxed{\text{C++ model}} \arrow[r] & \boxed{\text{Reusable Sampler}}
\end{tikzcd}$$


Efficient No-U-Turn Sampler

Simulation options

Algorithm	EfficientNUTS
Target distribution	donut
Autoplay	☐
Autoplay delay	285
Tweening delay	18
Step	
Reset	

Visualization Options

Algorithm Options

Leapfrog Δt	0.1

Close Controls

https://chi-feng.github.io/mcmc-demo/