# What Stan Does

Ben Goodrich

February 21, 2019

## Obligatory Disclosure

- Ben is an employee of Columbia University, which has received several research grants to develop Stan and is currently seeking an additional $10 million
- Ben is also a manager of GG Statistics LLC, which uses Stan for business purposes
- According to Columbia University policy, any such employee who has any equity stake in, a title (such as officer or director) with, or is expected to earn at least $\$5,000.00$ per year from a private company is required to disclose these facts in presentations

## Review of Ancient MCMC Samplers

- Metropolis-Hastings (M-H)
    - Only requires user to specify numerator of Bayes Rule
    - But only 22% of proposals ideally get accepted to get relatively big jumps
    - Effective Sample Size $(n_{eff})$ can be essentially zero
- Gibbs sampling (in general)
    - User has to work out all full-conditional distributions
    - Jumps always accepted but might not be very big
    - Effective Sample Size is low if the parameters are highly correlated
- What the BUGS family (WinBUGS, JAGS, OpenBUGS) of software does
    - User does not have to work out full-conditional distributions
    - Falls back to some other algorithm (such as M-H but not actually M-H) when kernel of $k$-th full-conditional distribution is not recognized

## Comparing Stan to Ancient MCMC Samplers

- Like M-H, only requires user to specify numerator of Bayes Rule
- Like M-H but unlike Gibbs sampling, proposals are joint
- Unlike M-H but like Gibbs sampling, proposals always accepted
- Unlike M-H but like Gibbs sampling, tuning of proposals is (often) not required
- Unlike both M-H and Gibbs sampling, the effective sample size is typically 25% to 125% of the nominal number of draws from the posterior distribution because $\rho_1$ can be negative in $n_{eff} = \frac{S}{1+2\sum_{k=1}^{\infty}\rho_k}$
- Unlike both M-H and Gibbs sampling, Stan produces warning messages when things are not going swimmingly. Do not ignore these!
- Unlike BUGS, Stan does not permit discrete unknowns but even BUGS has difficulty drawing discrete unknowns with a sufficient amount of efficiency

# Hamiltonian Monte Carlo

- Instead of simply drawing from the posterior distribution whose PDF is $f\left(\boldsymbol{\theta}|\,\mathbf{y}\dots\right) \propto f\left(\boldsymbol{\theta}\right) L\left(\boldsymbol{\theta};\mathbf{y}\right)$ Stan augments the “position” variables $\boldsymbol{\theta}$ with an equivalent number of “momentum” variables $\boldsymbol{\phi}$ and draws from

$$f\left(\boldsymbol{\theta}|\,\mathbf{y}\dots\right) \propto \int_{-\infty}^{\infty}\dots\int_{-\infty}^{\infty}\prod_{k=1}^{K}\frac{1}{\sigma_k\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{\phi_k}{\sigma_k}\right)^2} f\left(\boldsymbol{\theta}\right) L\left(\boldsymbol{\theta};\mathbf{y}\right) d\phi_1\dots d\phi_K$$

- Since the likelihood is NOT a function of $\phi_k$, the posterior distribution of $\phi_k$ is the same as its prior, which is normal with a “tuned” standard deviation. So, at the $s$-th MCMC iteration, we just draw each $\widetilde{\phi}_k$ from its normal distribution.
- Using physics, the realizations of each $\widetilde{\phi}_k$ at iteration $s$ “push” $\boldsymbol{\theta}$ from iteration $s-1$ through the parameter space whose topology is defined by the negated log-kernel of the posterior distribution: $-\ln f\left(\boldsymbol{\theta}\right) - \ln L\left(\boldsymbol{\theta};\mathbf{y}\right)$
- See HMC.R demo on Canvas

## Demo of Hamiltonian Monte Carlo

Reverse Play Slower Faster Reset 1.00

## No U-Turn Sampling (NUTS)

- The location of $\boldsymbol{\theta}$ moving according to Hamiltonian physics at any instant would be a valid draw from the posterior distribution
- But (in the absence of friction) $\boldsymbol{\theta}$ moves indefinitely so when do you stop?
- Hoffman and Gelman (2014) proposed stopping when there is a “U-turn” in the sense the footprints turn around and start to head in the direction they just came from. Hence, the name No U-Turn Sampling.
- After the U-Turn, one footprint is selected with probability proportional to the posterior kernel to be the realization of $\boldsymbol{\theta}$ on iteration $s$ and the process repeates itself
- NUTS discretizes a continuous-time Hamiltonian process in order to solve a system of Ordinary Differential Equations (ODEs), which requires a stepsize that is also tuned during the warmup phase

## What is Stan?

- Includes a high-level probabilistic programming language
- Includes a translator of high-level Stan syntax to somewhat low-level C++
- Includes new (and old) gradient-based algorithms for statistical inference, such as NUTS
- Includes a matrix and scalar math library that supports autodifferentiation
- Includes interfaces from R and other high-level software
- Includes R packages with pre-written Stan programs
- Includes (not Stan specific) post-estimation R functions
- Includes a large community of users and many developers

## What is Autodifferentiation?

- A language like C++ supports operator overloading of `+`, `-`, etc. to do whatever
- In Stan, `c = a / b` computes `c` and both $\frac{\partial c}{\partial a} = \frac{1}{b}$ and $\frac{\partial c}{\partial b} = -\frac{a}{b^2}$
- Similarly, `d = g(c)` computes `d` and $\frac{\partial d}{\partial c} = g'\left(c\right)$
- Evaluating the chain rule is tedious for a human but easy for a computer
- Autodifferentiation allows the human to write an arbitrary (differentiable) mathematical expression and the (C++) compiler generates the code to compute the derivative automatically, even with vector / matrix expressions
- This is more accurate and / or faster than symbolic differentiation or numerical differentiation
- For Stan's purposes, Stan does autodifferentiation faster than anything else; see http://arxiv.org/abs/1509.07164

## Using Stan via R

1. Write the program in a (text) .stan file w/ R-like syntax that ultimately defines a posterior log-kernel. We will not do this until April. Stan's parser, `rstan::stanc`, does two things
    - checks that program is syntactically valid & tells you if not
    - writes a conceptually equivalent C++ source file to disk
2. C++ compiler creates a binary file from the C++ source
3. Execute the binary from R (can be concurrent with 2)
4. Analyze the resulting samples from the posterior
    - Posterior predictive checks
    - Model comparison
    - Decision

## Drawing from a Bivariate Normal with NUTS

```
library(rstan)
xy <- stan("binormal.stan", refresh = 0)
xy
```

```
## Inference for Stan model: binormal.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##       mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## x     0.03    0.03 1.02 -2.01 -0.66  0.05  0.72  1.98  1123 1.00
## y     0.02    0.03 1.01 -1.98 -0.64  0.03  0.70  1.99  1178 1.00
## lp__ -2.44    0.03 1.06 -5.22 -2.83 -2.11 -1.71 -1.45  1093 1.01
##
## Samples were drawn using NUTS(diag_e) at Thu Feb 21 15:22:04 2019.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

## Divergent Transitions

- NUTS only uses first derivatives
- First order approximations to Hamiltonian physiscs are fine for if either the second derivatives are constant or the discrete step size is sufficiently small
- When the second derviatives are very not constant across $\Theta$, Stan can (easily) mis-tune to a step size that is not sufficiently small and $\theta_k$ gets pushed to $\pm\infty$
- When this happens there will be a warning message, suggesting to increase `adapt_delta`
- When `adapt_delta` is closer to 1, Stan will tend to take smaller steps
- Unfortunately, even as `adapt_delta` $\lim 1$, there may be no sufficiently small step size and you need to try to reparameterize your model

## Exceeding Maximum Treedepth

- When the step size is small, NUTS needs many (small) steps to cross the "typical" subset of $\Theta$ and hit the U-turn point
- Sometimes, NUTS has not U-turned when it reaches its limit of 10 steps (by default)
- When this happens there will be a warning message, suggesting to increase `max_treedepth`
- There is always a sufficiently high value of `max_treedepth` to allow NUTS to reach the U-turn point, but increasing `max_treedepth` by 1 approximately doubles the wall time to obtain $S$ draws

## Low Bayesian Fraction of Missing Information

- When the tails of the posterior PDF are very light, NUTS can have difficulty moving through $\Theta$ efficiently
- This will manifest itself in a low (and possibly unreliable) estimate of $n_{eff}$
- When this happens there will be a warning message, saying that the Bayesian Fraction of Missing Information (BFMI) is low
- In this situation, there is not much you can do except increase $S$ or preferably reparameterize your model to make it easier for NUTS

## Runtime Exceptions

- Sometimes you will get a “informational” (not error, not warning) message saying that some parameter that should be positive is zero or some parameter that should be finite is infinite
- This means that a 64bit computer could not represent the number accurately
- If it only happens a few times and only during the warmup phase, do not worry
- Otherwise, you might try to use functions that are more numerically stable, which is discussed throughout the Stan User Manual