Stan: A platform for Bayesian inference

Andrew Gelman, Bob Carpenter, Matt Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell

Department of Statistics, Columbia University, New York
(and other places)

5 Mar 2014
Eminem - Stan (Short Version) ft. Dido - YouTube
www.youtube.com/watch?v=aSLZFdqwh7E

Artists: Eminem, Dido
Album: No Angel
Released: 1999

Stan (song) - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Stan_(song)
"Stan" is the third single from the The Marshall Mathers LP, recorded in 1999 by American rapper Eminem and featuring British singer Dido. It peaked at number ...
Thank You - Rock City - Robert Browning - Murder ballad

Stan: Project Home Page
mc-stan.org/
Stan modeling language and C++ library for Bayesian inference. NUTS adaptive HMC (MCMC) sampling, automatic differentiation, R, shell interfaces. Gelman.

Urban Dictionary: stan
www.urbandictionary.com/define.php?term=stan
Based on the central character in the Eminem song of the same name, a "stan" is an overzealous maniacal fan for any celebrity or athlete.
Stan is a probabilistic programming language implementing statistical inference with

- Markov chain Monte Carlo sampling (NUTS, HMC)
- Optimization-based point estimation (BFGS)

Stan is coded in C++ and runs on all major platforms (Windows, Mac, Linux).

Stan is freedom-respecting, open-source software.

Interfaces

Download and getting started instructions, organized by interface:

- RStan v2.2.0 (R)
- PyStan v2.2.0 (Python)
- CmdStan v2.2.0 (shell, command-line terminal)

Manual & Examples

Models are portable across interfaces, so these are cross-platform:

- Modeling Language Manual
- Example Models

More Info

Follow the links on the navigation bar (on the right) for information about users and developers discussion groups, bug and feature issue trackers, how to cite Stan and citations of Stan, how to contribute code, the development team, and the source repositories.
data {
  int N;
  int K;
  vector[N] y;
  matrix[N,K] X;
}

parameters {
  vector[K] b;
  real<lower=0> sigma;
}

model {
  y ~ normal(X*b, sigma);
}
data {
  int<lower=1> I; // # questions
  int<lower=1> J; // # students
  int<lower=1> K; // # schools
  int<lower=1> N; // # observations
  int<lower=1,upper=I> ii[N]; // question for n
  int<lower=1,upper=J> jj[N]; // student for n
  int<lower=1,upper=J> kk[N]; // school for n
  int<lower=0,upper=1> y[N]; // correctness for n
}

parameters {
  vector[J] alpha; // ability for student j
  vector[I] beta; // difficulty for item i
  vector[K] gamma; // ability for school k
  vector<lower=0>[I] delta; // discrimination for item i
  real<lower=0,upper=20> sigma_gamma; // school ability scale
}
model {
    alpha ~ normal(0,1);  // priors
    beta ~ normal(0,10);
    gamma ~ normal(0,sigma_gamma);
    delta ~ lognormal(-0.5,1);
    for (n in 1:N) // likelihood
        y[n] ~ bernoulli_logit(delta[ii[n]]
                * ( alpha[jj[n]] + gamma[kk[n]] - beta[ii[n]] ));
}
Stan overview

- Fit open-ended Bayesian models
- Specify log posterior density in C++
- Code a distribution once, then use it everywhere
- Hamiltonian No-U-Turn sampler
- Autodiff
- Runs from R or Python; postprocessing
People

Core:
- Andrew Gelman (stats/poli-sci): R, proto-user ...
- Bob Carpenter (comp sci/linguistics): C++, language design...
- Matt Hoffman (comp sci/acoustics): NUTS, optimization, C++, ...
- Daniel Lee (comp sci/stats): C++, make, integration, ...
- Ben Goodrich (stats/poli-sci): linear algebra, R, C++, ...
- Michael Betancourt (physics): geometric samplers, C++, ...
- Marcus Brubaker (comp sci): linear algebra, optimization, C++, ...
- Jiqiang Guo (stats): R, C++, ...
- Peter Li (stats/math): C++, ...
- Allen Riddell (comp sci): Python, C++, ...

Research collaborators

Users
Funding

- U.S. National Science Foundation
- Novartis
- Columbia University
- U.S. Department of Energy
Roles of Stan

- Bayesian inference for unsophisticated users (alternative to BUGS)
- Bayesian inference for sophisticated users (alternative to programming it yourself)
- Fast and scalable gradient computation
- Environment for developing new algorithms
Example from toxicology

Venous Blood

WELL PERFUSED
\( V_{wp}, P_{wp} \)

POORLY PERFUSED
\( V_{pp}, P_{pp} \)

FAT
\( V_f, P_f \)

LIVER
\( V_l, P_l \)

Exhaled Air

\( ALV \)

\( V_{PR}, P_{ba} \)

F_{wp}, F_{pp}, F_{f}, F_{l}

VMI, KMI

METABOLITES

Stan: A platform for Bayesian inference
Sparse data

Exposure of 72 ppm

Exposure of 144 ppm

Subject:
- A
- F

Exhaled Air Concentration (µg/L)

Blood Concentration (mg/L)

Time (hr)
Validation using predictive simulations

![Graph showing alveolar to inhaled concentration ratio over time.](image)
Public opinion: Health care reform

Should federal gov't spend more money on health care for the uninsured (2004 survey)?

<table>
<thead>
<tr>
<th>Income under $20,000</th>
<th>$20-40,000</th>
<th>$40-75,000</th>
<th>$75-150,000</th>
<th>Over $150,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age 18-29</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30-44</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45-64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>65+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

U.S. avg - 25%         U.S. avg.         U.S. avg. + 25%

U.S. avg is 73%

The state is left blank where a category represents less than 1% of the voters of a state.
Hierarchical time series model
Open-ended
Speed dating

- Each person meets 10–20 “dates”
- Rate each date on Attractiveness, Sincerity, Intelligence, Ambition, Fun to be with, Shared interests
- Outcomes
  - Do you want to see this person again? (Yes/No)
  - How much do you like this person (1–10)
- How important are each of the 6 attributes?
- Logistic or linear regression
- Hierarchical model: coefficients vary by person
Steps of Bayesian data analysis

- Model building
- Inference
- Model checking
- Model understanding and improvement
Background on Bayesian computation

- Point estimates and standard errors
- Hierarchical models
- Posterior simulation
- Markov chain Monte Carlo (Gibbs sampler and Metropolis algorithm)
- Hamiltonian Monte Carlo
Solving problems

- **Problem**: Gibbs too slow, Metropolis too problem-specific
  - **Solution**: Hamiltonian Monte Carlo

- **Problem**: Interpreters too slow, won’t scale
  - **Solution**: Compilation

- **Problem**: Need gradients of log posterior for HMC
  - **Solution**: Reverse-mode algorithmic differentiation

- **Problem**: Existing algo-diff slow, limited, unextensible
  - **Solution**: Our own algo-diff

- **Problem**: Algo-diff requires fully templated functions
  - **Solution**: Our own density library, Eigen linear algebra
“One practical impediment to the use of Hamiltonian Monte Carlo is the need to select suitable values for the leapfrog stepsize, $\epsilon$, and the number of leapfrog steps $L \ldots$ Tuning HMC will usually require preliminary runs with trial values for $\epsilon$ and $L \ldots$ Unfortunately, preliminary runs can be misleading ...”
The No U-Turn Sampler

- Created by Matt Hoffman
- Run the HMC steps until they start to turn around (bend with an angle \( > 180^\circ \))
- Computationally efficient
- Requires no tuning of \#steps
- Complications to preserve detailed balance
Figure 2: Example of a trajectory generated during one iteration of NUTS. The blue ellipse is a contour of the target distribution, the black open circles are the positions $\theta$ traced out by the leapfrog integrator and associated with elements of the set of visited states $B$, the black solid circle is the starting position, the red solid circles are positions associated with states that must be excluded from the set $C$ of possible next samples because their joint probability is below the slice variable $u$, and the positions with a red "x" through them correspond to states that must be excluded from $C$ to satisfy detailed balance. The blue arrow is the vector from the positions associated with the leftmost to the rightmost leaf nodes in the rightmost height-3 subtree, and the magenta arrow is the (normalized) momentum vector at the final state in the trajectory. The doubling process stops here, since the blue and magenta arrows make an angle of more than 90 degrees. The crossed-out nodes with a red "x" are in the right half-tree, and must be ignored when choosing the next sample.

Being more complicated, the analogous algorithm that eliminates the slice variable seems empirically to be slightly less efficient than the algorithm presented in this paper.
NUTS vs. Gibbs and Metropolis

Figure 7: Samples generated by random-walk Metropolis, Gibbs sampling, and NUTS. The plots compare 1,000 independent draws from a highly correlated 250-dimensional distribution (right) with 1,000,000 samples (thinned to 1,000 samples for display) generated by random-walk Metropolis (left), 1,000,000 samples (thinned to 1,000 samples for display) generated by Gibbs sampling (second from left), and 1,000 samples generated by NUTS (second from right). Only the first two dimensions are shown here.

4.4 Comparing the Efficiency of HMC and NUTS

Figure 6 compares the efficiency of HMC (with various simulation lengths $\lambda \approx L$) and NUTS (which chooses simulation lengths automatically). The x-axis in each plot is the target $\delta$ used by the dual averaging algorithm from section 3.2 to automatically tune the step size $\epsilon$. The y-axis is the effective sample size (ESS) generated by each sampler, normalized by the number of gradient evaluations used in generating the samples. HMC's best performance seems to occur around $\delta = 0.65$, suggesting that this is indeed a reasonable default value for a variety of problems. NUTS's best performance seems to occur around $\delta = 0.6$, but does not seem to depend strongly on $\delta$ within the range $\delta \in [0.45, 0.65]$. $\delta = 0.6$ therefore seems like a reasonable default value for NUTS.

On the two logistic regression problems NUTS is able to produce effectively independent samples about as efficiently as HMC can. On the multivariate normal and stochastic volatility problems, NUTS with $\delta = 0.6$ outperforms HMC's best ESS by about a factor of three.

As expected, HMC's performance degrades if an inappropriate simulation length is chosen. Across the four target distributions we tested, the best simulation lengths $\lambda$ for HMC varied by about a factor of 100, with the longest optimal $\lambda$ being 17.62 (for the multivariate normal) and the shortest optimal $\lambda$ being 0.17 (for the simple logistic regression). In practice, finding a good simulation length for HMC will usually require some number of preliminary runs. The results in Figure 6 suggest that NUTS can generate samples at least as efficiently as HMC, even discounting the cost of any preliminary runs needed to tune HMC's simulation length.
NUTS vs. Basic HMC

- 250-D normal and logistic regression models
- Vertical axis shows effective #sims (big is good)
- (Left) NUTS; (Right) HMC with increasing $t = \epsilon L$
NUTS vs. Basic HMC II

- Hierarchical logistic regression and stochastic volatility
- Simulation time is step size $\epsilon$ times $\#$steps $L$
- NUTS can beat optimally tuned HMC

**Hierarchical Logistic Regression**

![Graph showing ESS per gradient vs. target acceptance rate statistic $\delta$ for NUTS and HMC with different $\epsilon L$ values.]

**Stochastic Volatility**

![Graph showing ESS per gradient vs. target acceptance rate statistic $\delta$ for NUTS and HMC with different $\epsilon L$ values.]

Gelman Carpenter Hoffman Lee Goodrich Betancourt...

Stan: A platform for Bayesian inference
Solving more problems in Stan

- **Problem**: Need ease of use of BUGS
  - **Solution**: Compile directed graphical model language

- **Problem**: Need to tune parameters for HMC
  - **Solution**: Auto tuning, adaptation

- **Problem**: Efficient up-to-proportion density calcs
  - **Solution**: Density template metaprogramming

- **Problem**: Limited error checking, recovery
  - **Solution**: Static model typing, informative exceptions

- **Problem**: Poor boundary behavior
  - **Solution**: Calculate limits (e.g. $\lim_{x \to 0} x \log x$)

- **Problem**: Restrictive licensing (e.g., closed, GPL, etc.)
  - **Solution**: Open-source, BSD license
Example: the “Kumaraswamy distribution”

\[ p(\theta|a, b) = ab \theta^{\theta-1}(1 - \theta^a)(b - 1) \text{ for } a, b > 0 \text{ and } \theta \in (0, 1) \]

model {
  // Put priors on a and b here if you want

  // Put in the rest of your model

  // Kumaraswamy log-likelihood
  increment_log_prob(N*(log(a)+log(b))+(a-1)*sum_log_theta);
  for (n in 1:N)
    increment_log_prob((b-1)*log1m(pow(theta[n],a)));
}
Check that it worked

R code:

```r
N <- 1000
a <- 3
b <- 2
theta <- rbeta(N,1,b)^(1/a)
Kumaraswamy <- stan (file="Kumaraswamy.stan", data=c("N","theta"))
print (Kumaraswamy)
```

Result:

Inference for Stan model: Kumaraswamy.
4 chains: each with iter=2000; warmup=1000; thin=1; 2000 saved.

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>sd</th>
<th>2.5%</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>97.5%</th>
<th>n_eff</th>
<th>Rhat</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>2.9</td>
<td>0.1</td>
<td>2.7</td>
<td>2.9</td>
<td>2.9</td>
<td>3.0</td>
<td>3.1</td>
<td>766</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1.9</td>
<td>0.1</td>
<td>1.8</td>
<td>1.9</td>
<td>1.9</td>
<td>2.0</td>
<td>2.1</td>
<td>787</td>
<td>1</td>
</tr>
<tr>
<td>lp__</td>
<td>282.4</td>
<td>1.0</td>
<td>279.7</td>
<td>282.1</td>
<td>282.7</td>
<td>283.1</td>
<td>283.4</td>
<td>683</td>
<td>1</td>
</tr>
</tbody>
</table>
Thinking about scalability

Hierarchical item response model:

<table>
<thead>
<tr>
<th># items</th>
<th># raters</th>
<th># groups</th>
<th># data</th>
<th>Stan</th>
<th>memory</th>
<th>JAGS</th>
<th>memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>2,000</td>
<td>100</td>
<td>40,000</td>
<td>:02m</td>
<td>16MB</td>
<td>:03m</td>
<td>220MB</td>
</tr>
<tr>
<td>40</td>
<td>8,000</td>
<td>200</td>
<td>320,000</td>
<td>:16m</td>
<td>92MB</td>
<td>:40m</td>
<td>1400MB</td>
</tr>
<tr>
<td>80</td>
<td>32,000</td>
<td>400</td>
<td>2,560,000</td>
<td>4h:10m</td>
<td>580MB</td>
<td>:??m</td>
<td>?MB</td>
</tr>
</tbody>
</table>

Also, Stan generated 4x effective sample size per iteration
Successes and struggles

- Simple linear and logistic regression
  - Simple, extendable, full Bayes
- Hierarchical item response model
  - Scalable, fast
- Discrete parameters
  - Mixture formulation
- Cormack-Jolly-Seber mark-recapture model
  - 2 minutes for Stan vs. overnight for Bugs
- Diff eq models in pharmacology
  - Stability, priors, multilevel, full Bayes
- Age-period-cohort model of voting
  - Modular model building
- 5000 downloads, 800 people on users list
  - Existing programs don’t run, don’t converge, or are slow
Future work

- Programming
  - Faster gradients and higher-order derivatives
  - Functions

- Statistical algorithms
  - Riemannian Hamiltonian Monte Carlo
  - (Penalized) mle
  - (Penalized) marginal mle
  - Black-box variational Bayes
  - Data partitioning and expectation propagation