Mock Your RNGs

Lessons in testing Stan,
a probabilistic programming language

Daniel Lee
Stan Development Team / Generable
Who's heard of Stan?

(we usually don't run in AI circles)
What is Stan?
Language for Statistical Models

• **Goal:** specify statistical models
Language for Statistical Models

• **Goal:** specify statistical models

\[ \theta, x \]

- parameters
- data
Language for Statistical Models

- **Goal:** specify statistical models

\[ p(\theta, x) \]

- model
- parameters
- data
Language for Statistical Models

- **Goal:** specify statistical models

- Probabilistic programming language (has its own grammar)
  - imperative, statically typed
  - Turing complete

- User can specify any *differentiable* joint probability distribution function over data and parameters

\[ p(\theta, x) \]
Example: Hello World

data {
}

parameters {
}

model {
    print("hello world!");
}
Example: Logistic Regression

data {
  int<lower=0> N;
  vector[N] x;
  int<lower=0, upper=1> y[N];
}

parameters {
  real alpha;
  real beta;
}

model {
  y ~ bernoulli_logit(alpha + beta * x);
}
Language defines the statistical model

\[ p(\theta, x) \]
Inference algorithms use $p(\theta, x)$

- Bayesian inference; Markov Chain Monte Carlo (MCMC)
- Approximate Bayesian inference
- Optimization
Inference algorithms use $p(\theta, x)$

- Bayesian inference; Markov Chain Monte Carlo (MCMC)
  - $p(\theta | x)$ approximated with $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(N)}\}$

- Approximate Bayesian inference
  - ex: $\hat{p}(\theta | x) \approx q(\hat{\phi})$ where $\hat{\phi} = \arg\min_{\phi} D_{KL}(q(\theta | \phi) \| p(\theta, x))$

- Optimization
  - $\hat{\theta} = \arg\max_{\theta} p(\theta, x)$ (only holds when there's a single optima)
Inference algorithms use $p(\theta, x)$

- Bayesian inference; Markov Chain Monte Carlo (MCMC)
  - $p(\theta | x)$ approximated with $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(N)}\}$

- Approximate Bayesian inference
  - ex: $\hat{\theta} = \text{argmax}_{\theta} p(\theta, x)$

- Optimization
  - $\hat{\theta} = \text{argmax}_{\theta} p(\theta, x)$ (only holds when there's a single optima)
Interfaces

- CmdStan, RStan, PyStan
- C++ API
- C++ automatic differentiation library
- RStanArm, brms, prophet, ...
Stan: mc-stan.org

- Language
- Inference algorithms
- Interfaces

- Open Source:
  - BSD: lang, math, CmdStan
  - GPL: RStan, PyStan

- github.com/stan-dev
If you can't remember anything else in this talk...
Mock your random number generators
Testing challenges in Stan

• Inference algorithms are randomized algorithms
  Markov chain Monte Carlo (MCMC)

  • Run the same input with a different seed, the output changes

  • These algorithms have nice asymptotic properties, but... we don't have asymptotic time
Testing challenges in Stan

- Inference algorithms are randomized algorithms
  Markov chain Monte Carlo (MCMC)
  - Run the same input with a different seed, the output changes
  - These algorithms have nice asymptotic properties, but... we don't have asymptotic time
- Stan is pretty complex (~100k sloc, ~250k test sloc)
  - Stan programs transpiled to C++
  - automatic differentiation library: heavily templated C++
  - custom derivatives for speed
Testing is...

- Testing software: tough
Testing is...

- Testing software: tough
- Testing a programming language: tougher
Testing is...

- Testing software: tough
- Testing a programming language: tougher
- Testing randomized algorithms that operate on a programming language... priceless
Things we know

• Computers aren't random (either hardware failure or unsafe memory access)
  • use of pseudo random number generators

• Software: modularity combats complexity
  • How do we extend this to testing randomized algorithms?
Mock your RNGs

• Mock your random number generators
  • Test the randomized algorithm with a known sequence of (not so random any more) random numbers
  • Reproducible, easier to break down code
  • Acts like any other unit test
  • (Beware... IEEE floating point behavior can still get you)

• (Also test asymptotic properties for correctness)
Thank you!
Breakthroughs in stats computing

- Inference algorithms
- Autodiff
250 dimensional Normal distribution

2d projection

<table>
<thead>
<tr>
<th>Metropolis</th>
<th>Gibbs</th>
<th>NUTS</th>
<th>Independent</th>
</tr>
</thead>
</table>

1000 draws

Quick aside: HMC is not enough

- Hamiltonian Monte Carlo isn't enough

Autodiff comparison

- For open-source C++ packages: Stan is **fastest** (for gradients), most **general** (functions supported), and most easily **extensible** (simple OO).