Comparing Bayesian Computational Strategies

Evan Miyakawa

Supervised by David J. Kahle, Ph.D.

Baylor University
COLLEGE OF ARTS & SCIENCES
Statistical Science

Overview

1. Introduction to bayesian paradigm
2. Bayesian computational methods
3. Implementations
4. Data / model specification
5. Benchmarking
6. Conclusion / further research

Introduction to Bayesian Paradigm

Conditional probability:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

Bayes theorem:

$$p(\theta|\mathbf{x}) = \frac{\ell(\theta|\mathbf{x})\pi(\theta)}{\int \ell(\theta|\mathbf{x})\pi(\theta)d\theta}$$

$\ell(\theta|\mathbf{x})$ is the likelihood, and $\pi(\theta)$ is the prior distribution

Important components:

- Prior distribution
- Conditioning on the data

$$p(\theta|\mathbf{x}) = \frac{\ell(\theta|\mathbf{x})\pi(\theta)}{\int \ell(\theta|\mathbf{x})\pi(\theta)d\theta}$$

Important components:

- Prior distribution

- Conditioning on the data

$$p(\theta|\mathbf{x}) = \frac{\ell(\theta|\mathbf{x})\pi(\theta)}{\int \ell(\theta|\mathbf{x})\pi(\theta)d\theta}$$

Challenges:

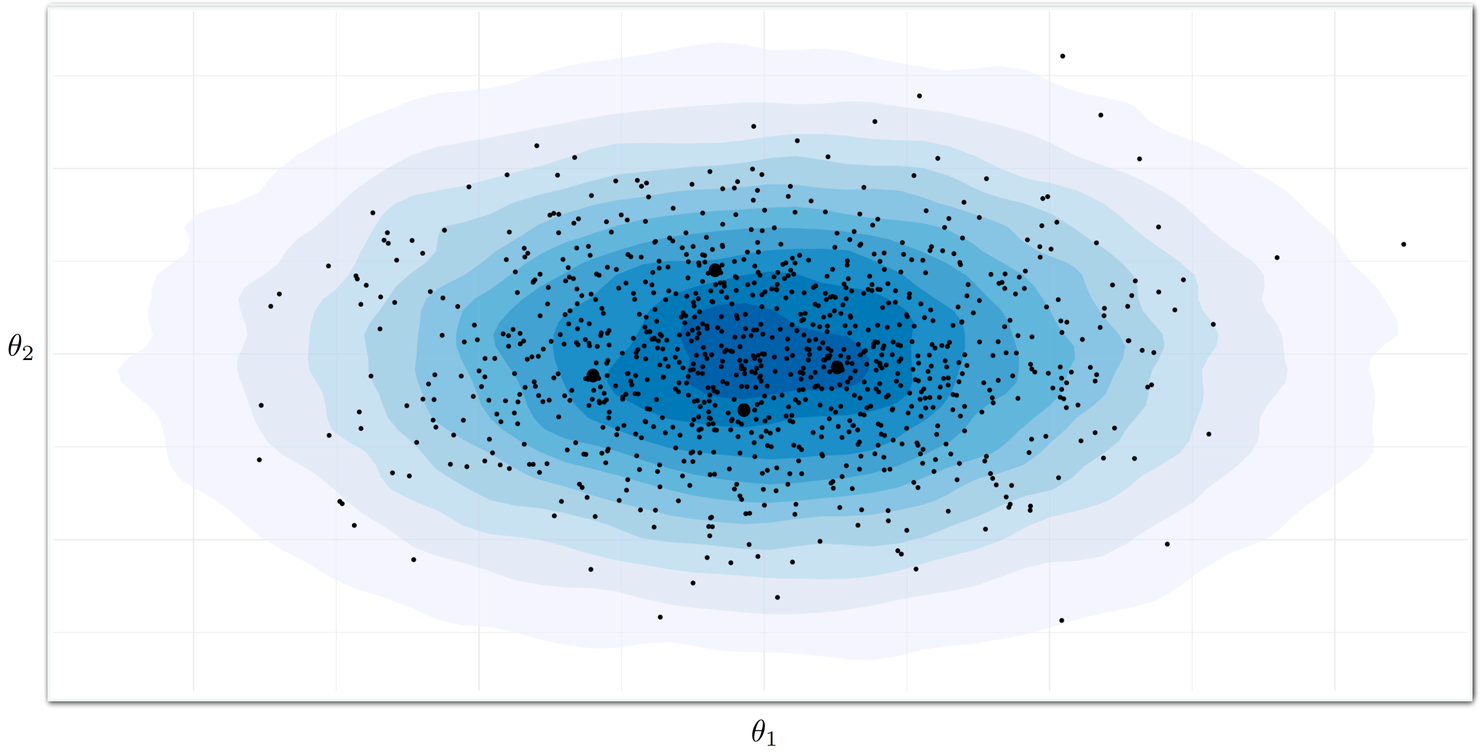
- Taking integrals is hard

- Analytical posterior calculation is often impossible if we have a complex model or non-conjugate distributions

Bayesian Computational Methods

Goal

- Approximate the posterior distribution by getting draws from the posterior



Markov Chain Monte Carlo

- Markov chain conditions on most recent draw
- Each draw more likely to come from posterior
- Marginal distributions of draws asymptotically converge to true posterior

Gibbs sampling

- Break down complex parameter spaces into one-dimensional conditional distributions

- Each single parameter MCMC draw conditioned on most recent values of other parameters

- Random walk behavior

Gibbs algorithm:

1. Choose starting values
2. Sample:

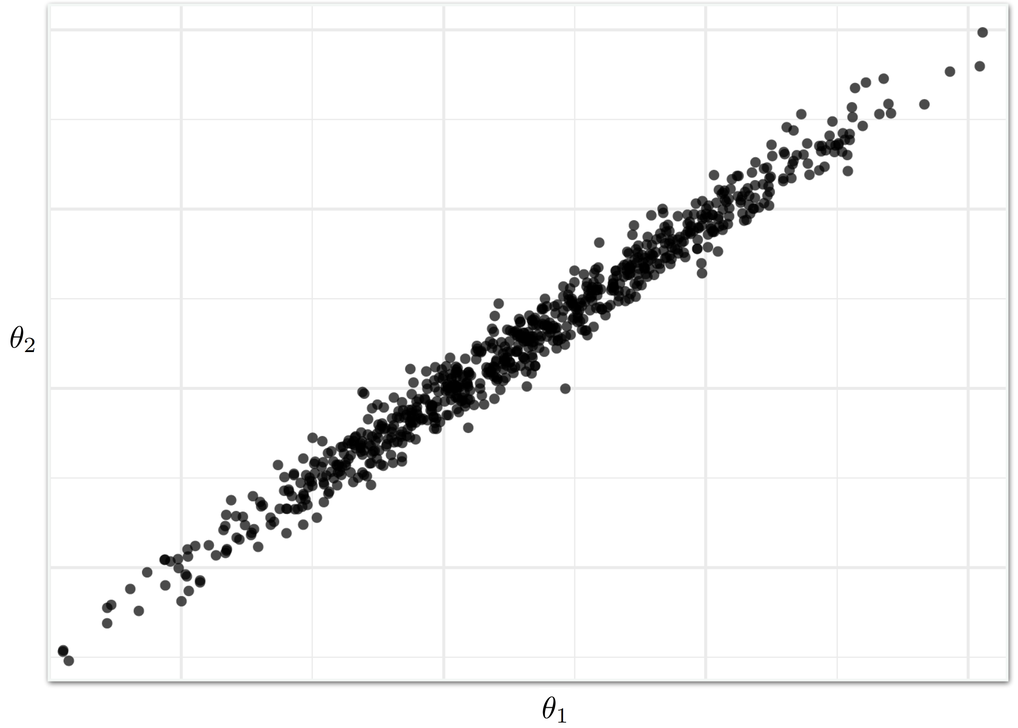
   $\theta_1^{(1)} \sim p(\theta_1 | \theta_2^{(0)}, \theta_3^{(0)}, \ldots, \theta_k^{(0)}, y)$
3. Repeat for all parameters
4. Do entire process for N iterations

Gibbs sampling strengths:

- Complex posteriors made more simple

- Easy implementation

- Simpler explanation

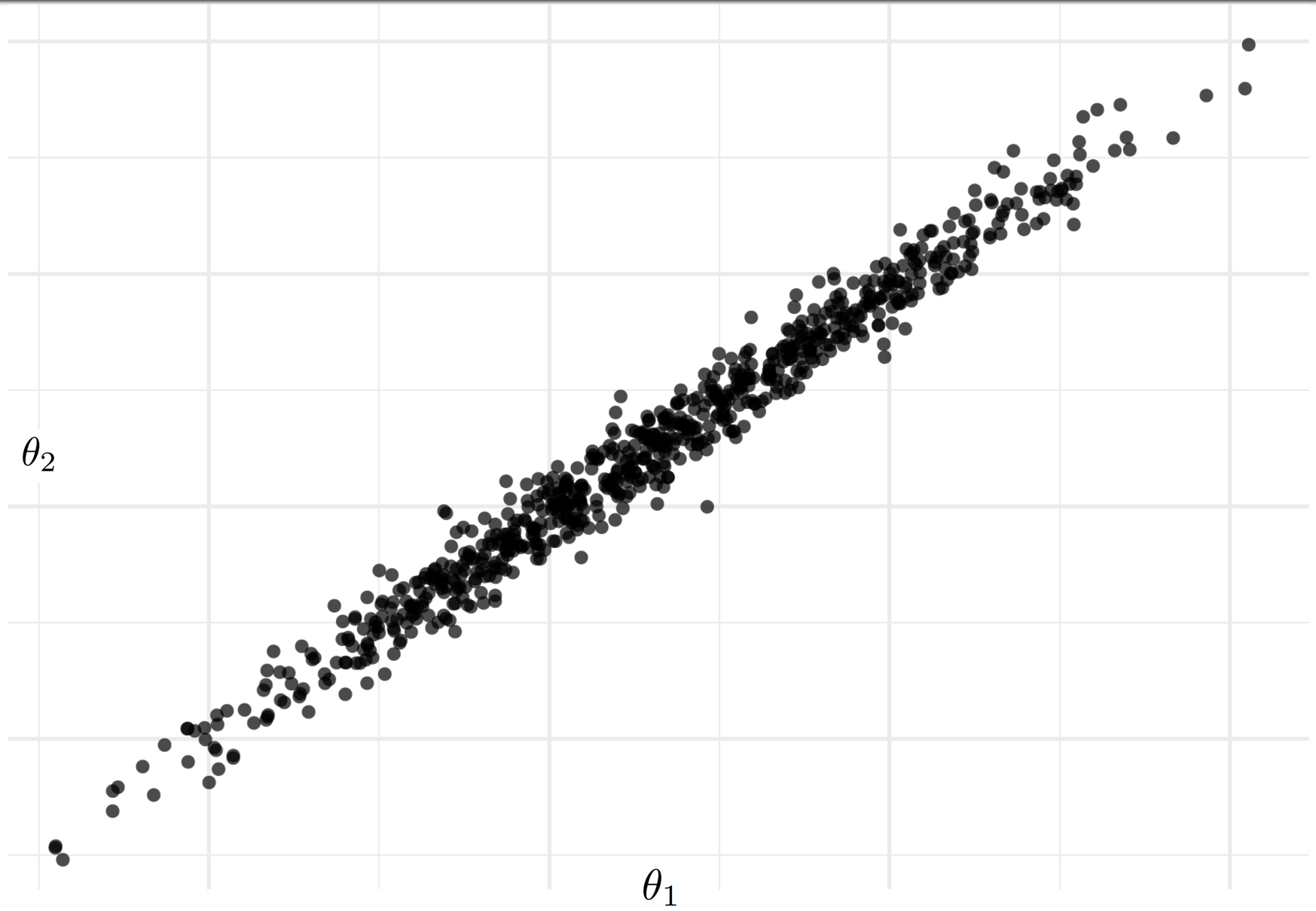


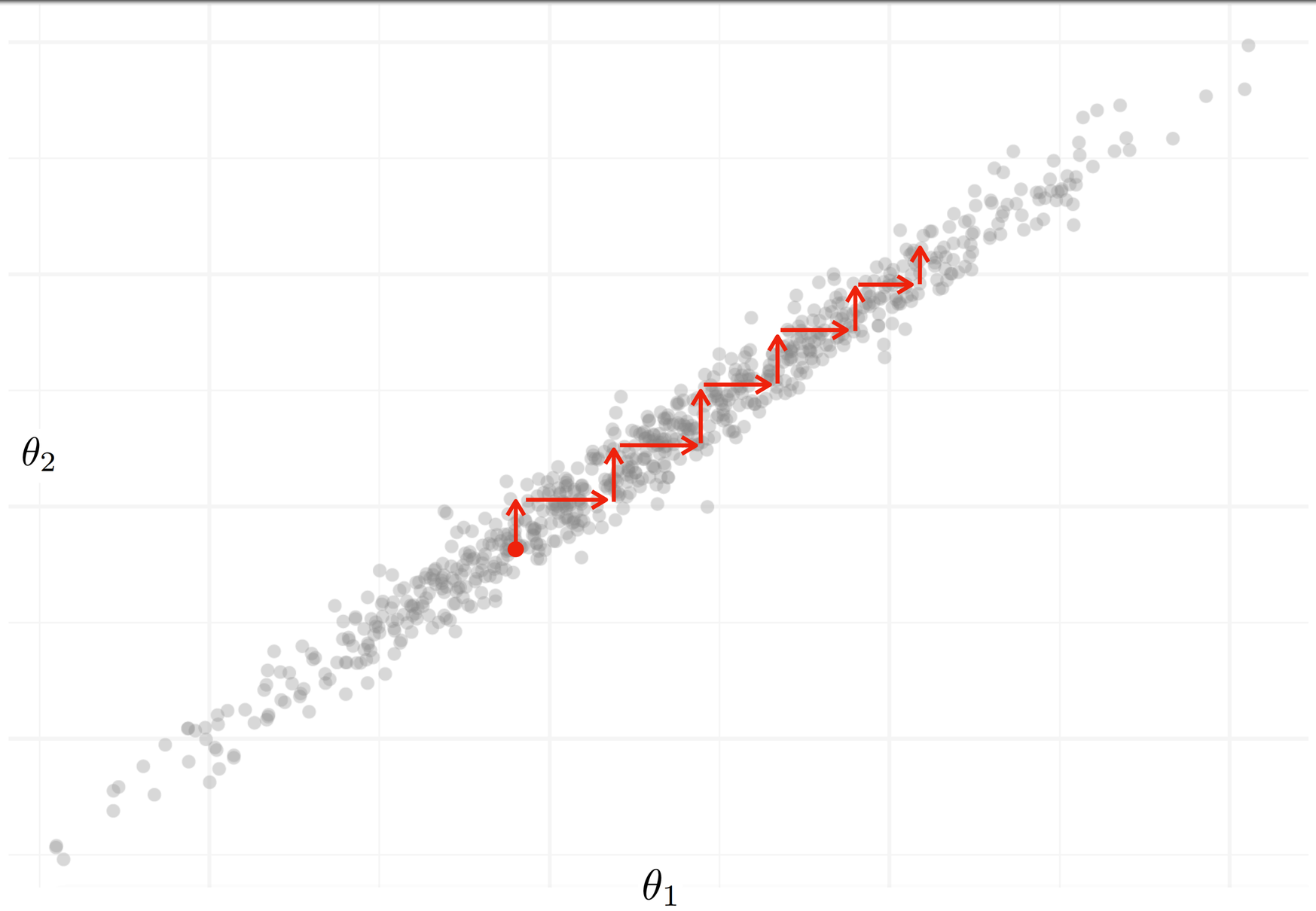
Gibbs sampling weaknesses:

- Must have full conditional distributions

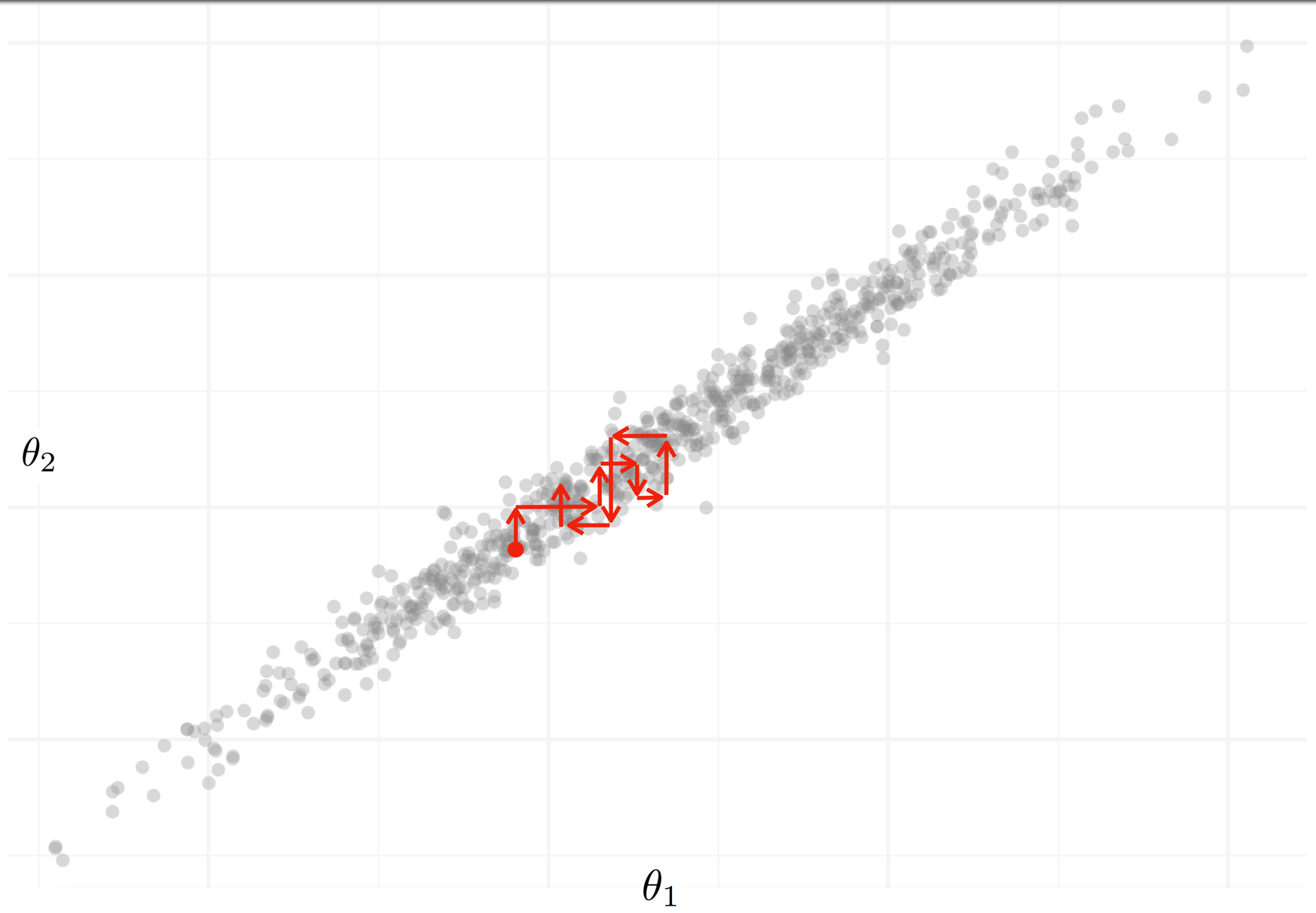
- Very small step-sizes needed for highly correlated variables

Used by default in WinBUGS/OpenBUGS, JAGS, and Nimble


θ_2
θ_1


θ_2
θ_1

Computational Methods



Hamiltonian Monte Carlo

- Use the geometry of the target distribution to guide our sampling

- Translate target distribution density function into a potential energy function

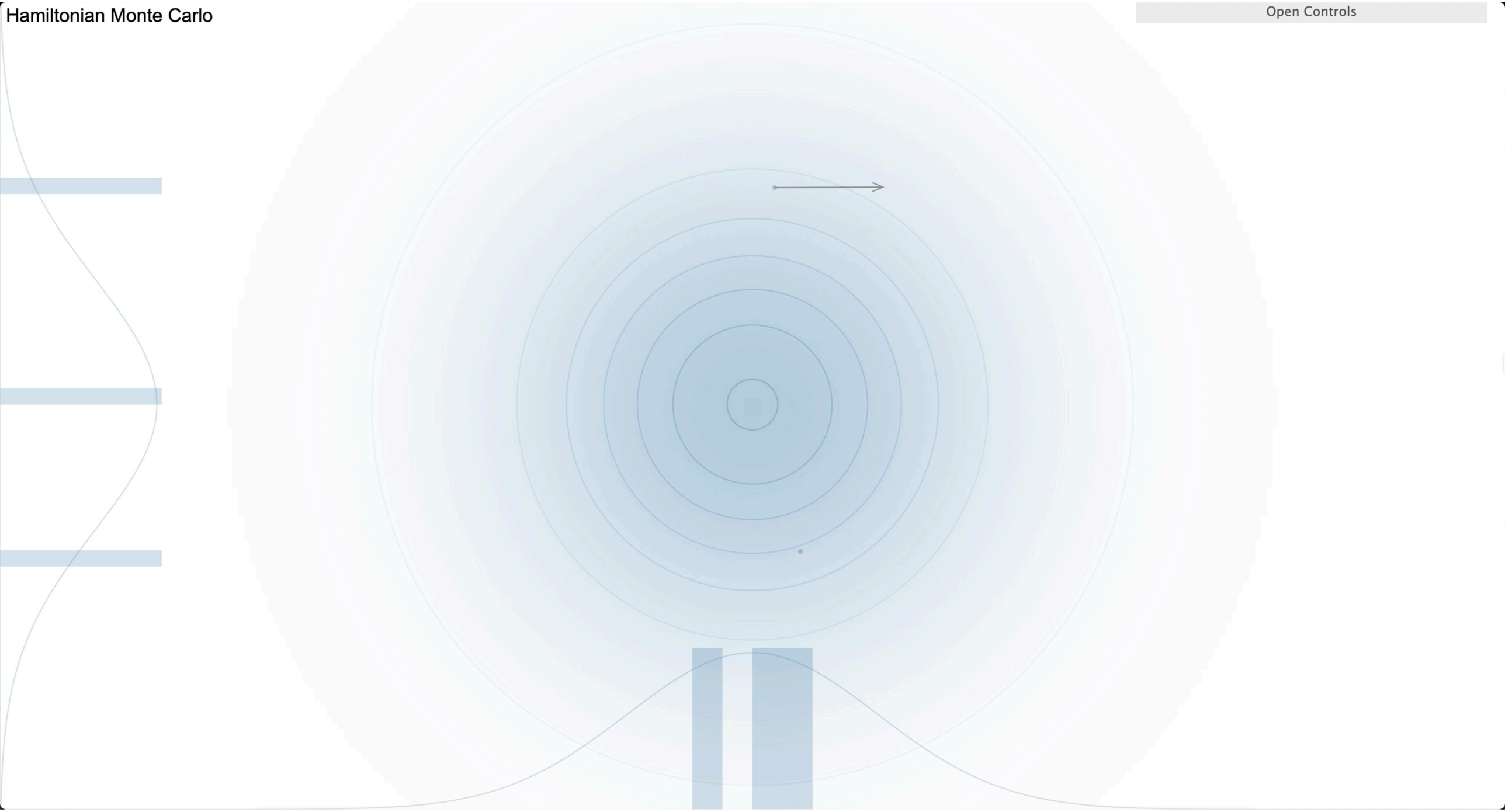
- Each parameter transformed into a position variable (q) with an associated momentum variable (p)

Joint distribution of q and p:

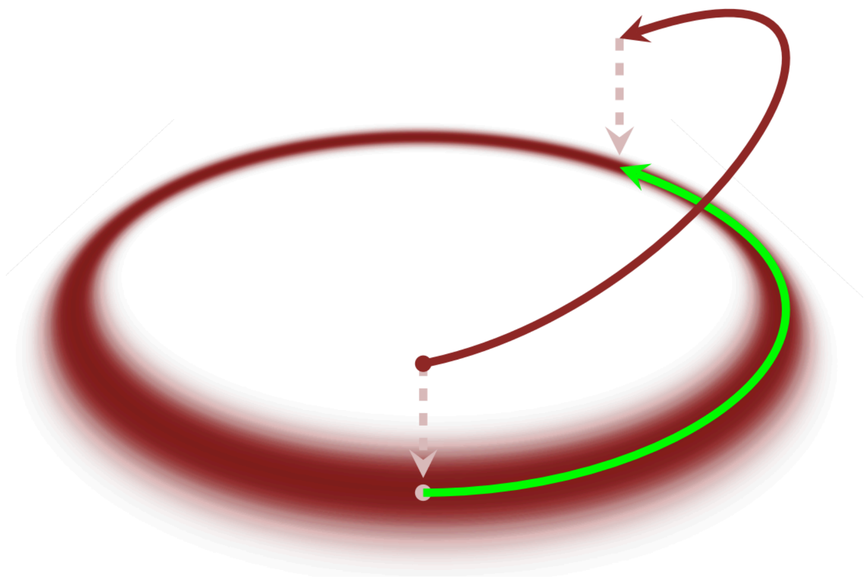
$$\pi(q, p) = \pi(p|q)\pi(q)$$

Hamiltonian Monte Carlo

Open Controls



Hamiltonian Monte Carlo



Lifting up off the parameter space to sample when using HMC

HMC algorithm:

1. Choose starting values
2. Random draws for momentum variables
3. Use momentum to propose new state for position variables
4. Accept / reject new state based on HMC probability function used
5. Project down into parameter space

Hamiltonian Monte Carlo strengths:

- Runs into fewer sampling issues than Gibbs sampling
- Efficient
- Sampling issues easier to spot

Weaknesses:

- Additional challenge to solve system of differential equations
- Lots of tuning parameters to pick

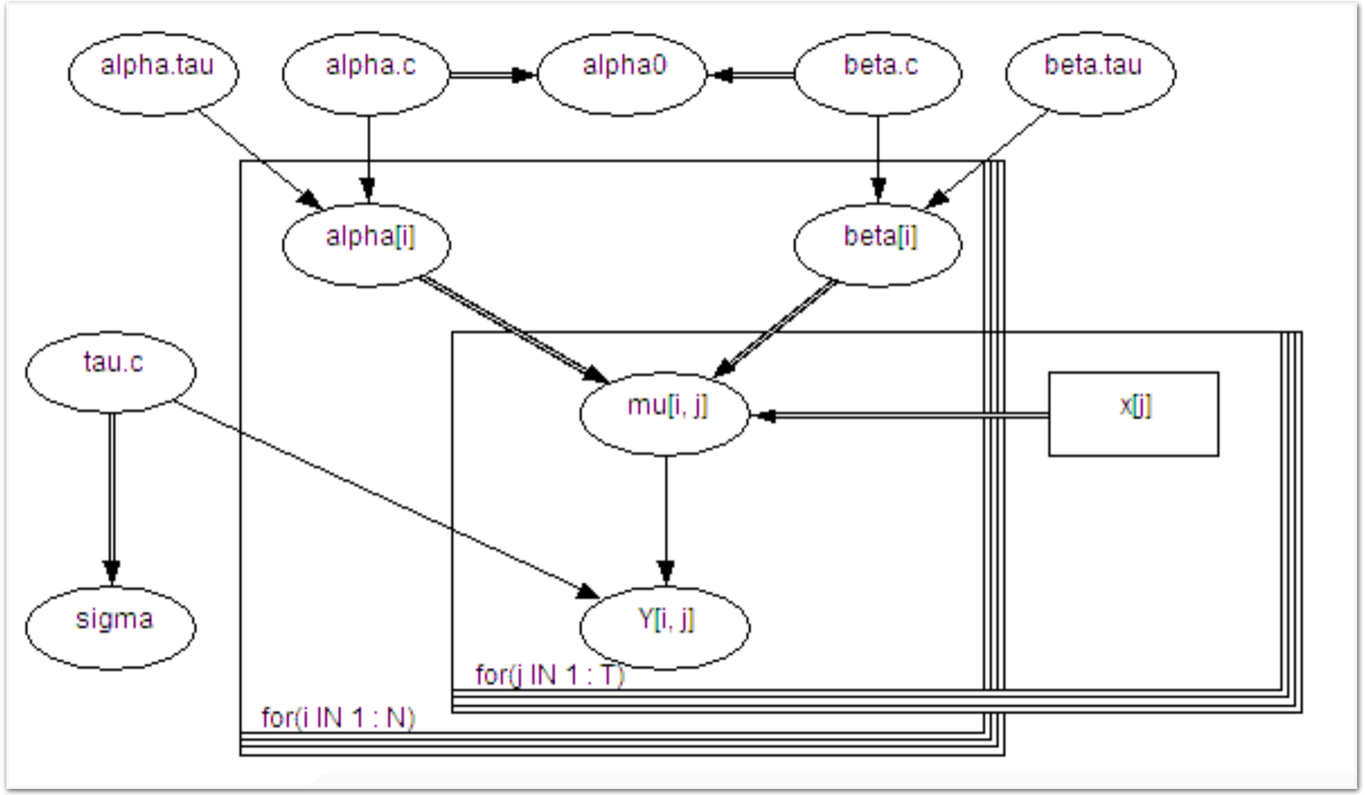
Stan and Greta use HMC

Implementations

OpenBUGS

- Earliest commonly used Bayesian sampling method
- Uses BUGS model specification language
- Windows-based software
- R can interact through R2OpenBUGS package
- Can show graphical representation of models

OpenBUGS


alpha.tau
alpha.c
alpha0
beta.c
beta.tau
alpha[i]
beta[i]
tau.c
mu[i, j]
x[j]
sigma
Y[i, j]
for(j IN 1 : T)
for(i IN 1 : N)

JAGS

- Interact with JAGS entirely in R
- Based on BUGS model specification language
- C++ under the hood
- Model script inputted as string
- Lots of documentation

Nimble

- R package developed for flexible statistical modeling
- Compiles models in C++ for speed
- Based on BUGS model specification language
- Can write custom algorithms to be compiled
- Compilation takes time

Stan

- Unique programming language for bayesian computation
- Available for R, Python, MATLAB, on any machine
- Uses C++ under the hood
- Stores compiled objects for repeated use
- Extensive documentation

Greta

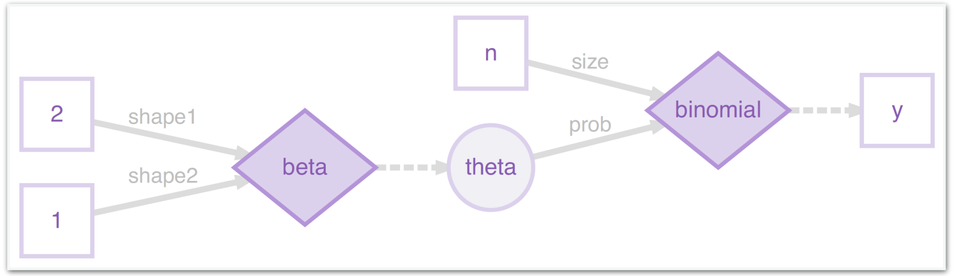
- Build statistical objects directly in R

- HMC / Google Tensorflow for calculation

- Flexible delegation of machine resources

- Offers visualization of model relationships

- Documentation not as robust



Data preparation

Basic data for a beta-binomial model

```r
# Prepare Data

p <- .25 # binomial p
n <- 10 # binomial n

set.seed(1)

y <- rbinom(1, n, p)
```

```r
# Put data in a list

model_data <- list(
  "y" = y,
  "n" = n
)
```

```r
# configure model settings

n_chains <- 4
n_iter <- 10000
n_initial <- 1000
```

Specify separate “constants” block for Nimble

```
# Put data and constants in separately

model_data <- list(
  "y" = y
)

nimble_constants <- list(
  "n" = n
)
```

Model Specification

Model Specification — OpenBUGS

Baylor University

model.bugs

```
model{
   y ~ dbin(p,n)
   p ~ dbeta(1,1)
}
```

bugs_model.R

```
library(R2OpenBUGS)

bugs_monitor <- "p"

bugs_fit <- bugs(
   "model.file" = "model.bugs",
   "data" = model_data,
   "parameters.to.save" = bugs_monitor,
   "inits" = NULL,
   "n.chains" = n_chains,
   "n.iter" = n_iter,
   "n.burnin" = n_initial
)
```

Model Specification — JAGS

Baylor University

model.jags

```
model{
   y ~ dbin(p,n)
   p ~ dbeta(1,1)
}
```

jags_model.R

```
library(rjags); library(runjags)

jags_monitor <- "p"

jags_fit <- run.jags(
  "model" = "model.jags",
  "data" = model_data,
  "monitor" = jags_monitor,
  "n.chains" = n_chains,
  "sample" = n_iter,
  "burnin" = n_initial
)
```

Model Specification — Nimble

Baylor University

nimble_model.R

```r
library(nimble)

nimble_model <- nimbleCode({
  y ~ dbin(p,n)
  p ~ dbeta(1,1)
)}

nimble_inits <- list(
  "p" = rbeta(1,1,1)
)
```

```r
nimble_monitor <- "p"

nimble_fit <- nimbleMCMC(
  "code" = nimble_model,
  "data" = model_data,
  "inits" = nimble_inits,
  "monitors" = nimble_monitor,
  "nchains" = n_chains,
  "niter" = n_iter,
  "nburnin" = n_initial,
  "summary" = TRUE
)
```

Model Specification — Stan

Baylor University

model.stan

```
data {
  int<lower=0> y;
  int<lower=0> n;
}

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

model {
  y ~ binomial(n,p);
  p ~ beta(1,1);
}
```

stan_model.R

```
# load library
library(rstan)

# Run the model
stan_fit <- stan(
  "file" = "model.stan",
  "data" = model_data,
  "chains" = n_chains,
  "iter" = n_iter,
  "warmup" = n_initial
)
```

Model Specification — Greta

Baylor University

greta_model.R

```
library(greta)

y <- as_data(y)
n <- as_data(n)

p <- beta(1,1)

distribution(y) <- binomial(n,p)
```

```
greta_model <- model(p)

greta_fit <- mcmc(
  "model" = greta_model,
  "n_samples" = n_iter,
  "warmup" = n_initial,
  "chains" = n_chains
)
```

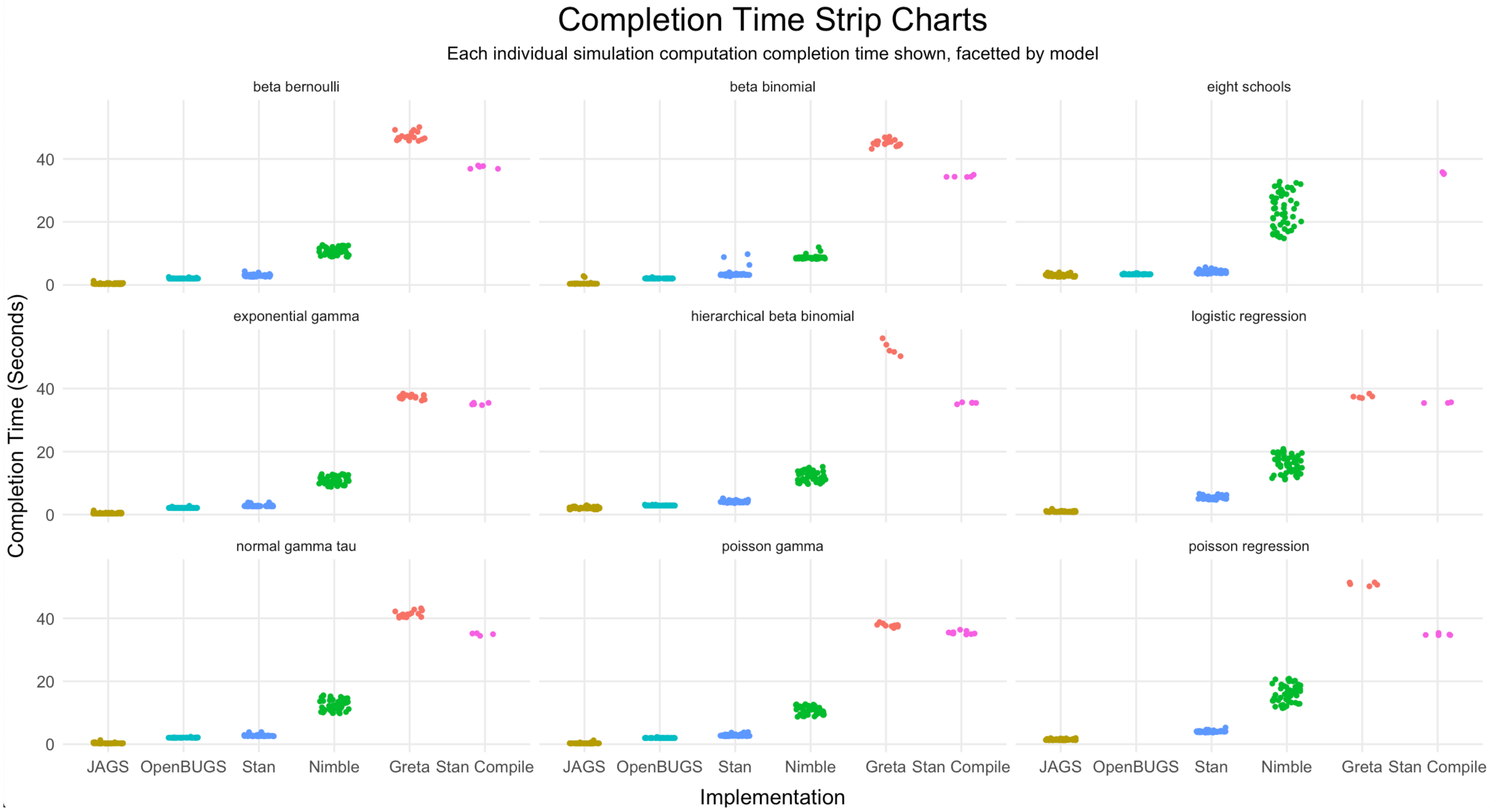
Benchmarking

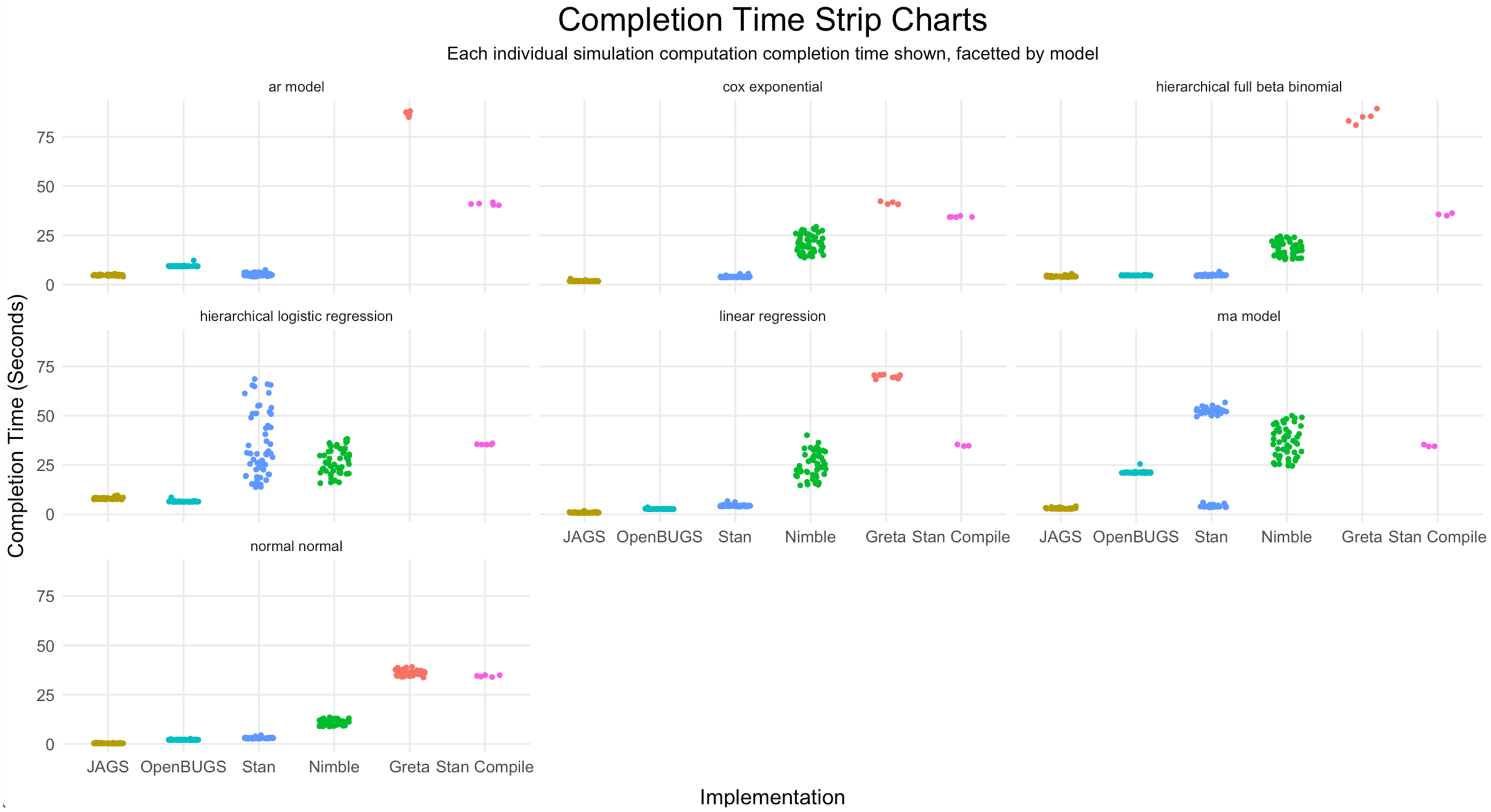
Comparing Implementations

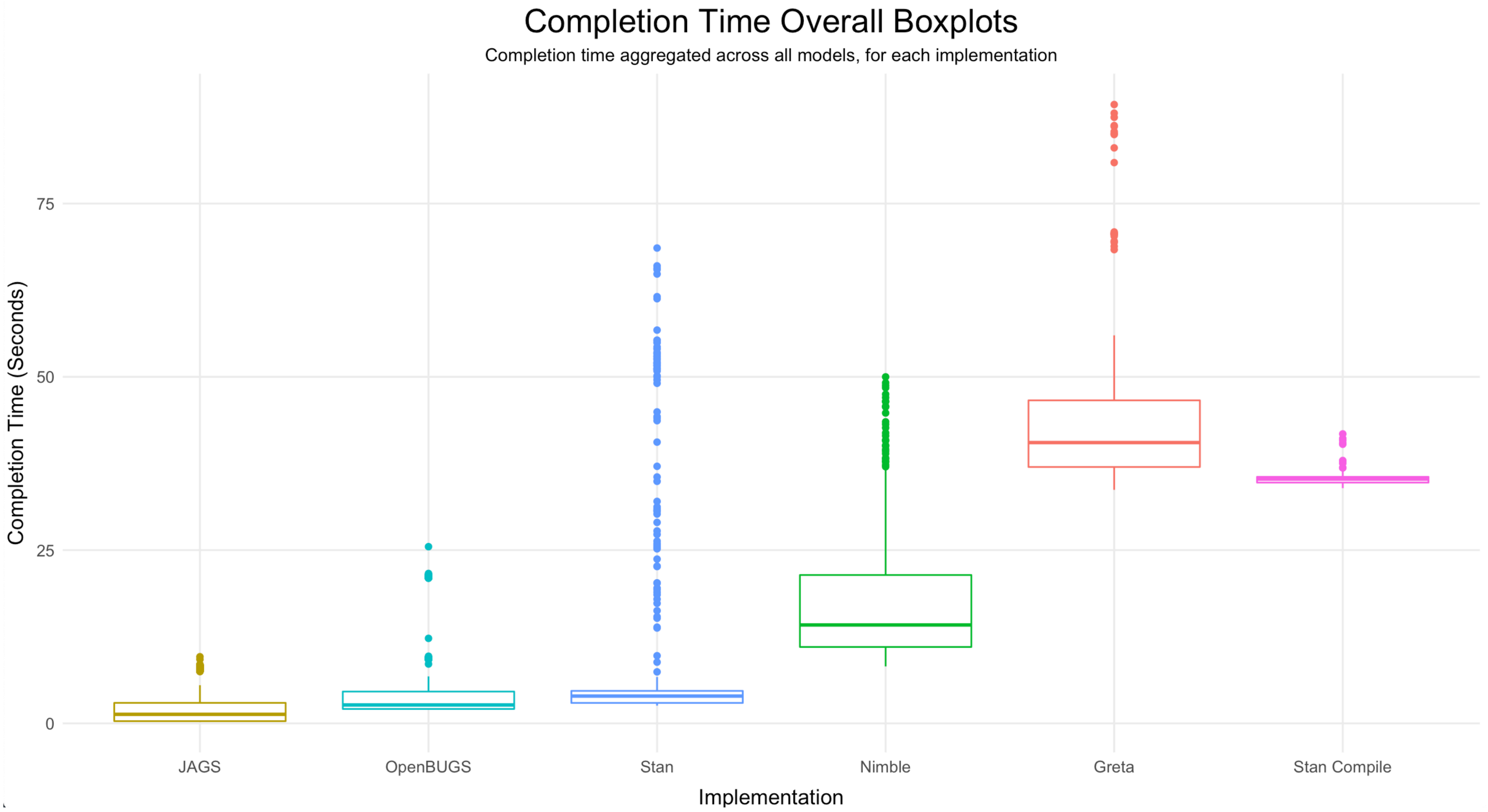
1. Specifying models
2. Computation time
3. Posterior estimation accuracy

Computation time benchmarking method

- Sampling done on 16 different models
    - Simple conjugate models, regression models, hierarchical models, time series, survival, and others
- MCMC with four chains, 10,000 iterations, 1000 warmup iterations
- 50 simulations for each method/implementation measuring computation time


Completion Time Strip Charts
Each individual simulation computation completion time shown, facetted by model
beta bernoulli
beta binomial
eight schools
exponential gamma
hierarchical beta binomial
logistic regression
normal gamma tau
poisson gamma
poisson regression
Completion Time (Seconds)
0
20
40
JAGS
OpenBUGS
Stan
Nimble
Greta Stan Compile
Implementation


Completion Time Strip Charts
Each individual simulation computation completion time shown, facetted by model
ar model
cox exponential
hierarchical full beta binomial
hierarchical logistic regression
linear regression
ma model
normal normal
Completion Time (Seconds)
0
25
50
75
JAGS
OpenBUGS
Stan
Nimble
Greta Stan
Compile
Implementation


Completion Time Overall Boxplots
Completion time aggregated across all models, for each implementation
Completion Time (Seconds)
75
50
25
0
JAGS
OpenBUGS
Stan
Nimble
Greta
Stan Compile
Implementation

Comparing implementations

1. Specifying models
2. Computation time
3. Posterior estimation accuracy

Posterior estimation accuracy benchmarking method

- Total variation distance used to assess the distance between true posterior and empirical posterior:

$$\delta(P, Q) = \sup_{A \in \mathcal{B}} \left| P(A) - Q(A) \right| = \frac{1}{2} \int_{\mathbb{R}} |p(x) - q(x)| \, dx$$

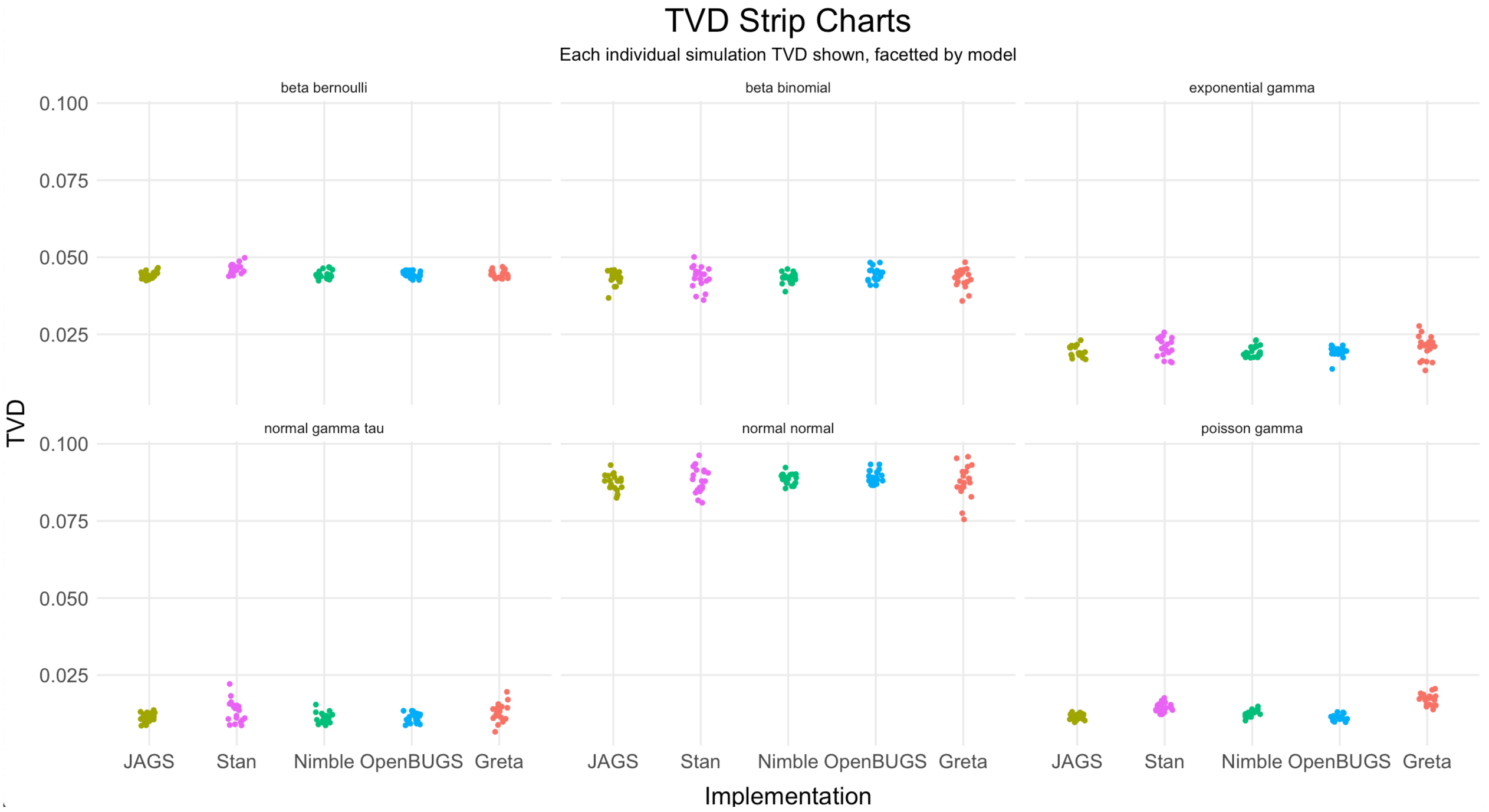
Posterior estimation accuracy benchmarking method

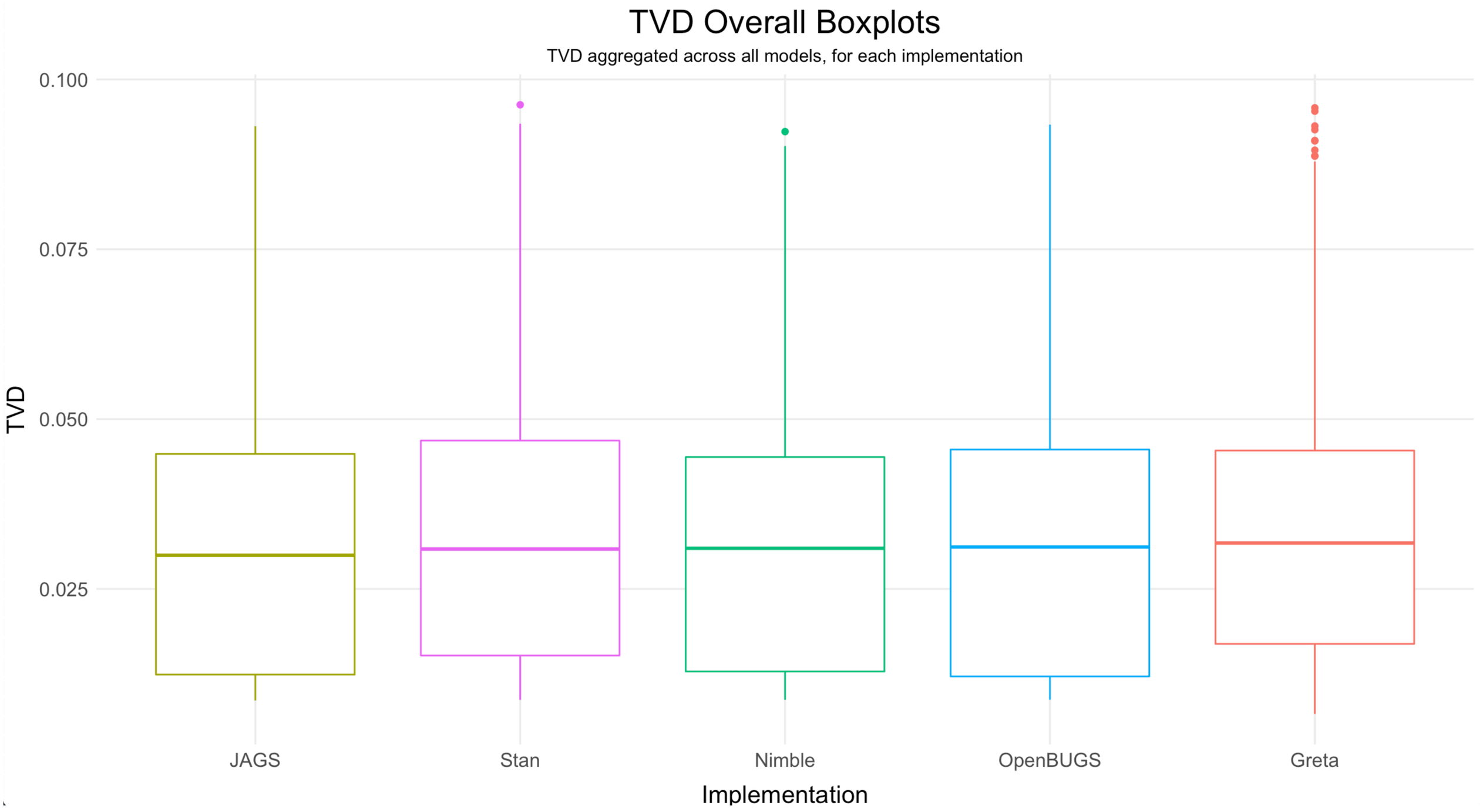
- Sampling done on 6 conjugate models with known posteriors

    1. For each model, sampling done using each implementation

    2. TVD calculated for difference between true posterior and empirical posterior for each implementation

- This process repeated many times for each model


TVD Strip Charts
Each individual simulation TVD shown, facetted by model
beta bernoulli
beta binomial
exponential gamma
normal gamma tau
normal normal
poisson gamma
0.100
0.075
0.050
0.025
TVD
JAGS
Stan
Nimble
OpenBUGS
Greta
Implementation


TVD Overall Boxplots
TVD aggregated across all models, for each implementation
0.100
0.075
0.050
0.025
TVD
JAGS
Stan
Nimble
OpenBUGS
Greta
Implementation

Conclusion

Conclusion

Takeaways

- There is not one criterion by which to rank the methods

- No clear-cut favorite

    - Choice of implementation depends on objective

- General recommendations:

    - JAGS > OpenBUGS

    - Stan has brightest future

Further Research

- How do the samplers handle large data / parameters?

- How extensive is the BUGS language compared to Stan?

- Installation difficulty

References

Betancourt, M. (2017). A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*.

Christensen, R., Johnson, W., Branscum, A., & Hanson, T. E. (2011). Bayesian ideas and data analysis: an introduction for scientists and statisticians. CRC Press

Feng, C. (n.d.). The Markov-chain Monte Carlo Interactive Gallery. Retrieved May 9, 2020, from https://chi-feng.github.io/mcmc-demo/

Fienberg, S. E. (2006). When did Bayesian inference become "Bayesian"?. *Bayesian analysis*, *1*(1), 1-40.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian data analysis*. CRC press.

Green, P. J., Łatuszyński, K., Pereyra, M., & Robert, C. P. (2015). Bayesian computation: a summary of the current state, and samples backwards and forwards. *Statistics and Computing*, *25*(4), 835-862.

[London Machine Learning Meetup] (2018, July 15). *Michael Betancourt: Scalable Bayesian Inference with Hamiltonian Monte Carlo* [Video]. Youtube. https://www.youtube.com/watch?v=jUSZboSq1zg&t=596s

Lunn, D., Jackson, C., Best, N., Thomas, A., & Spiegelhalter, D. (2012). *The BUGS book: A practical introduction to Bayesian analysis*. CRC press.

Neal, R. M. (2011). MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, *2*(11), 2.