Bayesian Analysis with rstan, rstanarm, and brms

Los Angeles R User Group Presentation

Seo-young Silvia Kim
July 10, 2019

California Institute of Technology
GitHub: `https://github.com/sysilviakim`
Twitter: @sysilviakim

Bayesian Data Analysis

What should I believe about θ
in light of the data available for analysis, y?

(Canonical example) Consider two baseball players:

1. One had 6 hits in 10 at-bats
2. One had 0 hits in 5 at-bats

Should we estimate each player's ability as 60% (0%)?

(Slightly more real-life example) Uber/Lyft

1. A driver has 5.0 stars and ride history of 5 rides. Should you cancel?
2. A driver has 4.68 stars and ride history of 1,000 rides. Should you cancel?

Bayes Estimates as Weighted Averages of Priors and Data

Given a binomial likelihood over r successes in n Bernoulli trials, each independent conditional on an unknown success parameter $\theta \in [0, 1]$, i.e.,

$$\mathcal{L}(\theta; r, n) = \binom{n}{r} \theta^r (1 - \theta)^r$$

then the prior density $p(\theta) = \text{Beta}(\alpha, \beta)$ is conjugate with respect to the binomial likelihood—posterior density is

$$p(\theta) = \text{Beta}(\alpha + r, \beta + n - r)$$

Note then the posterior mean of θ is

$$\tfrac{\alpha+r}{\alpha+\beta+n} = \tfrac{n_0}{n_0+n}\theta_0 + \tfrac{n}{n_0+n}\hat{\theta}$$

where θ_0 is the mean of prior density and $\hat{\theta} = \frac{r}{n}$ is the MLE estimate of θ

Bayes Estimates as Weighted Averages of Priors and Data

In this case, posterior mean is a (linearly) weighted average of prior mean and MLE—a convex combination.

- Bayes estimate ”shrinks” the posterior towards the prior
- If $n_0 < n$ (lots of data), $\frac{n_0}{n_0+n}\theta_0 + \frac{n}{n_0+n}\hat{\theta}$ closer to MLE
- If $n_0 > n$ (precise prior), $\frac{n_0}{n_0+n}\theta_0 + \frac{n}{n_0+n}\hat{\theta}$ closer to prior mean

Why Simulate?

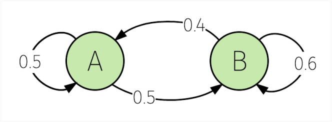
Sometimes, analytical solutions for posterior density not really tractable e.g. not conjugate. Hence a numerical solution:
appeal to the Monte Carlo methods

- Repeatedly random sample from $\Pr(\theta|y)$: a Markov chain
- Draws not independent, but chain's equilibrium distribution = $\Pr(\theta)$ we are interested in) — the target posterior.
- With large sample, can extract information about posterior
- No need to solve integrals

Monte Carlo Principle and Markov Chains

Anything we want to know about a random variable θ can be learned by sampling many times from $f(\theta)$, the density of θ.

- The precision with which we learn about θ is limited only by the number of samples from $f(\theta)$ we are willing to generate, store, and summarize.
- Computational resources matter with MCMC!


$$\begin{tikzcd} A \arrow[loop left, "0.5"] \arrow[r, bend right, "0.5"'] & B \arrow[l, bend right, "0.4"'] \arrow[loop right, "0.6"] \end{tikzcd}$$

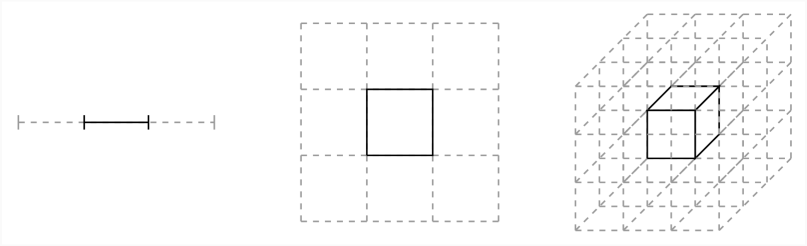
- We can learn about random quantities even when samples from target distribution are serially dependent or autocorrelated!
- As long as each state can be reached by the machine, it can reach equilibrium.
- No matter where you start, number of times you visit each state converges to a stationary state.

Markov Chains

1. Does the chain have a stationary/limiting distribution?
2. If so, is it unique?
3. How long does it take to get there?
4. How can we quantify how close it is getting there?
5. Are summaries of Markov chain trajectory summaries of the stationary distribution?

   One way to ensure computational inefficiency is to waste computational resources evaluating the target density and relevant functions in regions of parameter space that have negligible contribution to the desired expectation.

Target Distribution and Volume Both Matters

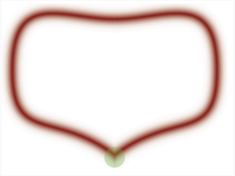


- We are often interested in expectations.
- Only considering regions where the target density is high is problematic
- Volume largest in tails of target distribution, away from the mode.
- Disparity grows with parameter space dimension.

Ideal Markov Chain Exploration

- When a Markov transition preserves the target distribution, it concentrates towards the typical set, no matter where it is applied.
- Three phases (ideally) of Markov chain explorations: converging (warmup), initial exploration, then detailed exploration where estimates are refined.

What usually happens = “pathological behavior”



Metropolis-Hastings and Gibbs Sampling

Two core algorithms commonly used in MCMC.

- Metropolis-Hastings:
    - Has a proposal distribution (stochastic perturbation of initial state)
    - Sample from it
    - Calculate the acceptance probability
    - Accepts with that probability (correction; reject if too far from typical set)
- Gibbs: multivariate setting, sample from conditional probability with one parameter fixed

(But our interest doesn’t lie with them today.)

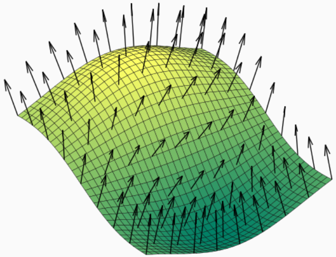
Hamiltonian Monte Carlo (HMC)

Both MH/Gibbs performs poorly when target distribution is complex / dimensions high

- It’s a guess-and-check method
- With many dimensions, only one way to check!
- We need to exploit info about geometry of the typical set
  Hamiltonian Monte Carlo can rapidly explore a target distribution even when dimensions are high

Informing Effective Markov Transitions

With a vector field aligned with the typical set,



… and some more differential geometry, physics metaphor, satellites and orbits and momentums, and Hamiltonian trajectories!

Stan Language

Stan is named in honour of Stanislaw Ulam, pioneer of the Monte Carlo method. Before Stan,

- BUGS e.g. WinBUGS, OpenBUGS
- JAGS

Implemented in C++ but has R- and Python-wrappers (and many others).

It does have higher-level interface called RStanArm package in R, but we will briefly investigate the lower level interfaces and the lower-level R wrapper RStan.

Skeletal Stan Program

```
functions {
   // … function declarations and definitions …
}
data {
   // … declarations …
}
transformed data {
   // … declarations … statements …
}
parameters {
   // … declarations …
}
transformed parameters {
   // … declarations … statements …
}
model {
   // … declarations … statements …
}
generated quantities {
   // … declarations … statements …
}
```

Laplace's Analysis of Sex Ratios at Birth

In 1745-1770 Paris, there were 251,527 boys and 241,945 girls born. Male birth more likely?

Analytically,

- Prior is $\theta = \frac{1}{2}$ (male birth proportion), $Beta(\theta|1, 1)$
- Likelihood is $Binomial(y|N, \theta)$
- Posterior is $Beta(\theta|y + 1, N - y + 1)$, with mean $\frac{1+251527}{1+241945+1+251527} \approx 0.51$

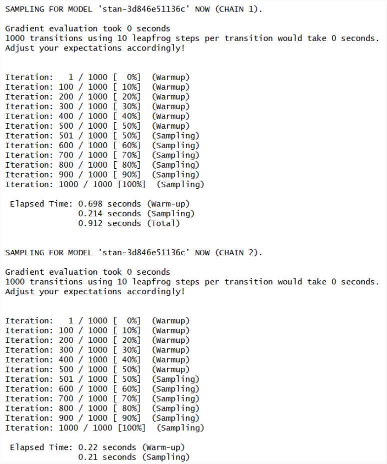
Laplace’s Analysis of Sex Ratios at Birth

```
transformed data {
  int male;
  int female;
  male = 251527;
  female = 241945;
} parameters {
  real<lower=0, upper=1> theta;
}
model {
  male ~ binomial(male + female, theta);
}
generated quantities {
  int<lower=0, upper=1> theta_gt_half;
  theta_gt_half = (theta > 0.5);
}
```

Laplace’s Analysis of Sex Ratios at Birth

> fit <- stan(“laplace.stan”)

Default options: 2,000 iterations (half warmup), 4 chains



```
SAMPLING FOR MODEL 'stan-3d846e51136c' NOW (CHAIN 1).

Gradient evaluation took 0 seconds
1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
Adjust your expectations accordingly!


Iteration:   1 / 1000 [  0%]  (Warmup)
Iteration: 100 / 1000 [ 10%]  (Warmup)
Iteration: 200 / 1000 [ 20%]  (Warmup)
Iteration: 300 / 1000 [ 30%]  (Warmup)
Iteration: 400 / 1000 [ 40%]  (Warmup)
Iteration: 500 / 1000 [ 50%]  (Warmup)
Iteration: 501 / 1000 [ 50%]  (Sampling)
Iteration: 600 / 1000 [ 60%]  (Sampling)
Iteration: 700 / 1000 [ 70%]  (Sampling)
Iteration: 800 / 1000 [ 80%]  (Sampling)
Iteration: 900 / 1000 [ 90%]  (Sampling)
Iteration: 1000 / 1000 [100%]  (Sampling)

 Elapsed Time: 0.698 seconds (Warm-up)
               0.214 seconds (Sampling)
               0.912 seconds (Total)


SAMPLING FOR MODEL 'stan-3d846e51136c' NOW (CHAIN 2).

Gradient evaluation took 0 seconds
1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
Adjust your expectations accordingly!


Iteration:   1 / 1000 [  0%]  (Warmup)
Iteration: 100 / 1000 [ 10%]  (Warmup)
Iteration: 200 / 1000 [ 20%]  (Warmup)
Iteration: 300 / 1000 [ 30%]  (Warmup)
Iteration: 400 / 1000 [ 40%]  (Warmup)
Iteration: 500 / 1000 [ 50%]  (Warmup)
Iteration: 501 / 1000 [ 50%]  (Sampling)
Iteration: 600 / 1000 [ 60%]  (Sampling)
Iteration: 700 / 1000 [ 70%]  (Sampling)
Iteration: 800 / 1000 [ 80%]  (Sampling)
Iteration: 900 / 1000 [ 90%]  (Sampling)
Iteration: 1000 / 1000 [100%]  (Sampling)

 Elapsed Time: 0.22 seconds (Warm-up)
               0.21 seconds (Sampling)
```

Rubin (1981)'s Eight Schools Example

Data from 8 schools to analyze the effect of SAT coaching. No prior reason to believe any program was

- more effective than others
- more similar than others

	school	y	sigma
1	A	28	15
2	B	8	10
3	C	-3	16
4	D	7	11
5	E	-1	9
6	F	1	11
7	G	18	10
8	H	12	18

Rubin (1981)'s Eight Schools Example

```
data {
  int<lower=0> J;   vector[J] y;
  vector<lower=0>[J] sigma;
}
parameters {
  real mu;   real<lower=0> tau;
  vector[J] theta_tilde;
} transformed parameters {
  vector[J] theta;
  theta = mu + tau * theta_tilde;
}
model {
  mu ~ normal(0, 5);   tau ~ cauchy(0, 5);
  theta_tilde ~ normal(0, 1);
  y ~ normal(theta, sigma);
}
generated quantities {
  vector[J] log_lik;   vector[J] y_tilde;
  for(j in 1:J){
    log_lik[j] = normal_lpdf(y[j] | theta[j], sigma[j]);
    y_tilde[j] = normal_rng(theta[j], sigma[j]);
  }
}
```

rstan

1. Write lower-level Stan code.
2. Run `rstan`.
3. Check for diagnostics red flags and posterior predictive checks (I'll get to this soon)

Some example code:

```r
senate_stanfit <- stan(
  "stan/senate_full.stan",
  data = c(X, list(Y = senate_obs$dv_unit)),
  init_r = 0.5, seed = 100,
  control = list(adapt_delta = 0.95)
)
```

rstanarm

Do I have to code lower-level Stan codes? :(

If you are doing standard regressions/hierachical models, no! $\Rightarrow$ use `rstanarm`!

- Goal of package: make Bayesian estimation *routine*
- Syntax similar to previous `lme4` package

Same routine:

1. Specify a posterior distribution
2. Draw from the posterior distribution
3. Criticize the model
4. Analyze manipulations of predictors

Some example code:

```r
fit_partialpool <- stan_glmer(
  cbind(Hits, AB - Hits) ~ (1 | Player), data = bball, family = binomial("logit"),
  prior_intercept = wi_prior, seed = 101,
  QR = TRUE, chains = 4, cores = 4, iter = 2000
)
```

brms

You could also use `brms` by Paul Bürkner who recently joined the Stan team
Similar to `rstanarm`, but with some improvements

> *The rstanarm package is similar to brms in that it also allows to fit regression models using Stan for the backend estimation. Contrary to brms, rstanarm comes with precompiled code to save the compilation time (and the need for a C++ compiler) when fitting a model. However, as brms generates its Stan code on the fly, it offers much more flexibility in model specification than rstanarm. Also, multilevel models are currently fitted a bit more efficiently in brms.*

Currently my favorite. Some example code:

```r
fit1 <- brm(count ~ zAge + zBase * Trt + (1|patient),
            data = epilepsy, family = poisson())
```

More packages

There's also Richard McElreath's `rethinking`! (It is a vibrant community.)
Pedagogically useful as all lectures are online (Statistical Rethinking Winter 2019)

Also check out `bayesplot` which provides integration of Bayesian model results and ggplot.

Diagnostics

Whichever package you use, every time you run a Stan model, several things to check: Launching a `shinystan` object will generate all these.

1. Traceplots
2. Rhats
3. Effective sample size (Neffs)
4. Autocorrelation
5. Posterior predictive checks

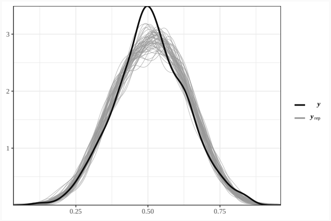
```r
schools <- stan_demo("eight_schools")
check_hmc_diagnostics(schools)
## check_divergences(schools)
## check_treedepth(schools)
## check_energy(schools)
```

You will be tempted—as I have—to compromise when the model is giving you divergence/treedepth/energy warnings. **Don't. Not ever.** Analyze the results only after model passes all checks.

Posterior Predictive Checks

Is the model okay if it passes all HMC checks? Nope.
PPC = A validation step.

- Is the posterior distribution approximating the underlying parameters?
- Does the true value lie within a reasonable interval of predicted value, with the estimated model?



Extremely important, always should be checked after a model run

Miscellaneous

Not discussed today:

- Centered vs. non-centered parameterization (the issue of units!)
- Numerical overflow (use logs)
- Think again and again about the data generating process!

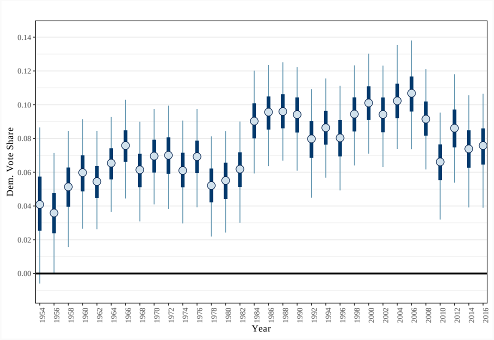
Some tips:

- Use Atom text editor with langauage-stan package installed. It highlights syntaxes (Notepad++ has none.)
- Post questions on the Stan Forum. Lots of updates going on.
- Check out Betancourt's case studies e.g. Towards A Principled Bayesian Workflow

Stan and Social Science Research

I use Stan to analyze political behavior and election results, especially estimating parameters of small legislatures or subset of voters. Examples:

- Estimating ballot returns by county (handful of counties in state of interest)
- Senate incumbency advantage (handful of elections per year)



Concluding Remarks

There are lots of applications of Stan besides a simple linear hierarchical model e.g. state space models, spatial smoothing, splines, item response theory, ...

Checkout YouTube videos of StanCon and the latest case studies!

Sources

1. Stan Language Manual, Version 2.19.0.
2. Stan Wiki on GitHub.
3. Betancourt. (2017). A Conceptual Introduction to Hamiltonian Monte Carlo.
4. Betancourt et al. (2017). The geometric foundations of Hamiltonian Monte Carlo.
5. Betancourt & Girolami. (2013). Hamiltonian Monte Carlo for Hierarchical Models.
6. Carpenter et al. (2016). Stan: A probabilistic programming language.
7. Carpenter, Gabry, and Goodrich. (2017). Hierarchical Partial Pooling for Repeated Binary Trials.
8. Hoffman, & Gelman. (2014). The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo.
9. Neal. (2011). MCMC using Hamiltonian dynamics. (In Handbook of Markov Chain Monte Carlo)