Bayesian Data Analysis

Class 3: Stan

Daniel Lee

http://mc-stan.org

What is Stan

- Language: statistical modeling language
- Implementation: MCMC, approx. Bayes, optimization
- Interfaces: R, Python, command line, Matlab, Julia, Stata

Today

- **RStan / R**: how to interact with Stan
- **Language:** what is a Stan program? how to use other blocks
- **NUTS**: no details, just usage

Practical Goals

- Run Stan programs
- Code whatever model you want
- Get out of trouble
- How to get help after today's class
- ...

Download materials:

https://github.com/syclik/class3

RStan: running Stan programs

Looks like:

```
data <- read_rdump('data_file.data.R')
fit <- stan("model_file.stan", data = data)
print(fit)
plot(fit)
```

1. Run normal.stan with data from normal1.data.R: $\alpha = 2$, $\beta = [6]$, $\sigma = 1$
2. Run normal.stan with data from normal2.data.R: $\alpha = -100$, $\beta = [1, 3]$, $\sigma = 10$ (don't need to recompile. How?) What's going on? Fixes?
3. Generate new simulated data and run. (use `stan_rdump` or save to list) (Types are important in Stan!)

RStan: more options

Looks like:

```
?stan
```

1. set: `chains`, `iter`
2. use: `control`. Common changes:
   `adapt_delta, stepsize, max_treedepth`

Compile time errors

Run the Stan compiler:

```
stanc("model_file.stan")
```

1. compiletime-error1.stan. Fix it.
2. compiletime-error2.stan. Fix it.
3. compiletime-error3.stan. What is it telling you? (See the manual)
4. If you have time, generate data and run them.

Run time errors

Run the Stan compiler:

```
stanc("model_file.stan")
```

1. runtime-error1.stan with runtime-error1.data.R. What's wrong?
2. runtime-error2.stan with runtime-error2.data.R. What's missing in data?
3. runtime-error3.stan. How to fix this one? (use print() in the Stan language)

Converting data

- Data frames don't convert directly
- Static types / shapes are important
- In Stan: matrix of 1 column is different than a vector
- Questions?

Language

Stan program

- Recall: $p(\theta|x) = \frac{p(\theta,x)}{p(x)} \propto p(\theta,x)$
- Alternatively: $p(\theta|x) = p(\theta,x) \times f(x)$

Stan program

- Define x
- Define θ
- Define $\log p(\theta, x)$
- Stan provides samples from $p(\theta|x)$

Stan program: subtleties

- imperative
- log space for numeric stability $\log p(\theta, x)$
- arbitrary joint distribution.
- not necessarily factorized as: $\log p(\theta) + \log(x|\theta)$.
- not limited to graphical models
- no easy way to swap θ and x, though it seems natural

increment_log_prob()

Rewrite normal.stan

Use increment_log_prob in the model block (save as a new file):

```
...
model {
  increment_log_prob(...);
}
```

1. Check inference against old runs
2. When can you drop terms?

Another example?

1. Truncated Poisson example? (time dependent)

Sampling statements

There is nothing special about them

```
y ~ normal(mu, sigma);
```

is (almost) the same as

```
increment_log_prob(normal_log(y, mu, sigma));
```

Review Homework 1 and 2

Stan program blocks

data block

- Declare x, but don't define it. Can't define it.
- Data does not change once a program is running.
- Try: `data { int N; N <- 1; } model {}`
- Data block is validated. Recall runtime-error1.stan
- Excuted once and only once.

transformed data block

- Declare AND define variables of the form $z = f(x)$.
- Transformed data block validated after all the statements in the block.
- Executed once and only once.

parameters block

- Declare θ, but don't define it. Can't define it.
- Constraints are enforced by construction.

transformed parameters block

- Declare AND define variables of the form $z = f(x, \theta)$
- Transformed parameters validated after all the statements in the block.
- Executed each leapfrog step = multiple times per iteration

model block

- Defines the log joint distribution of θ and x.
- If you use a transformed parameter on the left side, need increment the log probability by the absolute log determinant of the Jacobian.
- Executed each leapfrog step = multiple times per iteration

generated quanitities block

- Quantities of interest.
- Posterior predictive check.
- Can also generate random numbers here.
- Executed once per iteration.

functions block

- Before the data block.
- Define and declare functions to be used in following blocks.
- Sampling works (call something _log and you can use a sampling statement)
- Static sizes are left off

Additional topics

- Numeric stability
- Static types: vector vs. row_vector vs. matrix vs. array
- Workflow
- Conjugacy doesn't matter

Help?

Where to get help

- Users group (1300+ and counting):
  https://groups.google.com/forum/#!forum/stan-users
- Stan User's Guide and Reference Manual:
  http://mc-stan.org/documentation/
- Website:
  http://mc-stan.org
- @mcmc_stan