Stan interface

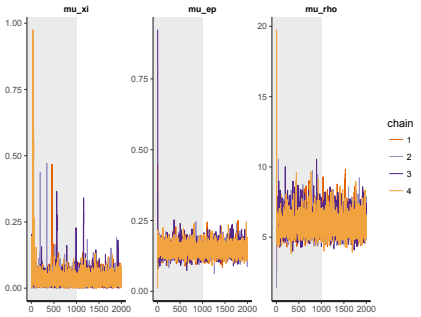
Week #8

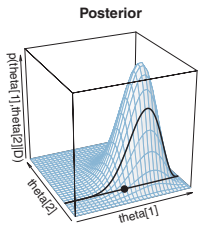
Woo-Young Ahn
Department of Psychology
Seoul National University
ccs-lab.github.io

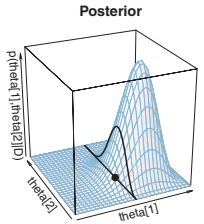
Recap of Week 7

- *Markov Chain Monte Carlo*
  - *What is a Markov Chain?*
  - *What is a Monte Carlos process?*
- *Metropolis algorithm*
  - *How does it work? E.g., travelling politician*
  - *What are pros and cons?*
- *Gibbs sampling*
  - *How does it work?*
  - *What are pros and cons?*
- *Things to know when using MCMC*
  - *Traceplots ("fat hairy caterpillar")*
  - *Rhat*
  - *Why multiple chains?*
  - *Burn-in (warmup) samples*
  - *Autocorrelation, effective sample size, and thinning*

$$p_{\text{move}} = \min\left(\frac{P(\theta_{\text{proposed}})}{P(\theta_{\text{current}})}, 1\right)$$

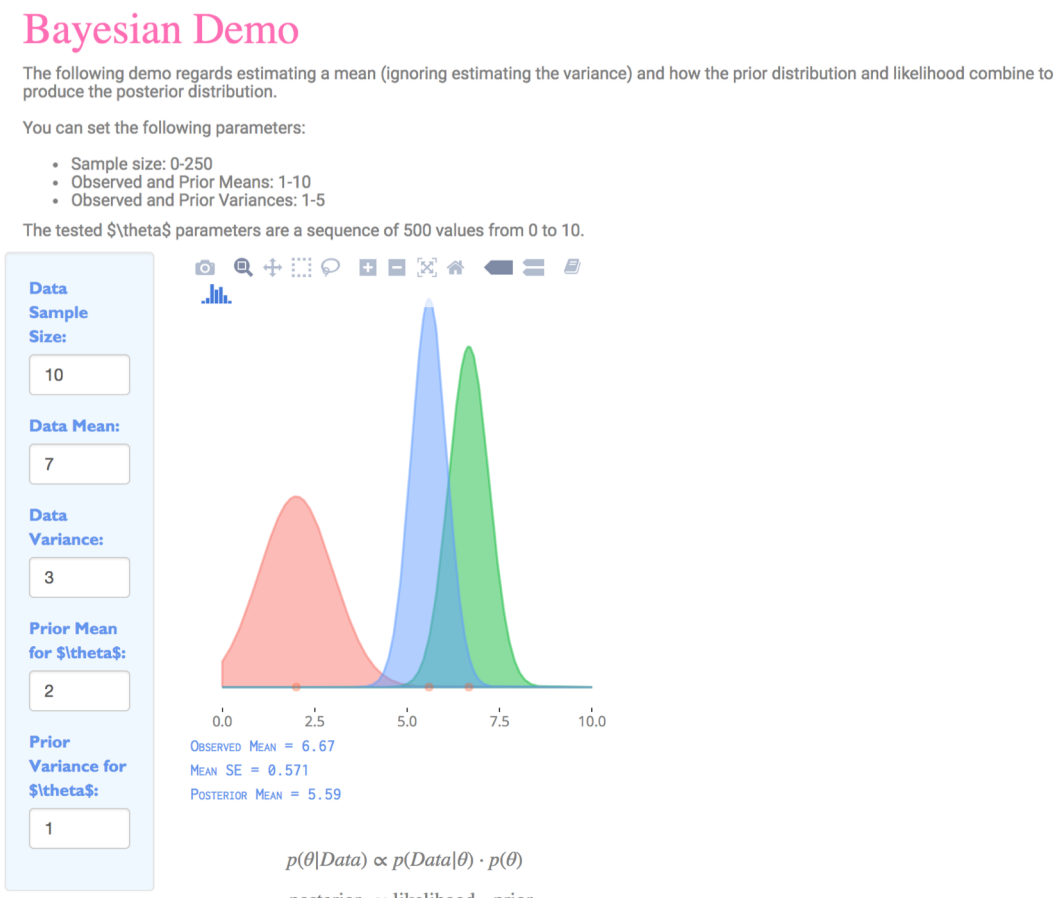






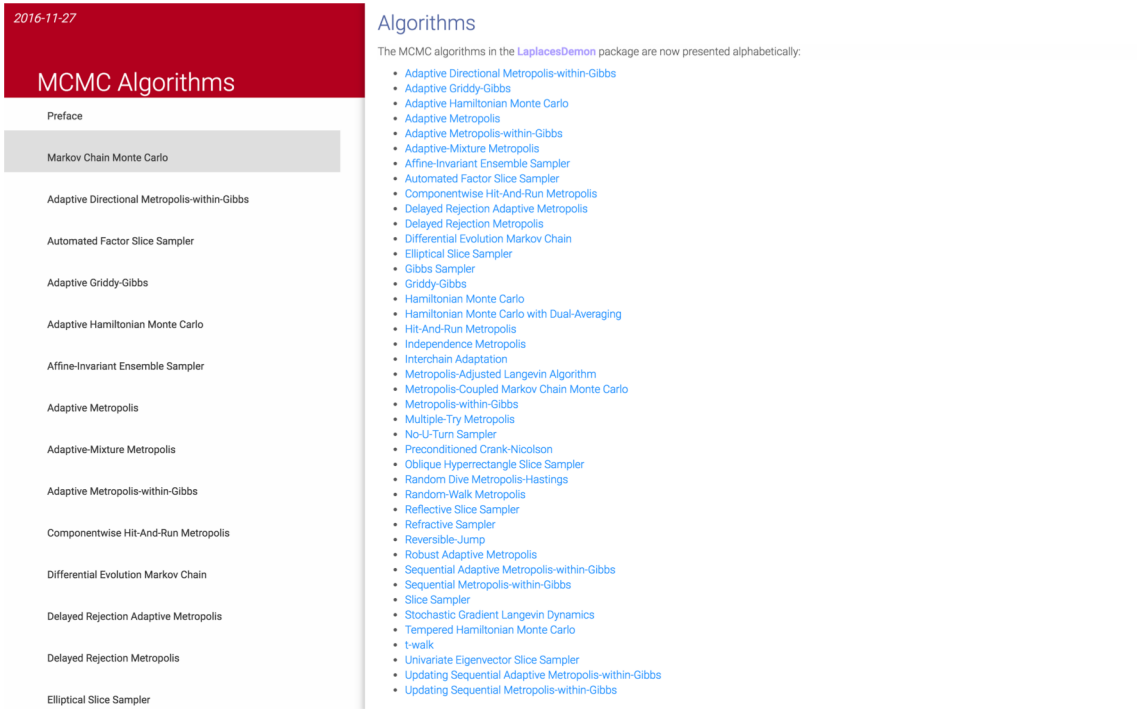
Bayesian Demo

- https://micl.shinyapps.io/prior2post/



More MCMC algorithms

- *http://m-clark.github.io/docs/ld_mcmc*
- *Dozens of MCMC algorithms covered*



2016-11-27

MCMC Algorithms

Preface

Markov Chain Monte Carlo

Adaptive Directional Metropolis-within-Gibbs

Automated Factor Slice Sampler

Adaptive Griddy-Gibbs

Adaptive Hamiltonian Monte Carlo

Affine-Invariant Ensemble Sampler

Adaptive Metropolis

Adaptive-Mixture Metropolis

Adaptive Metropolis-within-Gibbs

Componentwise Hit-And-Run Metropolis

Differential Evolution Markov Chain

Delayed Rejection Adaptive Metropolis

Delayed Rejection Metropolis

Elliptical Slice Sampler

Algorithms

The MCMC algorithms in the LaplacesDemon package are now presented alphabetically:

- Adaptive Directional Metropolis-within-Gibbs
- Adaptive Griddy-Gibbs
- Adaptive Hamiltonian Monte Carlo
- Adaptive Metropolis
- Adaptive Metropolis-within-Gibbs
- Adaptive-Mixture Metropolis
- Affine-Invariant Ensemble Sampler
- Automated Factor Slice Sampler
- Componentwise Hit-And-Run Metropolis
- Delayed Rejection Adaptive Metropolis
- Delayed Rejection Metropolis
- Differential Evolution Markov Chain
- Elliptical Slice Sampler
- Gibbs Sampler
- Griddy-Gibbs
- Hamiltonian Monte Carlo
- Hamiltonian Monte Carlo with Dual-Averaging
- Hit-And-Run Metropolis
- Independence Metropolis
- Interchain Adaptation
- Metropolis-Adjusted Langevin Algorithm
- Metropolis-Coupled Markov Chain Monte Carlo
- Metropolis-within-Gibbs
- Multiple-Try Metropolis
- No-U-Turn Sampler
- Preconditioned Crank-Nicolson
- Oblique Hyperrectangle Slice Sampler
- Random Dive Metropolis-Hastings
- Random-Walk Metropolis
- Reflective Slice Sampler
- Refractive Sampler
- Reversible-Jump
- Robust Adaptive Metropolis
- Sequential Adaptive Metropolis-within-Gibbs
- Sequential Metropolis-within-Gibbs
- Slice Sampler
- Stochastic Gradient Langevin Dynamics
- Tempered Hamiltonian Monte Carlo
- t-walk
- Univariate Eigenvector Slice Sampler
- Updating Sequential Adaptive Metropolis-within-Gibbs
- Updating Sequential Metropolis-within-Gibbs

Installation (rstan)

- *For MAC users:*
  1. *Make sure you have Xcode installed: https://developer.apple.com/xcode/*

- *For Windows users:*
  1. *Download R tools: https://cran.r-project.org/bin/windows/Rtools/*
     - *For more details, see: https://github.com/stan-dev/rstan/wiki/Installing-RStan-on-Windows*

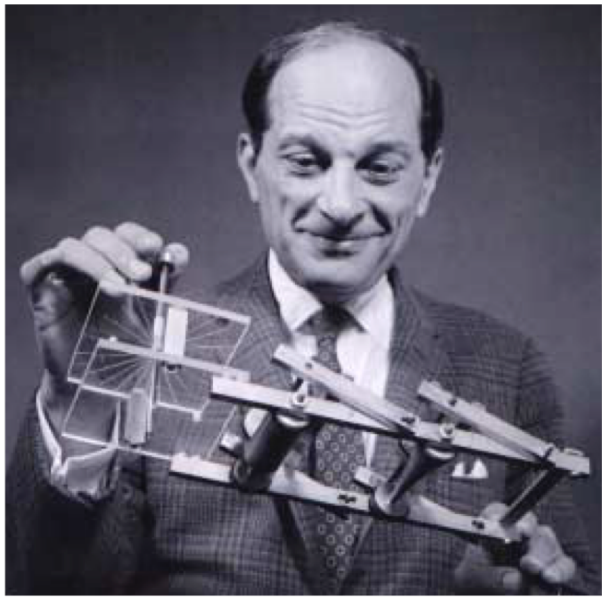
Run the following command through the R console:

```
install.packages("rstan", dependencies = T)
```

Stan

Stanislaw Ulam, namesake of Stan and co-inventor of Monte Carlo methods (Metropolis and Ulam, 1949), shown here holding the Fermiac, Enrico Fermi's physical Monte Carlo simulator for neutron diffusion.

Image from (Giesler, 2000).



Stan

- *A modeling language (software) for Bayesian data analysis*
- *MCMC is actually done in C++*
- *Stan interfaces in R, Python, Matlab, Julia, Stata, Mathematica, and Scala*
- *Hamiltonian Monte Carlo (HMC), which works well even for complicated models and models with highly correlated parameters*
- *Also supports variational Bayes*
- *http://mc-stan.org/*
- *Maintained by Andrew Gelman and his team at Columbia University*

Stan

- *Manual (version 2.17.0)*
  - *https://github.com/stan-dev/stan/releases/download/v2.17.0/stan-reference-2.17.0.pdf*
  - *Over 600 pages long…*

- *Stan forum: http://discourse.mc-stan.org/*

- *Similar to BUGS (and JAGS)*
  - *See Appendix B of Stan manual*

Grammar in Stan

- *Use // and /* */ for comments*
- *Every assignment statement must be followed by a semicolon (;)*

  e.g., n = n + 1;

- *Use = instead of <- for assignment*
- *…*

Structure of Stan model specification

- *6 model blocks*

```
data {
... read in external data ...
}
transformed data {
... pre-processing of data ...
}
parameters {
... parameters to be sampled by HMC ...
}
transformed parameters {
... pre-processing of parameters ...
}
model {
... statistical/cognitive model ...
}
generated quantities {
... post-processing of the model ...
}
```

Optional

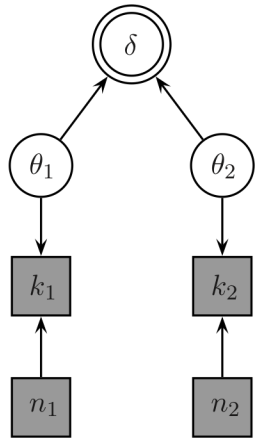
From Ahn et al (2017) Computational Psychiatry

Group discussion

- *Discuss what should be in each block in Stan.*
- *Explain to each other what's unique and essential part of each block.*
- *Look up Stan manual if needed.*

  https://github.com/stan-dev/stan/releases/download/v2.17.0/stan-reference-2.17.0.pdf

Graphical model



$$
\begin{aligned}
k_1 &\sim \text{Binomial}(\theta_1, n_1) \\
k_2 &\sim \text{Binomial}(\theta_2, n_2) \\
\theta_1 &\sim \text{Beta}(1, 1) \\
\theta_2 &\sim \text{Beta}(1, 1) \\
\delta &\leftarrow \theta_1 - \theta_2
\end{aligned}
$$

Fig. 3.3 Graphical model for inferring the difference, $\delta = \theta_1 - \theta_2$, in the rates of two binary processes.

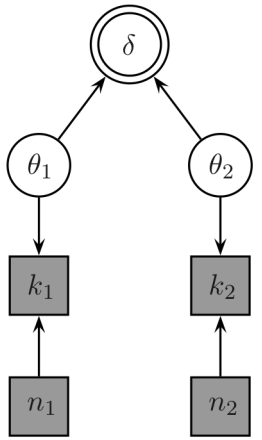
Variables without shading: unobserved
Variables with shading: observed

Continuous variables: circular nodes
Discrete variables: square nodes

Single vs double bordered: probabilistic vs deterministic

1. Data block in Stan

- *We tell Stan which data should be loaded*
- *Data types*
  - *Real (*`real`*)*
  - *Integer (*`int`*)*
  - *Vector and matrix types*
    - `vector` *for column vectors*
    - `row_vector` *for row vectors*
    - `matrix` *for matrices*



- *Constrained data types*
  - *Data types may be given lower and upper bounds*
  - *e.g.,* `vector<lower = -1, upper = 1>[3] rho;`

Conditional operators (page 62 of Stan manual)

Op.	*Prec.*	*Assoc.*	*Placement*	*Description*
? :	10	right	ternary infix	conditional
||	9	left	binary infix	logical or
&&	8	left	binary infix	logical and
==	7	left	binary infix	equality
!=	7	left	binary infix	inequality
<	6	left	binary infix	less than
<=	6	left	binary infix	less than or equal
>	6	left	binary infix	greater than
>=	6	left	binary infix	greater than or equal
+	5	left	binary infix	addition
-	5	left	binary infix	subtraction
*	4	left	binary infix	multiplication
/	4	left	binary infix	(right) division
%	4	left	binary infix	modulus
\	3	left	binary infix	left division
.*	2	left	binary infix	elementwise multiplication
./	2	left	binary infix	elementwise division
!	1	n/a	unary prefix	logical negation
-	1	n/a	unary prefix	negation
+	1	n/a	unary prefix	promotion (no-op in Stan)
^	0.5	right	binary infix	exponentiation
'	0	n/a	unary postfix	transposition
()	0	n/a	prefix, wrap	function application
[]	0	left	prefix, wrap	array, matrix indexing

Indexing in Stan (page 65 of Stan manual)

index type	*example*	*value*
integer	`a[11]`	value of `a` at index 11
integer array	`a[ii]`	`a[ii[1]], ..., a[ii[K]]`
lower bound	`a[3:]`	`a[3], ..., a[N]`
upper bound	`a[:5]`	`a[1], ..., a[5]`
range	`a[2:7]`	`a[2], ..., a[7]`
all	`a[:]`	`a[1], ..., a[N]`
all	`a[]`	`a[1], ..., a[N]`

Figure 4.2: *Types of indexes and examples with one-dimensional containers of size* `N` *and an integer array* `ii` *of type* `int[]` *size* `K`.

2. Transformed Data block in Stan

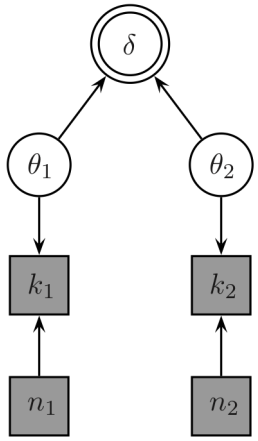
- *Sometimes we need to transform data loaded in the Data block before we feed the (transformed) data into our model*
- *This is optional. Often easier to do it in R first.*
- *E.g.,*

```
transformed data {          // Transformed data block
   vector[N] logX;
   logX = log(X);
}
```

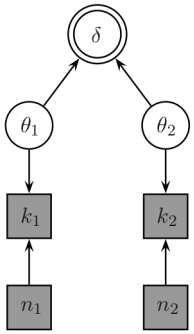
3. Parameters & transformed parameters blocks in Stan

- *We define parameters to be sampled*

- *Transformed parameters → parameters that are dependent on the parameters*



4. Model block



$$
\begin{aligned}
k_1 &\sim \text{Binomial}(\theta_1, n_1) \\
k_2 &\sim \text{Binomial}(\theta_2, n_2) \\
\theta_1 &\sim \text{Beta}(1, 1) \\
\theta_2 &\sim \text{Beta}(1, 1) \\
\delta &\leftarrow \theta_1 - \theta_2
\end{aligned}
$$

- *Specify priors and likelihood*
- *Declare any variables necessary*
- *Look up Stan manual and find functions/commands for each prior and transformation*
- *You can use for loop, if statement, etc.*
  - *(not just for model but for other blocks as well)*

5. Generated quantities block

- *We can generate some sorts of quantities based on already sampled variables*
- *Posterior predictive checks*
- *Log probability for a model*

Distributions and functions used in Stan

- *We need to look up Stan manual*

- *Distributions → part VIII (pages 502~)*

54.1. Normal Distribution

Probability Density Function

If $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}^+$, then for $y \in \mathbb{R}$,

$$\text{Normal}(y|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2\right).$$

Sampling Statement

`y ~ normal(mu, sigma);`

Increment log probability with `normal_lpdf(y | mu, sigma)`, dropping constant additive terms; Section 5.3 explains sampling statements.

Distributions and functions used in Stan

- *Built-in functions (pages 427-)*

R **inv_logit**(T *x*)

Returns the (elementwise) logistic sigmoid function applied to *x*,

$$\texttt{inv\_logit}(x) = \frac{1}{1 + \exp(-x)}$$

for any argument type T; see Section 41.1 for details including return type R.

Example
`ra_prospect_stan_singleSubj`

- *Let's look at the code together!*