STK 880: Bayesian Statistical Modelling and Computing using R, JAGS and (Stan)

The basics of Stan

Dr. Rianne Jacobs

University of Groningen, The Netherlands

14-18 September 2018

https://github.com/Stattie/JagsTutorial.git

Simple linear regression

Example: Osteoporosis study (Lesaffre, Lawson, 2012

- TBBMC (in kg): total body bone mineral content - marker for osteoporosis
- BMI (in kg/m^2): body-mass index
- Simple linear regression: $TBBMC_i = \beta_0 + \beta_1 BMI_i + \epsilon_i$
- Model

$$y_i \sim \mathrm{N}(\beta_0 + \beta_1 x_i, \sigma^2)$$

The Stan program

```
data {
   int N; //the number of observations
   vector[N] y; //response
   vector[N] x; //covariate
}
parameters {
   vector[2] beta; //the regression parameters
   real sigma; //the standard deviation
}
model {
   beta ~ normal(0, 1000); // prior, normal(mean, sd)
   sigma ~ cauchy(0, 5); // half-cauchy implicitly
   y ~ normal(beta[1] + beta[2] * x, sigma);
}
```

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

Stan programming blocks

- Stan program contains blocks
- Blocks contain variable declarations and (sometimes) statements
- All of the blocks are optional
- Stan program blocks that occur must occur in the correct order
- Within each block, both declarations and statements are optional
- Declarations come before the statements

Data types

- Primitive Types
  - `real` for continuous values
  - `int` for integer values
- Vector and Matrix Types
  - `vector` for column vectors
  - `row_vector` for row vectors
  - `matrix` for matrices
- Array Types, can contain any type

Declarations

- Integer and real: type <constraint> name
  - `int <lower=0,upper=1> N;`
  - `real<lower=-1,upper=1> sig;`
- Vectors: type <constraint> [length] name
  - `vector<lower=0>[3] u;`
- Matrices: type <constraint> [dimension] name
  - `matrix<upper=0>[3, 4] A;`
- Arrays: type of entries <constraint> name dimension
  - `real a[3, 4];`
  - `rvector[7] mu[3];`
  - `matrix[7, 2] mu[15, 12]`

Some notes

- Variable names are case sensitive
- Fullstops not allowed in variable names
- May declare and assign in one line: `int p = 5;`
- Distributions: e.g. `normal(mu,sigma)`
- Comments indicated with //
- Stan program to be saved as .stan

Working with R

- Call Stan

```
> library(rstan)
> options(mc.cores = parallel::detectCores())
> rstan_options(auto_write = TRUE)
```

- Prepare the data

```
> data1a = list(N = N, x = x, y=y)
```

- Provide initial values (optional)
- Run Stan

```
> model1 = stan(file="model1a.stan", data = data1a,
+ iter=1000, chains = 4, pars = c('beta', 'sigma'))
```

- Process output

Working with R

- Convergence diagnosis using plots

```
> stan_trace(model1, pars = 'beta', inc_warmup = T)
> stan_ac(model1)
```

- Formal convergence diagnosis given by $\hat{R}$ and effective sample size

```
> summary(model1)$summary
> stan_rhat(model1)
```

- Summary measures

```
> summary(model1)$summary
> stan_plot(model1)
```