Probabilistic programming languages for Bayesian inference in R

Måns Magnusson

StiMa, Linköping University

Probabilistic programming languages

- BUGS (**B**ayesian inference **U**sing **G**ibbs **S**ampling)
- JAGS (**J**ust **A**nother **G**ibbs **S**ampler)
- Stan (**Stan**islaw Ulam)

When not to use

- Big Data
- Big Models

Why JAGS?

- Easy to use (BUGS type)
- Robust (version 4.x.x)
- Wrappers for Python and R
- No C++ compiler needed
- Good starting point

JAGS website

Why Stan?

- Easy to use (BUGS type)
- Effective samplers/fast
- Integrates nice with RStudio
- Wrappers from Python, R, Matlab, Stata, Julia
- Automatic/Black box variational inference
- (Really) good documentation

mc-stan.org

The examples

- Bernoulli-Beta (Coin flips, conjugate)
  - Model:

$$y_i \sim \text{Bern}(p)$$

  - Prior:

$$p \sim \text{Beta}(a, b)$$

- Poisson regression (Number of roaches caught in buildings, non-conjugate)
  - Model:

$$y_i \sim \textit{Poisson}(\log(\text{exposure}) + \beta_1 + \beta_2 \cdot \text{treatment} + \beta_3 \cdot \text{senior})$$

  - Prior:

$$\beta \sim \text{Norm}(0, 1000)$$

Stan

- Needs C++ compiler - good installation instructions here
- Six parts in a Stan model:
  - `data`
  - `transformed data`
  - `parameters`
  - `transformed parameters`
  - `model*`
  - `generated quantities`

Model in Stan: data

- Read in data (for example from R or Python) once
  - Only variable declarations
  - A lot of different data types
    - `int`, `real`, `vector`, arrays, `matrix` and more Stan specific (?) data types as `cholesky_factor_cov` and `unit_vector`

Example of the data block

```
data {
   int<lower=0> N; # The number of observations
   int<lower=0> y;
   vector[N] exposure2;
   vector[N] senior;
   vector[N] treatment;
}
```

Model in Stan: `transformed data`

- Variable declarations **and** statements (done once)
  - See chapter V in the documentation for all functions that can be used.

Example of the `transformed data` block

```
transformed data {
   vector[N] log_expo;
   log_expo <- log(exposure2);
}
```

Model in Stan: `parameters`

- Parameters (that should be sampled)
  - Parameter declarations only.

Example of the `parameters` block

```
parameters {
   vector[3] beta;
}
```

Model in Stan: `transformed parameters`

- Parameter declarations **and** statements
  - The transformations is done in each sampling step

Example of the `transformed parameters` block (not example model)

```
transformed parameters {
   real<lower=0> sigma;
   sigma <- 1.0 / sqrt(tau);
}
```

Model in Stan: `model`

- Declare the priors and data with sampling statements ~
  - Distributions can be found in chapter VI and VII in the documentation

Example of the `model` block

```
model {
   // Priors
   beta ~ normal(0.0, 1000.0);
   # Model
   y ~ poisson_log(log_expo + beta[1] + beta[2] * treatment +
beta[3]*senior);
}
```

Model in Stan: `generated quantities`

- Computations after the sampling has been done, used for
- Is used for:
  - model checking
  - predictive distributions for new data
  - applying full Bayesian decision theory
  - transforming parameters for reporting, etc

Example of the `generated quantities` block (not example model)

```
generated quantities {
   real my_weight_pred;
   my_weight_pred <- alpha + beta * MySHeight +
normal_rng(0,sigma);
}
```

How to specify a model in JAGS

- One parts in a JAGS model:
  - `model*`

Example of the `model` block

```
model {
   # Model
   for( i in 1:N){
      y[i] ~ dpois(lambda[i])
      log(lambda[i]) <- log(exposure2[i]) + beta1 + beta2 *
treatment[i] + beta3 * senior[i])
   # Priors
   beta1 ~ dnorm(0.0, 0.0001)
   beta2 ~ dnorm(0.0, 0.0001)
   beta3 ~ dnorm(0.0, 0.0001)
}
```

demo

Demonstration