INTRODUCTION TO STAN

BAYESIAN STATISTICS FOR ECOLOGISTS

IGB 18. TO 26. NOVEMBER 2019

WHAT IS STAN?

- General purpose modelling and scientific programming language
- Imperative (like R, unlike BUGS) and probabilistic language
- Fully Bayesian inference using Hamiltonian Monte Carlo
  - Your model is compiled to a C++ program that is executed from within R
  - HMC converges faster and is more stable than Gibbs in many cases
  - Lower autocorrelation than Gibbs

WHY STAN?

- In many cases faster than Gibbs, even Gibbs in C++
- Much of the tuning, warmup, care and feeding is done automatically
- Model specification is very concise

VARIABLES

- Stan is **strongly typed**: all variables must be declared and used in their proper scope
  - real, int for single numbers
  - vector[i], matrix[i, j] for real-valued arrays to be used with linear algebra
  - arrays (for all other sequences)
- Stan supports variable constraints

```
real my_real;
int my_int;

vector [10] my_vector;
matrix [10, 20] my_matrix;

int my_int_array [10];
vector [10] array_of_vectors [5, 2];

real <lower=0, upper=1> constrained_real;
```

STRUCTURE OF A PROGRAM

- A Stan program consists of a series of optional **blocks** delimited by curly braces
- Order matters, and variables are visible from the top down (**model** can see variables declared in **data**, but not vice-versa!)
- Variable declarations must come first in each block
- **data**: only declares the data variables that will be passed to Stan
- **parameters**: only declares parameters that will be used in the model
- **model**: defines the log probability of the model

```
functions {}

data {}

transformed data {}

parameters {}

transformed parameters {}

model {}

generated quantities {}
```

STAN PRACTICE—TREE MORTALITY

- Often helpful to work backwards, start with the model definition
- `died` is a data vector giving the number of trees that died
- `ntrees` is the vector of the total number of trees
- `theta` is the parameter to estimate
- To do:
    - Fill in the missing blocks
    - Add prior
    - Run the model in R

STAN PRACTICE—TREE MORTALITY

- Often helpful to work backwards, start with the model definition
- `died` is a data vector giving the number of trees that died
- `ntrees` is the vector of the total number of trees
- `theta` is the parameter to estimate
- To do:
    - Fill in the missing blocks
    - Add prior
    - Run the model in R

```
data {
    // fill in data declarations
}
parameters {
    // fill in parameters
}
model {
    died ~ binomial(ntrees, theta);
}
```

STAN PRACTICE—TREE MORTALITY

- It is helpful to include the number of data points as a data variable
- All data and parameters must be declared
- **Data constraints** serve as error checking, in this case checking for negative numbers
- **Parameter constraints** prevent the sampler from selecting impossible values
- All **unobserved** objects (e.g., theta) should appear on the left side of a statement:
  either = or ~
- Check carefully! It's not an error to forget this step

STAN PRACTICE—TREE MORTALITY

- It is helpful to include the number of data points as a data variable
- All data and parameters must be declared
- **Data constraints** serve as error checking, in this case checking for negative numbers
- **Parameter constraints** prevent the sampler from selecting impossible values
- All **unobserved** objects (e.g., theta) should appear on the left side of a statement:
  either = or ~
- Check carefully! It's not an error to forget this step

```
data {
    int<lower=0> n; // number of data points
    int<lower=0> died[n];
    int<lower=0> ntrees[n];
}
parameters {
    real<lower=0, upper=1> theta;
}
model {
    died ~ binomial(ntrees, theta);
    theta ~ ?????
}
```

STAN PRACTICE—TREE MORTALITY

- Running the model is simply a matter of pointing the `stan()` function at the .stan file you created
- Examine the summary of the model. Is the parameter estimate different from the MLE?
- You can use bayesplot to produce some useful diagnostics

STAN PRACTICE—TREE MORTALITY

- Running the model is simply a matter of pointing the `stan()` function at the .stan file you created
- Examine the summary of the model. Is the parameter estimate different from the MLE?
- You can use bayesplot to produce some useful diagnostics

```
library(rstan)
library(data.table)
trees <- readRDS("data/trees.rds")
dat <- trees[grepl("TSU-CAN", species)
             & year == 2005 & n > 0]

model <- stan("code/3_ii_trees.stan",
              data = list(n = nrow(dat),
              died = dat$died, ntrees = dat$n))
```