Stan for the people

Two day introductory workshop
on Bayesian modeling

McGill University
January 26th 2019

mc-stan.org

IV

Conversational Stan

Parallel chains

- Each chain is completely independent and can be run on a different core.

Parallel chains

```
fit <- stan(file = "model/hierarchy_linear.stan",
            data = data,
            init = init,
            cores = 4)
```

Parallel chains

```
fit <- stan(file = "model/hierarchy_linear.stan",
            data = data,
            init = init,
            cores = min(4, parallel::detectCores())
```

Variable types

- standard: real, int, vector, matrix
- arrays: real[], int[], vector[], matrix[]
- specialized: simplex, cov_matrix, cholesky_cov_matrix, ...

Variable types

Can a parameter have any type?

Variable types

Arrays can be declared as

```
real votes[5] = {1.4, 2.0, 3.1, 4.4, -1.1};
```

Containers can be segmented à la R:

```
real votes_quebec[3] = votes[2:4];
```

Variable types

Some handy functions:

`to_vector()`

`to_matrix()`

`to_array_1d()`

`to_array_2d()`

- when should I use an array or a vector / matrix?

Additional language blocks

- transformed data
- transformed parameters
- functions

The transformed data block allows us to:

- declare *fixed* variables which do not come from an exterior input, or are transformation of variables declared in the data block.
- manipulate and transform data.

The transformed parameter block allows us to:

- manipulate objects which depend on parameters and data.
- it does *not* allow us to write sampling statements.

What difference does it make whether I write a statement in the transformed data, transformed parameter, or model block?

The function block allows us to:

- define new functions
- define systems of algebraic and ordinary differential equations

Here's an example:

```
functions {
 real compute_norm (vector x) {
   real norm = 0;
   for (i in 1:length(x)) norm += x[i];
   return norm;
 }
}
```

Similarly:

```
functions {
 real compute_norm (real[] x) {
   real norm = 0;
   for (i in 1:length(x)) norm += x[i];
   return norm;
 }
}
```

```
functions { . . . }

data { . . . }

transformed data { . . . }

parameters { . . . }

transformed parameters { . . . }

model { . . . }

generated quantities { . . . }
```