Intro to rstan

Ian Laga

09/30/2019

Example 1: Truncated Exponential

For this example, we know the data come from an exponential distribution with rate λ, but we don't observe observations below L or above U (for this example, we know these truncation points). So, the distribution is specified by

$$\lambda \sim \pi_0(\theta)$$
$$Y|\lambda \sim Exp(\lambda)I(L < \lambda < U)$$

For now, we will consider a flat prior on λ. The model is specified in the stan file by

```
// The input data is a vector 'y' of length 'N'.
data {
  int<lower=0> N;
  real L;
  real U;
  real<lower=L,upper=U> y[N];
}

parameters {
  real<lower=0> lambda;
}

// The model to be estimated. We model the output
// 'y' to be exponentially distributed with rate 'lambda'
model {
  // If you wanted a prior, it would go here
  for (n in 1:N)
   y[n] ~ exponential(lambda) T[L,U]; # The truncation occurs between L and U
}
```

Example 2: Censored Exponential

This is similar to the truncated exponential, but now we know how many observations were censored. To incorporate these censored observations, we sample them as latent variables.

```
data {
  int<lower=0> N_obs;
  int<lower=0> N_cens;
  real y_obs[N_obs]; // vector[N]<lower = L, upper = U> foo
  real<lower=max(y_obs)> U;
}

parameters {
  real<lower=U> y_cens[N_cens];
  real<lower=0> lambda;
}

model {
  y_cens ~ exponential(lambda);
  y_obs ~ exponential(lambda);
}
```

Notice that we now have two likelihoods in the model part of the code and two parameters. Everything in the parameter section will be sampled from the posterior distribution.

Example 3: Censored Data with Covariates

We are extending the above example, except now,

$$\lambda = \beta X,$$

where X is known. We have to specify a few more data components now.

```
data {
  int<lower=0> N_obs;
  int<lower=0> N_cens;
  int<lower=0> K;
  matrix[N_obs, K] x_obs;
  matrix[N_cens, K] x_cens;
  real y_obs[N_obs];
  real<upper=min(y_obs)> U;
}

parameters {
  vector<lower = 0>[K] beta;
  real<lower = 0,upper=U> y_cens[N_cens];
}

model {
  y_cens ~ exponential(x_cens * beta);
  y_obs ~ exponential(x_obs * beta);
}
generated quantities {
  vector[K] dif = beta - beta[1];
}
```

Example 4: Updating the Log-Posterior Directly

Everything we have done so far uses built-in distributions. However, we can write our own posterior density as well. In Bayesian analysis, we need to evaluate the log-posterior at different parameter values. In Stan, we can add directly to the log-posterior using the "target" variable. Consider the modeling a standard exponential distribution. The log-likelihood is given by

$$L(\lambda, Y) = \log(1/\lambda) - Y/\lambda$$

Here we present several different ways to use the log-likelihood directly.

```
functions {
  real exp_mean_lpdf(vector x, real lambda) {
    vector[num_elements(x)] lprob;
    lprob = log(1/lambda) - x/lambda;
    return sum(lprob);
  }
}

data {
  int<lower=0> N;
  vector[N] y;
}

parameters {
  real<lower=0> lambda;
}

model {
  // target += log(1/lambda) - y / lambda;  // You can do it without the function
  // y ~ exp_mean(lambda); // Or you can use the function
  target += exp_mean_lpdf(y | lambda); // This is equivalent to the line above
  // notice that it doesn't have _lpdf when using ~ notation
}
```

Example 5: A More Complicated Example

Finally, let's take this all the way and see what we can model in Stan. This is a model from my own research.

$$Y_{ik} \sim Poisson\left(exp(d_i \frac{N_k}{N} b_{ik})\right), \qquad i \in \{1, \ldots, n\}, k \in \{1, \ldots, K\}$$

$$d_i \sim N(0, \sigma_d^2)$$

$$\pi(N_k) = \frac{1}{N_k} \qquad \text{for } k = K$$

$$\mathbf{b}_i \sim N(\alpha_i X_i + \beta \mathbf{Z}, \Sigma)$$

This is a fairly complicated model and would be incredibly difficult to code up using traditional MCMC (trust me, I did it). However, it's not too bad in Stan

```
data {
  int<lower=0> N;
  int<lower=0> n_i;
  int<lower=0> n_k;
  int<lower=0> x_size;
  int<lower=0> z_size;
  int<lower=0> n_known;
  matrix[x_size, n_i] x_cov;
  matrix[z_size, n_k] z_cov;
  int y[n_i,n_k];
  row_vector<lower=0,upper=N>[n_known] known;
}


parameters {
  vector[n_i] di;
  real<lower=0> sigma_di;
  real mu_alpha;
  row_vector<lower=1, upper=N>[n_k - n_known] unknown;
  row_vector[x_size] beta_par;
  row_vector[z_size] alpha_par;
  cholesky_factor_corr[n_k] L_Omega;
  vector<lower=0>[n_k] tau;
  matrix[n_i,n_k] eps;
}

transformed parameters {
  matrix[n_i, n_k] bias;
  bias = rep_matrix(beta_par * x_cov, n_k)' +
     rep_matrix((alpha_par * z_cov)', n_i)' +
     rep_matrix(di, n_k) + (diag_pre_multiply(tau, L_Omega) * eps')';
}

model {
  row_vector[n_k] prevalence;
  di ~ normal(0, sigma_di);
```

```
  target += sum(log(1 ./ unknown));

  to_vector(eps) ~ std_normal();

  prevalence = append_col(known, unknown) / N;
  for(i in 1:n_i){
    y[i,] ~ poisson(prevalence .* exp(bias[i,]));
  }
}

generated quantities{
  matrix[n_k, n_k] Corr;
  Corr = L_Omega * L_Omega';
}
```