Introduction to Bayesian inference using Rstan: practical 2

The aim of this practical is to provide some practical experience in writing Stan programmes and using the `rstan` package for posterior inference. Some sections require more experience with statistics than others. These are marked with an asterisk.

First load the `rstan` and `jrRstan` packages:

```
library("rstan")
library("jrRstan")
```

If you have enough RAM, set options to allow parallel computation:

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

1 *Binomial regression*

In Section 5.2 we considered a generalised linear model in which we assumed a *Poisson* error distribution. In this section we will consider another regression problem, but this time we will use a *binomial* error distribution. This is called *binomial regression*. In the special case where a *logistic*[1] link function is utilised, the name *logistic regression* is often used. We will use a logistic link function in this example.

[1] Other possibilities include the *probit* link or the *complementary log-log* link.

As you might expect, the proportion of people who suffer side effects from a drug typically depends on the dose of that drug they are given. The `jrRstan` package contains a data set called `sideeffect` which can be loaded via:

```
data(sideeffect)
head(sideeffect)

##   dose   n effects
## 1  0.9  46      17
## 2  1.1  72      22
## 3  1.8 118      52
## 4  2.3  96      58
## 5  3.0  84      56
## 6  3.3  53      43
```

For each of a number of doses (`dose`), the data set contains the number (`n`) of patients given a particular drug to treat a medical condition, and the number of those patients suffering from a particular side effect (`effects`).

Suppose that the aim is to model the number Y_i of the n_i patients receiving dose i that suffer side effects in terms of the dose $\tilde{x}_i$ of the drug they are given. The model can be expressed as:

$$Y_i | n_i, p_i \sim \text{Bin}(n_i, p_i), \quad \text{independently for } i = 1, 2, \ldots, N,$$

where here $N = 7$. We will mean centre the covariate, taking $x_i = \tilde{x}_i - \sum_{i=1}^{N} \tilde{x}_i / N$, and use a logistic link function to connect the linear predictor:

$$\eta_i = \beta_1 + \beta_2 x_i$$

to the probability p_i that an individual receiving dose i suffers side effects. In other words:[2]

$$p_i = \frac{e^{\eta_i}}{1 + e^{\eta_i}}.$$

or equivalently:[3]

$$\eta_i = \log\left(\frac{p_i}{1 - p_i}\right).$$

Our model contains two parameters: β_1 and β_2. We will adopt the following prior:

$$\beta_1 \sim \mathrm{N}(-0.27, 0.68^2), \quad \text{and, independently,} \quad \beta_2 \sim \mathrm{N}(0.47, 0.31^2).$$

- Write a Stan model to represent the model and prior above, remembering to mean-centre the covariate.
- Create a suitable data representation in R then compile and run the Stan programme.
- Check both numerical and graphical diagnostics.

[2] This is called the *logistic* transformation of η_i.

[3] This is called the *logit* (or *inverse logistic*) transformation of p_i.

2 *Random slope model for Oxboys data*

In Section 5.3 we examined the `Oxboys` data where $Y_{i,j}$ and $x_{i,j}$ denoted the j-th measurement of height and age on boy i, where $i = 1, \ldots, I$ and $j = 1, \ldots, J$. We can load the data in R through:

```
library(nlme)
data(Oxboys)
head(Oxboys)

## Grouped Data: height ~ age | Subject
##   Subject     age height Occasion
## 1       1 -1.0000  140.5        1
## 2       1 -0.7479  143.4        2
## 3       1 -0.4630  144.8        3
## 4       1 -0.1643  147.1        4
## 5       1 -0.0027  147.7        5
## 6       1  0.2466  150.2        6
```

In lectures, we considered a *random intercept* model in which the intercept term in a regression of the height of a boy on his age could vary between individuals. In this section, we will consider an extension with both a *random intercept* and a *random slope* that can vary from one boy to the next. In other words we will fit a hierarchical model of the form:

$$Y_{i,j} = \alpha_0 + \alpha_{1,i} + (\beta_0 + \beta_{1,i})x_{i,j} + \epsilon_{i,j} \quad \text{where} \quad \epsilon_{i,j}|\sigma^2 \sim \mathrm{N}(0, \sigma^2) \quad (1)$$

independently for $i = 1, 2, \ldots, I$, $j = 1, 2, \ldots, J$ with:

$$\alpha_{1,i}|\sigma^2_{\alpha_1} \sim \mathrm{N}(0, \sigma^2_{\alpha_1}) \quad \text{and} \quad \beta_{1,i}|\sigma^2_{\beta_1} \sim \mathrm{N}(0, \sigma^2_{\beta_1}) \quad (2)$$

independently for $i = 1, 2, \ldots, I$. We now have two sets of random effects: the $\alpha_{1,i}$ and the $\beta_{1,i}$.

We complete our hierarchical model by specifying a prior for the parameters in the "top" level of the model (1):

$$\alpha_0 \sim \mathrm{N}(m_{\alpha_0}, s^2_{\alpha_0}), \quad \beta_0 \sim \mathrm{N}(m_{\beta_0}, s^2_{\beta_0}), \quad \sigma^2 \sim \mathrm{Gam}(a_{\sigma^2}, b_{\sigma^2}),$$

and a *hyperprior* for the parameters in the "bottom" level of the model (2):

$$\sigma^2_\alpha \sim \mathrm{Gam}(a_{\sigma^2_\alpha}, b_{\sigma^2_\alpha}), \quad \sigma^2_\beta \sim \mathrm{Gam}(a_{\sigma^2_\beta}, b_{\sigma^2_\beta}).$$

As in Chapter 5, when fitting the model we will take:

$$\begin{aligned} m_{\alpha_0} &= 140, \quad s_{\alpha_0} = 3, & m_{\beta_0} &= 15, \quad s_{\beta_0} = 7.5 \\ a_{\sigma^2} &= 1.1, \quad b_{\sigma^2} = 0.025, \\ a_{\sigma^2_\alpha} &= 1.1, \quad b_{\sigma^2_\alpha} = 0.05. \end{aligned}$$

For the constants $a_{\sigma^2_\beta}$ and $b_{\sigma^2_\beta}$ in the extra hyperprior for σ^2_β we will take:

$$a_{\sigma^2_\beta} = 1.1, \quad b_{\sigma^2_\beta} = 0.1.$$

- Extend the Stan programme for the random intercept model to additionally include a random slope.
- Create a suitable data representation in R then compile and run the Stan programme.
- Check both numerical and graphical diagnostics.
- **More difficult:** Modify the Stan programmes for the random intercept model and random slope model to include a `generated quantities` block which computes the deviance of the model. (This will be similar to the code for the linear regression model in Chapter 3). Compute the DIC for each model and use it to decide which model is better according to this criterion.

3 *Hierarchical model for rat tumour data*

In this section we will consider a well-known data set that is amenable to Bayesian hierarchical modelling. The data are available from the `jrRstan` package and concern the proportion of rats developing tumours in a number of laboratory studies. We can load the data via:

```
data(rats)
head(rats)
```

```
##   y  n
## 1 0 20
## 2 0 20
## 3 0 20
## 4 0 20
## 5 0 20
## 6 0 20
```

Our goal is to learn about the population probability of tumour amongst rats.

For study i, $i = 1, \ldots, N$, n_i denotes the total number of rats and Y_i denotes the number of rats who developed a tumour. To allow for the differences between studies in rats and experimental conditions, we let the probability of developing a tumour vary between studies and denote it by p_i for study i. We then assume the Y_i are independent binomial random variables given the study-specific probabilities p_i:

$$Y_i | n_i, p_i \sim \text{Bin}(n_i, p_i), \quad \text{independently for } i = 1, 2, \ldots, N,$$

where here $N = 71$. Suppose we initially thought the probabilities p_i might be around 0.1. If we were to learn that one probability p_i was larger than 0.1, this would cause us to revise upwards our "best guess" at the probabilities for other studies because we expect the p_i to be similar. To formalise this idea, we will assume the p_i are samples from some population distribution with common mean and variance. This is an example of a *Bayesian hierarchical model*. For convenience, we will reparameterise the binomial distributions so that instead of working with probabilities p_i we work with their logit transformations:

$$\theta_i = \log\left(\frac{p_i}{1 - p_i}\right)$$

and assume that

$$\theta_i | \mu, \sigma^2 \sim \text{N}(\mu, \sigma^2).$$

We allow the parameters μ and σ^2 to be unknown and give them prior distributions:

$$\mu \sim \text{N}(m_\mu, s_\mu^2) \quad \text{and} \quad \sigma^2 \sim \text{Gam}(a_{\sigma^2}, b_{\sigma^2}).$$

The mean μ can be thought of as the population value for the logit-probability of tumour amongst rats. The variance σ^2 quantifies the degree of variation in the population; the larger its value, the more heterogeneity we see in the probabilities between studies.

- Write a Stan model to represent the hierarchical model above.
- Take
$$m_\mu = 0, \quad s_\mu^2 = 2, \quad a_{\sigma^2} = 1.1, \quad b_{\sigma^2} = 1.3.$$
Create a suitable data representation in R then compile and run the Stan programme.
- Check both numerical and graphical diagnostics.

4 *Three state mixture model**

In Section 5.4 we fitted a mixture model with $K = 2$ components to the `faithful` data on the times between successive eruptions of the Old Faithful geyser:

```
data(faithful)
head(faithful)

##   eruptions waiting
## 1     3.600      79
## 2     1.800      54
## 3     3.333      74
## 4     2.283      62
## 5     4.533      85
## 6     2.883      55
```

In the two-component case, we had component membership probabilities (π_1, π_2) which summed to one and so we reparameterised the model as $\pi_1 = \pi$ and $\pi_2 = 1 - \pi$ where $0 \leq \pi \leq 1$. We represented this in Stan as a constrained real:

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

In the more general case with K components we have vector $(\pi_1, \pi_2, \ldots, \pi_K)$ on the K-dimensional simplex. This can be represented in Stan as a `simplex` vector type:

```
simplex[K] pi_vec;
```

In the case of two-components, we assigned `pi` a symmetric Beta prior:

```
// Prior:
pi ~ beta(a_pi, a_pi);
```

The multivariate generalisation of this is a symmetric Dirichlet prior.

- Use the `lookup` function to find the Dirichlet distribution in the Stan manual:

  ```
  lookup("Dirichlet") # Simply search for string rather than
                      # the name of a particular R function.
  ```

  Try to generalise the Stan programme from Figure 5.7 so that it allows the number K of states to be a component of the `list` you pass to the `stan` function through its `data` argument.

- Test the programme by compiling it then running it with $K = 2$ and $K = 3$ components.

5 *Survival with right censoring**

In this section we will consider another generalised linear model, this time assuming an *exponential* error distribution. The data we will consider concern the survival times T_n for $n = 1, \ldots, N$, in months, of $N = 148$ renal patients following kidney transplants. There is one covariate x_n for each patient, namely the total number of HLA-B or DR antigen mismatches between the donor and recipient; we might expect survival times to be shorter if there are more mismatches. For some patients, the month of death is observed. However, for

others the survival time is *right-censored* meaning we do not observe the time of death, only a time at which the patient was known still to be alive. This can happen for lots of reasons, for example, the study may have ended before the patient died or the patient may have been lost to follow-up. We introduce an indicator variable s_n representing the censoring status of the patient. We set $s_n = 1$ if the corresponding observation t_n on T_n represents the survival time of patient n. On the other hand, we set $s_n = 0$ if the observation t_n on T_n is a right-censored time, meaning all we know is that the survival time of patient n is greater than t_n, i.e. $T_n > t_n$.

The `renal` data set in the `jrRstan` package contains the survival data:

```
data(renal)
head(renal)

##       t status x
## 1 0.035      1 3
## 2 0.068      1 0
## 3 0.100      1 0
## 4 0.101      1 1
## 5 0.167      0 4
## 6 0.168      1 2
```

Our model for the survival times can be expressed as:

$$T_n|\lambda_n \sim \text{Exp}(\lambda_n), \quad \text{independently for } n = 1, 2, \ldots, N,$$

where $\lambda_n = \exp(\eta_n)$ is the reciprocal of the mean survival time for patient n. The corresponding *linear predictor* η_n takes the form:

$$\eta_n = \beta_1 + \beta_2 x_n.$$

We adopt the following prior for the two model parameters:

$$\beta_1 \sim \text{N}(0, 30^2), \quad \text{and, independently,} \quad \beta_2 \sim \text{N}(0, 30^2).$$

If all the survival times were observed, i.e. if $s_n = 1$ for all $n = 1, \ldots, N$, the likelihood would take the form:

$$\prod_{n=1}^{N} p(t_n|\lambda_n)$$

in which patient n contributes $p(t_n|\lambda_n)$ to the likelihood function, namely the density function of the $\text{Exp}(\lambda_n)$ distribution evaluated at the observed survival time $T_n = t_n$. However, if the time for patient n is right-censored our observation t_n does not represent the survival time T_n, and we only know that $T_n > t_n$. For such patients the contribution to the likelihood is therefore $\Pr(T_n > t_n|\lambda_n)$ which is the *survival function*, or *complementary cumulative distribution function*, of the $\text{Exp}(\lambda_n)$ distribution evaluated at the censored time t_n. Overall, therefore, the likelihood takes the form

$$\prod_{n:s_n=1} p(t_n|\lambda_n) \times \prod_{n:s_n=0} \Pr(T_n > t_n|\lambda_n)$$

where the first term is a product over the patients for whom we observe the survival times and the second term is a product over the patients whose survival times are right-censored. Because the Stan modelling language is very flexible, we can handle the non-standard second term in the likelihood by incrementing the log posterior density using the `target` keyword[4] and the log complementary cumulative distribution function of the exponential distribution, `exponential_lccdf`.

[4] As in the mixture model example.

- Write a Stan model to represent the model and prior above.
- Create a suitable data representation in R then compile and run the Stan programme.
- Check both numerical and graphical diagnostics.

Solutions

Solutions are available as a vignette:

```
library("jrRstan")
vignette("solutions2", package="jrRstan")
```