Practical 5: Bayesian inference on Covid-19 incidence in England

This practical is to be completed **individually**. What is submitted must be solely your own work. You can discuss the mathematical and statistical aspects of the practical, but code must not be shared with others. We have automatic systems available for checking for this, but in addition students tend to make distinctive errors in coding, or use distinctively convoluted solutions to problems: these tend to stand out even if you do the obvious things to try and hide the code sharing. You can use code from the lecture notes without citation, but if you use any other code that you did not write yourself you should cite where it came from.

Data on deaths each day from Covid-19 provide some of the most reliable data on the state of the epidemic, because the data constitute a fairly complete record of fatal cases. There is also reliable information available on the distribution of time from infection to death in fatal cases[1]. This opens up the possibility of inferring the number of new (fatal) infections each day, from the data on fatalities each day. This can not be done in an entirely model free way, but it is sensible to use models that attempt to make the minimum of assumptions.

One general approach is to base a model only on the assumptions that the number of new infections per day changes fairly smoothly from day to day, and that the measured distribution of time from infection to death is correct. Since infections are non-negative, we'll model the log of the number of infections per day, x_i (on day i), and make the model prior assumption that it follows a second order random walk. This means that we assume that $x_{i+1} - 2x_i + x_{i-1} \sim N(0, \tau)$ (where τ is the precision - one over the variance). Re-arranging, we have

$$x_{i+1} \sim N(2x_i - x_{i-1}, \tau)$$

for $i > 1$. A simple way to start the process is to assume $x_1 \sim N(0, \tau_0)$ and $x_2 \sim N(x_1, \tau)$. So the number of new infections on day i is $n_i = \exp(x_i)$. To complete this *incidence* model let's specify that $\tau_0 = 0.01$ and that $\tau \sim \text{Gamma}(4, 0.04)$.

To permit inference about n_i we need to relate n_i to the available data on daily numbers of deaths, using the infection to death duration distribution. There are a number of published studies on this. Here use a distribution based on data from the ISARIC study of 24000+ hospitalized patients who died from Covid, which suggests that the fatal disease duration, D, can be modelled as $\log D \sim N(3.235, 0.4147^2)$ (parameterized in terms of variance, not precision here). `dlnorm` in R lets you evaluate the density of D. Because of the long disease durations, using the density as a model for the probability of a disease duration rounded to the nearest day is reasonable. Writing π_D for the log normal density of D, define a square matrix, $\mathbf{B}$, with elements

$$B_{ij} = \begin{cases} \pi_D(i-j) & j \leq i \\ 0 & j > i \end{cases}$$

Let m_i be the expected number of deaths on day i. We have $\mathbf{m} = \mathbf{Bn}$. To complete the model assume that the observed number of deaths on day i is $y_i \sim \text{Poi}(m_i)$. Note that this formulation only makes sense provided that we are modelling a death series that starts with a reasonable sequence of zeroes (otherwise we would need a non-square $\mathbf{B}$, with n starting well before m in time).

Here are the data on daily hospital deaths with Covid in England for the first 100 days from March 2nd 2020, from NHS England.

```
y <-c(1,2,0,2,2,0,4,4,1,9,14,20,22,27,40,46,66,63,105,103,149,159,204,263,326,353,360,437,498,576,645,647,
700,778,743,727,813,900,792,740,779,718,699,646,686,639,610,571,524,566,486,502,451,438,386,380,345,341,
325,313,306,268,251,259,252,266,256,215,203,196,166,183,162,180,172,167,138,160,144,153,150,122,128,116,
133,139,122,124,117,92,83,94,109,111,83,86,83,80,73,69)
```

There were no Covid deaths prior to this date, and for modelling purposes it makes sense to extend the data by appending 20 days of zero deaths to the start of the data. Note that while the same analysis can be conducted including deaths outside hospital, the interpretation is then more problematic: deaths outside hospital are usually of very frail patients in care homes, where the infection process does not reflect the general community transmission of most interest, and where the hospital record derived fatal disease duration distributions may not be applicable.

Your task is to implement and sample from the model for these data, using JAGS and `rjags`. Your aim should be to produce the posterior mean for the n_i trajectory, along with 95% credible intervals. You should produce a single summary plot, showing the daily deaths against day of the year (day 1 being Jan 1 2020), with the posterior mean incidence n_i and 95% credible interval overlaid on the plot (probably using `lines`). Also overlay the

[1] although there is also a good deal of poor information on duration distributions published by people you have not understood the difficulties of estimation under right truncation (where you have a more complete record of short durations than long durations.)

posterior mean for m_i to illustrate model fit. Finally mark the first day of UK lockdown (24th March 2020) on the plot as a vertical line on the appropriate day.

Before producing the final plot, your code should include the selected diagnostics that you consider most relevant. Simply running every possible diagnostic will lose marks - you should be selective and your code comments should include brief interpretation of the diagnostics, and a recommendation for the number of sampling iterations to run and how much if any burn-in is desirable. Diagnostic plots are allowed but must occupy **at most** 2 A4 pdf pages when printed to file using R's `pdf` graphics driver: the plots must be easily readable. However, do not include calls to graphics drivers (or `dev.off`) in your final submitted code.
Some hints:

1. The JAGS manual is available on the course materials page on Learn.
2. Note that `%*%` is used for matrix multiplication within JAGS, while `dgamma`, `dpois` and `dnorm` are the required densities (don't forget that `dnorm` in JAGS is parameterized in terms of precision, not variance).
3. Create **B** in R and pass it to JAGS, rather than creating it in JAGS.
4. See `?julian` and `?as.Date` for easy handling of dates and day of year in R. For example, this is quite useful: `julian(as.Date("2020-3-24"),origin=as.Date("2019-12-31"))`
5. Incidence (n_i) estimates will become unstable towards the end of the data, so do not plot them for the last 20 days or so of data.

What to submit:

1. One JAGS model specification file called `model.jags`. Include your name and student number in a comment in the first line of this file.
2. One R file containing the code to run your JAGS model and produce the specified results plot *as the final plot produced*. The first comment line should match the one in the JAGS file. Your submitted code should **not** set a working directory or refer to any specific directory. Also it should **not** call any graphics driver functions (e.g. `pdf` or `dev.off`). Your code should simply assume that the working directory has been set to the location of `model.jags`, before your code has been run. Your submitted code should use 10000 iterations of the JAGS sampler (this is to ensure fast enough run times for marking - not because the number is statistically ideal!). Your code can use anything supplied with base R, `jags`, `coda` and `ggplot2` but not other packages. When sourced your code should simply run and produce the required plot with no user interaction being required. Graphics devices and working directories will be set by us during marking, so it is essential that your code does not interfere with this. You will lose marks if you do not following these instructions and we have to manually modify your code to get it to run.

The deadline is 16:00 Friday 3rd December. It is strongly recommended that you submit well in advance of this, as last minute computer glitches will not be taken as an excuse for late submission[2].

Marking Scheme: Full marks will be obtained for code that:

1. is clearly commented, well laid out and well structured.
2. is reasonably clean and concise and avoids significant inefficiencies (e.g. JAGS sampling takes less than 2 mins for 10000 iterations for me).
3. includes sensibly selected diagnostics, demonstrating some engagement with and understanding of the lecture material on this topic.
4. sticks exactly to the submission specification without requiring manual intervention.
5. has been produced individually - collaborating on code is not acceptable for this project.
6. Produces a final plot clear enough that it could be published without misleading.
7. avoids significant faults not covered by the above!

[2]This is for reasons of fairness - otherwise it becomes too easy to game the system for an extension.