Practical 5 (Individual): UK excess deaths 2020-22

'Excess deaths' are the number of deaths over some period, relative to what would have been expected given data for some previous period. One way to get the expected number of deaths, for the UK say, is to take an average over the previous 5 years. Since death rates have a seasonal pattern, with more deaths in the winter, it is usual to average by week of the year. i.e. to compute the expected number of deaths in week 1 of the year, week 2, week 3 and so on. The obvious problem with such averages is that they will underestimate the number of deaths we should expect if the population is growing and/or ageing.

Less problematic is to compute the expected number of deaths from data on the annual probability of death for each one year age group, coupled with the populations of each age group at the start of the period of interest. In the UK both sets of data are available from the Office for National Statistics (ONS). The probability of death data are based on 3 year periods (e.g. 2017-2019). Given these data we can simply apply the dying and ageing process to update the population in each age group and obtain the expected number of deaths each week. That is we iterate a basic demographic model for the population in which the death process is based on the age specific death rates observed over the previous 3 years. That way we get the expected number of deaths if death rates stay the same, but without assuming that the population has stayed constant, or that the population is not ageing.

In this practical you will implement such a demographic model for England and Wales, to predict the expected number of deaths per week from the beginning of 2020 if death rates had stayed at the levels seen in 2017-19. The model will include an adjustment to allow for seasonal variation in mortality rates. You will then compare this to the actual deaths per week that occurred over this period[1], to obtain an excess deaths time series, and then model this series using a simple Bayesian model in JAGS.

The population model will have 101 age classes - from age 0 to age 100. Let N_i be the population in age class i. For each age class we have the instantaneous per capita death rate per year m_i. If an age class has population N_i at the start of the year, it will have population $N_i e^{-m_i}$ at the year end. In the absence of seasonality, if the population is N_i at the start of a week it will be $N_i e^{-m_i/52}$ at the end of the week. Hence the expected proportion of N_i dying in a week is $q_i = 1 - e^{-m_i/52}$. In reality there is seasonality in the death rates. A simple model is that the proportion dying is $d_j q_i$ in week j, where d_j can be computed from previous data. So for week j the model becomes

$$D_i = d_j q_i N_i, \quad N_i^* = N_i - D_i, \quad N_i^+ = N_i^* 51/52 + N_{i-1}^*/52, \quad i = 1, \ldots, 101$$

Assuming a constant birth rate, we can set N_0 to the initial value of the age 0 population, i.e. the value of N_1 at the start of simulation. If N_i is the population in age class i at the start of week j then N_i^+ is the population in the same age class at the start of week $j+1$, while D_i is the number of deaths in that age class over week j. So each week the model first computes the deaths and reduces the populations appropriately, and then applies the ageing process, moving 1/52th of the population of an age class up an age band. The ageing model is a little crude, just to avoid you having to store weekly age classes instead of yearly ones: the crudeness makes very little difference to the results here, but leads to slightly overstating the number of deaths. Changing the model to $D_i = 0.9885 d_j q_i N_i$ corrects this.

By summing up the D_i over age classes, you get the predicted number of deaths each week. The m_i are rather different between men and women, so the above model needs to be run separately for each sex, and the resulting weekly predicted deaths summed. On the Learn page are two text files of data. `lt1720uk.dat` contains columns

- `age` - the age classes `0`:0-1 years, `1`: 1-2 years, etc.
- `fpop17` and `mpop17` - the female and male populations in each 1 year age class at the start of 2017.
- `fpop20` and `mpop20` - the same for the start of 2020.
- `mf` and `mm` - female and male annual death rates for each 1-year age band, computed from 2017-19 data.

`death1722uk.dat` contains columns

- `deaths` the number of deaths that week.
- `week` the week since the start of 2017. 2020 starts in week 157.
- `d` the mortality rate modifier, d_j, value for that week.

[1] for weekly deaths ONS only provide England and Wales data and the Scottish and Northern Ireland data have to be obtained from the respective devolved administrations.

1. Write a function to take vectors of starting populations (male and female) by one year age class, corresponding vectors of m_i values, and a vector of d_j values (not sex specific). The function should iterate this model forward `length(d)` weeks. The function should return the predicted total number of deaths each week as a vector. It is advisable to test your model implementation by running for three years from the start of 2017 and checking that the total number of simulated deaths matches the total observed to within a few hundred. Also check that the predictions look plausible in relation to the actual data.

2. Compute the difference between the total actual deaths and the total predicted deaths from the start of 2020 to the end of the data. This is the excess deaths. Do the same just for 2020.

3. Plot the observed deaths against week (with the lower limit on the y axis scale being zero) and overlay a continuous curve showing the predicted deaths. Report the excess deaths in 2020 and overall in the plot title.

4. Compute the vector of excess deaths each week, and produce a plot of the cumulative excess deaths by week (`cumsum` is useful).

5. A simple time series model for the excess deaths, x_i, in week i is

$$\mu_1 = \alpha, \quad \mu_{i+1} = (x_i - \alpha)\rho + \alpha, \quad x_i \sim t_k(\mu_i, \tau)$$

   where $i = 1, \ldots, n$ (although we do not actually require μ_{n+1}, of course). $t_k(\mu_i, \tau)$ denotes a scaled t distribution with mean μ_i, precision parameter τ and k degrees of freedom (see the JAGS manual). To complete the Bayesian specification of this model requires priors. Let these be $\tau \sim \exp(1)$, $\rho \sim U(0, 0.9)$, $\alpha \sim N(0, \tau = 0.0001)$ and $k \sim U(2, 100)$. The model is useful for highlighting the parts of the series that are most unusual. Implement the model in JAGS and use it to draw 10000 samples form the posterior densities of the $\boldsymbol{\mu}$ vector, ρ and k given the observed x_i data. However set the x_i for weeks, 51, 52, 53,105 and 106 to NA as these are Christmas/New Year data with various recording problems (delays mean many deaths are reported late). You *must* call your jags file `model.jags` (all lower case).

6. Produce a trace plots and histograms for ρ.

7. Compute the posterior expected value vector for $\boldsymbol{\mu}$.

8. Produce a plot showing every 50th sampled $\boldsymbol{\mu}$ vector (against week) as a grey curve, with the estimated expectation for $\boldsymbol{\mu}$ overlaid in blue. Add the observed excess deaths, x_i, over these curves as black symbols, with the x_is not used for inference (i.e. set to NA in the JAGS call) shown in red.

9. Finally plot the 'residuals' from this model, $x_i - E(\mu_i)$, against time.

Your submitted code should include no setting up of graphics devices or manipulation of the plot window via `par` and should only produce the 6 plots asked for. Make sure that they have proper axes labels. If you use `ggplot` your code should *explicitly* print the plot object. These rules are because your code will be automatically run to produce the plots. To view the plots we will use code like this:

```
pdf("excess.pdf",height=10,width=6)
par(mfrow=c(2,1))
[your code here]
dev.off()
```

It is a good idea to check that your plots look reasonable when printed this way *but do not include the* `pdf`, `par` *or* `dev.off` *calls in your submitted code*. The auto-running process also means that it is essential that your jags model file is called `model.jags` and not something else. You can use any packages supplied with base R (i.e. base + recommended packages) and `rjags`. `coda` and `ggplot2` can also be used if you like. Other packages may not be used.

Two (commented) files, your R code file and your `model.jags` file, are to be submitted on Learn by 12:00 Friday 2nd December 2022. Format the code and comments tidily, so that they display nicely on an 80 character width display, without line wraps (or only occasional ones). No extensions are available on this course. So late work will automatically attract a hefty penalty (of 100% after work has been marked and returned). Technology failures will not be accepted as a reason for lateness (unless it is provably the case that Learn was unavailable for an extended period), so aim to submit ahead of time.

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

1. does what it is supposed to do, and has been coded in R approximately as indicated (that is marks will be lost for simply finding a package or online code that simplifies the task for you).
2. produces correct results displayed in appropriately labelled plots.
3. is carefully commented, so that someone reviewing the code can easily tell exactly what it is for, what it is doing and how it is doing it without having read this sheet, or knowing anything else about the code. Note that *easily tell* implies that the comments must also be as clear and *concise* as possible. You should assume that the reader knows basic R, but not that they know exactly what every function in R does.
4. is well structured and laid out, so that the code itself, and its underlying logic, are easy to follow.
5. is reasonably efficient, meaning that care has been taken to avoid excessively slow code that e.g. repeatedly computes the same quantity.
6. contains no evidence of having been copied, in whole or in part, from other students on this course, students at other universities (there are now tools to help detect this, including internationally), online sources etc.