Running the Bayesian Inference sampler with the joint-sex model

model version mk8

October 18, 2023

Celeste Winant

Written very informally – a supplement to the USCountyBayesianEstimationMethodsProtocol

****Note my convention. Any line that starts with $~: is followed by a command for a Linux/Unix xterminal. Any line that starts with >> is followed by a command for R (or rstudio)*

To run the Bayesian inference sampler as packaged in SubregionalMortalityBayesianEstimation_mk8.zip

First, move the archive to your top-level working directory, then expand the archive.

```
$~: unzip SubregionalMortalityBayesianEstimation_mk8.zip
```

This will create a folder called SubregionalMortalityBayesianEstimation_mk8/
In which you will find

- The USCounty Methods Protocol (“USCountyBayesianEstimationMethodsProtocol20230903web.pdf”) for reference.
- These instructions (“READMEWrapperScriptNotesYYYYMMDD.pdf”)
- The R-script (“WrapperScriptForJointModelmk8.R”) that launches Rstan / Stan. I also refer to this as the wrapper script.
- The Stan model (“joint_model_mk8.stan”) called by Stan
- The principal component file (“HMD_pcs.rda”) – formatted as an R binary
- A subfolder (for the fictitious state, “XX”) with a subdirectory structure for pulling inputs and writing output.
    - subdirectory InputDB/ with
        - “XXpop.csv” (Population inputs)
        - “XXdeaths.txt” (Deaths input. The death tabulations have been modified from actual state data with a (-1,0,1) random jitter to protect actual death counts from restricted-access files)
        - Note that the 5-digit Population Code (PopCode) has been recoded in the input files to shield the identity of the actual county which the input data were derived from.
    - subdirectory fit/ to which the output is written (it is currently empty)

The simplest way to test out the code is to run the wrapper script from the directory where it resides (SubregionalMortalityBayesianEstimation_mk8/). The paths in the script will load the model and principal components from the same directory, then descend the subdirectory structure to read the inputs (and write the output). You’ll see in the package that there is a folder “XX” (the name of our fabricated state) under which are two folders: “InputDB” (in which are the deaths and population files) and “fit” (to which the file with the output, ‘fit’ is written).

Read the USCountyBayesianEstimationMethodsProtocol20230903.pdf (Appendix B) to learn how to format your input data.

Name the files with the following convention: assign the name of your state (or country) to the variable 's'

```
>> s <- "FRANCE" ##(or "CA" or "XX")
```

The deaths file should be named (using R semantics)

```
>> paste0(s,"deaths.txt") ## (e.g. FRANCEdeaths.txt)
```

The population file should be named

```
>> paste0(s,"pop.csv")
```

The use of .txt for one file and .csv for the other is idiosyncratic. Feel free to use one or the other, but modify the lines that invoke read.csv() in the wrapper script accordingly.

The wrapper script will load the principal components and the model from those files in the same directory.

The principal components will be pulled from the R-binary file, HMD_pcs.rda, which contains principal components constructed from the 1959-2018 USMDB 5x1 life tables for all states. There are four different sets of components constructed various sex-specific lifetables:

1. Female-only (f)
2. Male-only (m)
3. Both-sex-only (b)
4. "Stacked" male and female life tables (mf)

The model calls on principal components from the stacked life tables (mf) only (so the wrapper reads the stacked set and the other three sets aren't used)

You should read the methods protocol to learn how the joint-sex model deviates from the model we worked with a few summers ago.

The script will require a large amount of memory and it's unlikely you can run the sampling (fit = sampling(….)) within rstudio. However, when you are reading and reformatting the inputs for the first time, I would run that section of code one line at a time in rstudio to make sure each step goes as intended. Run through the line below in rstudio

```
>> save(dat, file =file.path(…))
```

Which writes the input in a single tibble to file.

Then, relaunch R from an xterminal and source the entire script:

```
>> source("WrapperScriptForJointModelmk8.r")
```

Once sampling() is invoked, updates to percentage completed will print to the screen.

The program takes a very long time (a rough estimate is about 20 minutes per county).

Once the fit is complete, the program writes the large list-object called 'fit' to file.

There are a couple of ways to pull desired variables from the fit, both with the extract() function in the rstan library. This function allows you to extract each variable that is defined in the model. Note that the variables are labeled as, for example, 'logmx(i,j,k,l)'. where i,j,k,l are the respective indices for that variable and 'logmx' is the variable name for log-mortality-rate as defined in the model file. The index order begins with the sample number (1,…,10000), followed by the same indices and index order as specified for the variable in the methods protocol (x=age,s=sex,a=county,y=year).

If you want a data-frame where the variable names are the column-names, use the rstan function extract(). For example, to create a logmx dataframe, issue the command:

```
>> logmx <- as.data.frame(rstan::extract(fit,par="logmx"))
```

This creates a data frame with m(=number of samples) rows and n(=total multiplicity of the given parameter (e.g. i*j*k*l)) columns. The row-index is the sample number. The column name is the parameter name (with index) (e.g. 'logmx(4,1,2,5)'). I have it on good authority that you can use the function dplyr::separate() to parse the column names but have never tried it myself since I usually extract via the method below.

If you want something more compact (especially if you have concerns about memory), still use rstan::extract() but specify that you only want the [[1]] element of the output and do not reformat as a data-frame. This creates an n-dimensional array (where n = number of parameter indices + 1) for a given variable, where the m,i,j,k,l value of the array corresponds to the m,i,j,k,l value of the variable. The index order begins with m, the Markov-chain sample number (1,…,10000), followed by the same indices and index order as specified for the variable in the methods protocol (i=age,j=sex,k=county,l=year). For example, if we wanted to create the array of logmx values, use the command

```
>> logmx <- rstan::extract(fit,par='logmx')[[1]]
```

This creates a 5-dimensional numeric array stripped of any identifiers, but given our definition above, the dimensions are ordered (sample,age,sex,county,year). You can re-apply the identifiers in subsequent analysis