Practical 3. Introduction to R and cost-effectiveness analysis using BCEA

Monday, 20 June 2022

Lecture 3 PDF version

This is a very brief introduction to `R` (which can be downloaded from the website https://cran.r-project.org/) and its capabilities. `R` is available on the UCL Desktops and you can actually use the `R` frontend software called `Rstudio`. You do not *have* to use `Rstudio` and in fact, the standard `R` terminal would do just as well. However, `Rstudio` has some nice features and it is a better text editor/file manager, so it is probably a good idea to start using it. It will be extremely focussed on the characteristics that are instrumental to do health economic evaluations using a combination of `R`, `BUGS` and some useful packages (such as `BCEA`). Thus it is by no means exhaustive!

(Ridiculously short and incomplete) Introduction to `R`

When you open the `R` terminal, you are presented with the possibility of typing commands. You may want to open a text editor (e.g. the simple one built into the R Windows interface) in which you can type directly these commands, and save them to a script for future use. Another possibility is to use `R` from within a more sophisticated "integrated development environment", such as `Rstudio`, which has many more features than the basic Windows interface.

In any case, `R` is a very powerful tool; more importantly, it is free and you can find a wealth of documentation on the internet. `R` has a set of built-in commands, which you can use for basic operations. However, there are also many add-on packages containing sets of functions designed to perform specific statistical tasks. These packages can be installed to *your* `R` by typing the command

```
install.packages("package_name")
```

(assuming you have an internet connection). This command only needs to be executed one time. Once a package is installed in your local `library` (a collection of packages) you can make it available to the current `R` session by typing the command

```
library(package_name)
```

For these practicals, you will need to install and load the packages:

- `BCEA`, which can be used to post-process the results of a (Bayesian) model to perform a health economic evaluation.
- `R2OpenBUGS`, which can be used to interface `R` and `BUGS` .

You do this by typing in your `R` terminal the commands

```
install.packages("BCEA")
install.packages("R2OpenBUGS")
library(BCEA)
library(R2OpenBUGS)
```

Both `BCEA`and `R2OpenBUGS` will automatically load other packages that they *depend on* — this means that in order to work, they need to access functions that are part of other packages.

If you wish so, you can use `JAGS` in the practicals. To this end (and assuming you have actually installed the current version of `JAGS` to your computer), you will need to also install the package `R2jagsS`, which you can do by typing in your `R` terminal

```
install.packages("R2jags")
```

Notice that if you decide to use `JAGS` instead of `BUGS`, you will need to slightly modify some of the commands.

Once a package is loaded to your `R` workspace, you can type the command `help(package_name)`, which will open a window displaying a description of the package. For example `help(BCEA)` provides some basic information (including details of the current version). You can use the command `help` also on specific functions within the package, e.g. typing `help(bcea)` describes in detail how to use the `bcea` function (notice that in this case the package name is typeset in uppercase, while the function is lowercase!).

The very basic commands that are required to do a typical R session working with BUGS and BCEAwill be given and described later or in the scripts that we refer to in the practicals.

MCMC in R/BUGS

Consider Laplace's analysis of birth data, which included $y = 241\,945$ females out of $n = 493\,527$ babies born in Paris between 1745 and 1770. Assume the following modelling assumptions:

$$\begin{aligned} y &\sim \text{Binomial}(\theta, n) \\ \theta &\sim \text{Uniform}(0, 1). \end{aligned}$$

1. Write BUGS code to encode the modelling assumptions above and save it to a .txt file (you can choose whatever name and location you want. We assume you've chosen ModelLaplace.txt, which you have saved in the current directory).
2. Open R , which will be used to pre-process the data, call BUGS in background to perform the MCMC estimation and then post-process the results. At the terminal, type the following commands

```
y <- 241945
n <- 493527
data <- list(y=y,n=n)
```

This defines the variables y and n into the R workspace and create a list (named data), which contains them.

(Notice that in R you can use the notation "<-" to indicate assignment to a variable, as well as the more straightforward notation "=". Thus the commands a <- b and c = b create two variables a and c which both take the value associated with the variable b — irrespective of the sign used to define the assignment operator). Technically (and at the level of the data stored in the computer memory) the two statements are not identical, but for all intent and purposes, they are.

Now, set up some utility variables, such as the location of the BUGS model file, the parameters to be monitored and the initial values, by typing

```
filein <- "PATH_TO_FILE/NAME_OF_FILE.txt"
params <- "theta"
inits_det <- list(list(theta=.1),list(theta=.9))
inits_ran <- function(){list(theta=runif(1))}
```

The R workspace now contains a variable filein (a text string with the path to your model file, e.g. C:/bayes-hecourse/4_bcea/ModelLaplace.txt); a variable params (another text string containing the name of the variable you want to monitor); the two variables inits_det and inits_ran can be used to instruct BUGS which initial values to use. The former is a list containing two initial values for the node theta (they are arbitrarily set to 0.1 and 0.9, so that two chains can be run, starting from different points). The latter is an R function, which creates a list with a variable theta, which is assigned a random draw from a Uniform(0, 1) distribution.
3. Call BUGS from within R by using the following commands

```
model <- bugs(data=data,inits=inits_det,
          parameters.to.save=params,model.file=filein,
          n.chains=2,n.iter=10000,n.burnin=4500,
          n.thin=1,DIC=TRUE)
```

— notice that you can either use the syntax inits=inits_det or inits=inits_ran. BUGS is called in background and executes the MCMC analysis. When it has completed (which should be extremely quick in this simple case), you will regain use of the R terminal.

The R object model contains many variables; typing

```
names(model)
```

will print a list of them. Each can be accessed using the R "$" notation. For example, typing

```
model$n.iter
```

will print the number of iterations used by BUGS .
4. Check the results by using the command

```
print(model,digits=3,intervals=c(0.025,0.975))
```

This instructs R to give a tabular output reporting summary statistics (specifically reporting the 2.5% and 97.5% percentiles) from the posterior distribution(s) of the node(s) monitored, using 3 significant digits.
What can you say about convergence by just looking at the resulting tabular display that R will provide?

5. Attach the MCMC simulations to the R workspace by typing

```
attach.bugs(model)
```

This makes all the elements of the object `model` available in your R session.
Plot a histogram of the posterior distribution for `theta` by typing

```
hist(theta)
```

What can you say about the underlying probability of a female birth?

MCMC in R/JAGS

If you want to use JAGS instead of BUGS, there are not many difference with respect to the previous code/analysis. You can use these guidelines to perform the analyses in the practicals using JAGS .

The first thing to do is to load the package R2jags, which you do by typing in your R terminal the command `library(R2jags)`. You can now replicate steps 1. and 2. from the previous section to write your model and prepare the data and relevant variables. Notice, however, that while most of the time the BUGS code will apply with no problems to JAGS, there are a few differences that may prevent your model code from working. One such example is the way in which BUGS and JAGS manage truncation and censoring (see, for instance page 205 of BMHE).

The first change is that (unsurprisingly!) you need to call the function `jags`, instead of the function `bugs`. Thus, point 3. above becomes

```
model2 <- jags(data=data,inits=inits_det,
          parameters.to.save=params,model.file=filein,
          n.chains=2,n.iter=10000,n.burnin=4500,
          n.thin=1,DIC=TRUE)
```

This time, JAGS is called in background and performs the MCMC estimation. By default, R2jags shows a text bar with the progression through the required simulations; a running series of asterisks is printed and the counter is incremented while the iterations are generated.

The second and perhaps most important difference is in the nature of the objects created. If you type in your R terminal the command `names(model2)`, R will show you the names of the elements of the object `model2`

```
[1] "model"              "BUGSoutput"         "parameters.to.save"
[4] "model.file"         "n.iter"             "DIC"
```

You can access each of these elements using the syntax `object$element`, so for example R will respond to the command `model2$n.iter` by printing the value for the number of iterations that have been used (in this case 10000).

If, for instance, you had run the steps described in the previous section and had produced the BUGS object `model`, R reponse to a command `names(model)` would show different elements. In fact, the difference is that JAGS stores *these* elements in the object `model2$BUGSoutput`; thus typing `names(model2$BUGSOutput)` gives the following output

```
[1] "n.chains"        "n.iter"          "n.burnin"        "n.thin"
[5] "n.keep"          "n.sims"          "sims.array"      "sims.list"
[9] "sims.matrix"     "summary"         "mean"            "sd"
[13] "median"          "root.short"      "long.short"      "dimension.short"
[17] "indexes.short"   "last.values"     "program"         "model.file"
[21] "isDIC"           "DICbyR"          "pD"              "DIC"
```

which is basically the same as that produced by a call to BUGS .

The third basic difference is that if you want to make the results of your simulations available to the R workspace, you need to use the command `attach.jags(model2)` (or whatever the name of the object created with the call to the JAGS function).

Health economic evaluation in R using BCEA

Suppose the interest is in an infectious disease, e.g. influenza, for which a new vaccine has been produced. Under the current management of the disease some individuals treat the infection by taking over the counter (OTC) medications. Some subjects visit their GP and, depending on the severity of the infection, may receive treatment with antiviral drugs, which usually cures the infection. However, in some cases complications may occur. Minor complications will need a second GP visit after which the patients become more likely to receive antiviral treatment. Major complications are represented by pneumonia and can result in hospitalisation and possibly death. In this scenario, the costs generated by the management of the disease are represented by OTC medications, GP visits, the prescription of antiviral drugs, hospital episodes and indirect costs such as time off work. QALYs are used as a measure of clinical benefit. The focus is on the clinical and economic evaluation of the policy that makes the vaccine available to those who wish to use it ($t = 1$) against the null option ($t = 0$) under which the vaccine will remain unavailable.

- The file `Vaccine.RData` contains the results of a economic model, which are saved in an object `he`.

  NB: To download the file, right-click on the link and select "Save Link As…"

  From your `R` terminal, load the data and examine the object `he` using the commands

  ```
  load("Vaccine.RData")
  names(he)
  ```

  Load `BCEA`by typing the command

  ```
  library(BCEA)
  ```

  and compute the ICER using the elements `delta.e` and `delta.c`, which are included in the object `he` and contain the simulations from the posterior distributions of the variables (Δ_e, Δ_c) (i.e. the effectiveness and cost differential, respectively), obtained by running a suitable Bayesian model.

  Hint: the command `a = b / c` instructs `R` to compute a variable `a` defined as the ratio between the variables `b` and `c`; similarly, the command `d = mean(e)` saves the mean of the values contained in a vector `e` to a new variable `d`.

- Assuming a fixed value of the willingness-to-pay threshold of $k = 30\,000$, compute the value of the EIB.

  Hint: consider the definition of EIB and use the values for `delta.e` and `delta.c`. In `R`, the commands `a = b + c` and `d = e - f` define variables `a` and `d` as the sum and the difference of other variables, previously defined in the workspace.

- In the `R` terminal, execute the command

  ```
  ceplane.plot(he)
  ```

  to produce the cost-effectiveness plane for the comparison of $t = 1$ vs $t = 0$, using a default value of $k = 25\,000$. How can you interpret the resulting graph, in terms of economic evaluation?
- In the `R` terminal, execute the command

  ```
  ceplane.plot(he,wtp=10000)
  ```

  to produce the cost-effectiveness plane for the comparison of $t = 1$ vs $t = 0$, using a value of $k = 10\,000$. How is the economic interpretation of the results modified, in comparison to point c?
- In the `R` terminal, execute the command

  ```
  eib.plot(he, plot.cri=FALSE)
  ```

  to obtain a graph showing the EIB as a function of the willingness-to-pay. (Notice that you need to add the option `plot.cri=FALSE`, which prevents `BCEA` from drawing a credible interval around the EIB, in this particular case. The reason is that the `BCEA` object `he` has been tampered with for the purpose of running this exercise. In general, you do not need to specify the option and by default `BCEA` will also show the credible interval).
  How can you interpret this graph in terms of cost-effectiveness analysis?
- In the `R` terminal, execute the command

  ```
  contour(he)
  ```

  to produce the cost-effectiveness plane for the comparison of $t = 1$ vs $t = 0$ together with a contour plot of the bivariate distribution for (Δ_e, Δ_c). The graph also reports the proportion of simulated points in each quadrant. Comment on the cost-effectiveness results. What is the probability that vaccination ($t = 1$) is dominated by the status quo ($t = 0$)?