Presenting Simulation Results

Advanced Course Module 3

Nichola Naylor & Jack Williams

14 April 2021

Overview

There are 2 parts to this exercise:The purpose of this exercise is to show how R can be used to run simulations drawing from the distributions chosen for the previously constructed probabilistic model of THR (Advanced Course Module 2 Exercise) using functions to record the results.

PART 1: We will be using the "A3.3.2a" template and "A3.3.3a" solution files. The step-by-step guide covers five main areas:

1. Creating the model function
2. Running the simulations and storing the results
3. Analysing results: the CE plane
4. Another function for calculating CEACs
5. Multiple curves: different patient sub-groups

PART 2: We will be using the "A3.3.2b" template and "A3.3.3b" solution files. The purpose of this exercise is to introduce a further prosthesis into the THR model in order to illustrate the approach to calculating multiple CEACs for multiple treatment options.The step-by-step guide covers two main tasks:

1. Adding in a third prosthesis option
2. Multiple curves for mutually exclusive treatment options.

PART 1: Step-by-step guide

(1) Creating the model function

In the relevant template file you will find a new section . The aim is to record repeated results from the probabilistic analysis constructed in Module 2 and to analyse the results. The 'THR Model with simulation' (an additional resource from Module 2) just created a function that essentially re-ran the whole of the model built in Module 2 1,000 times. In this exercise we will aim to structure the model function to be more efficient (i.e. run through the code at a faster speed). We've started off by loading the packages and data files needed.

- Assign all the model parameters that are fixed throughout the model runs (i.e. those that aren't sampled in the probabilistic analyses) outside of the model function. We've started this process for you by naming the parameters that need defining, such as discount rate for costs. We then leave it up to you to define any other parameters that could be defined outside of the model function. *Hint: Go through the Module 2 solutions, to find the value assignments that aren't probabilistic, including alpha and beta value assignments (be careful not to miss out important parameters!)*

- Now code a function model.THR() that can be run across a probabilistic THR model. Assume, for now, that we are always running across the same sex (female) and age (60) groups for now, we will alter this to allow for varying age/sex groups later on in this exercise. This function should return a named vector (this should include discounted costs and effects of the standard treatment, the discounted costs and effects of new treatment, and the incremental costs and effects). We have layed out the structure of this function, with gaps that need filling before running (based on the work you did for Module 2, you may copy and paste parts of the code from the Module 2 Exercise R script into this one, but be sure to define all variables needed/used within the model and try to understand what each part is doing).
- Try running model.THR(age=60, male=0) to check it runs before moving to the next stage. This may still take a fair few seconds to run so be patient on first running. *Top tip: If you want to debug your function, after defining the deterministic parameter, set male <- 0 and female <- 0 and work through chunks of the function code to see where an error occurs*

It should be noted that you can have functions stored in a separate R script within your working directory and run this into your simulation analysis script through using the source() function [check ?source for more information]. However, for the simplicity of teaching this course this was not done here.

(2) **Running the simulation** Remember that we are still running the model only for females aged 60 currently (we will move on-to patient subgroups later in this exercise). Create the following first: (i) A 'sim.runs' variable that stores the number of runs we want (1,000 in this case) (ii) A data.frame with 6 columns (for costs and effects of standard treatment, for costs and the effects of new treatment, and incremental costs and effects - name the columns appropriately) that has the same number of rows as specified in sim.runs. Values can be set to NA for now, but set the class to numeric (this can be done through using 'is.numeric(NA)'). You should also use the rep() function. We've laid out the structure for you to fill.

Then run the simulations in a loop, and replace the blank with the results from 1,000 runs of the model.THR() function.
(3)**Creating the Cost-Effectiveness Plane** Now that we have stored the results of 1000 probabilistic trials of the model, the results can be analysed.

(i) Using a simple plot() function plot incremental QALYs against incremental costs, with incremental qalys on the x-axis and incremental cost on the y-axis.

Next, we have used the ggplot2 package to create more readable plots within the [find an available ggplot2-cheatsheet here; https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf].

(ii) Load in the "ggplot_CEA_functions.R" script within the "additional_resources" folder and run the ce.plane() function we've written out for you.

Notice the slightly strange shape of the 'cloud' of points. This is due to the non-linearity relating input and output parameters inherent in the model structure. As a consequence, the point estimates from the deterministic model (evaluated at the means of the distributions) are not the same as the expectations across the output parameters (which can be estimated by averaging the cost and QALY value estimates for each arm and estimating the ICER). Try calculating the average values accordingly to estimate the expected average value from the THR probabilistic model. How does this compare to the deterministic ICER of £2,918 per QALY gained ?

(4)**Creating the Cost-Effectiveness Acceptability CUrve**

Once probabilistic results are obtained as in the form of step (i) above, the analysis of the results can proceed much like the analysis of bootstrap replications of real data in a statistical analysis (Briggs & Fenn 1998). This includes the potential for calculating confidence intervals for the ICER (using, for example, simple percentile methods) or cost-effectiveness acceptability curves (CEACs). We focus on the calculation of CEACs as the appropriate way of presenting uncertainty in CEA as a method that avoids the problems that can arise when uncertainty covers all four quadrants of the CE plane.

The calculation of CEACs will involve another function (p.CE). However, some preparation of the section is required first. In particular, the presentation mentioned that two alternative approaches to the use of the net benefit statistic could be used to generate acceptability curves. This exercise will involve calculating both in order to convince you that the methods are equivalent.

In the section we will first demonstrate the concept of estimating the average net monetary benefit of introducing the new prosthesis, relative to standard treatment.

- First define the 'ceiling ratio' (i.e. the maximum acceptable willingness to pay for health gain) to be £100,000.

- Use this value to calculate the average net monetary benefit for the standard and new prostheses respectively based on the average results you obtained in estimating the expected average incremental costs and effects of new treatment (PSA.inc.cost and PSA.inc.qalys).
  Based on this estimate would you accept or reject the new intervention (in terms of it's net monetary benefit impact)? What is the decision rule you use for this?

- Note there are a few ways this can be coded but the aim is to output 1 numeric value of how often the new treatment would be deemed cost-effective over the 1,000 runs with a given WTP. Based on the above concepts, write a function that calculates for each simulation run the net monetary benefit and whether that is considered cost-effective or not (based on the decision rule with net-monetary benefit) and then calculates the average number of times the new treatment was cost-effective. In the function, allow the willingness-to-pay (WTP) input of the function - you can do this by specifying inputs within the brackets (e.g. p.CE <- function(WTP, simulation.results) { }). Then, generate the cost-effectiveness acceptability curve table by using a for loop and the above function to estimate the probability of cost-effectiveness for different WTP values ranging from 0 to £50,000 (by increments of 10). This can be done by first creating a data.frame with 2 columns; 1 with the WTP values and 1 with the probability of cost-effectiveness (filling the latter with numeric,missing values in the first instance, then replacing with outputs of p.CE() using a for loop).

- Finally, use the plot() function to simply plot WTP values against probability of cost-effectiveness (Hint: use "type="l").

- We've used the pre-defined function plot.ceac (from"ggplot_CEA_functions.R" script which should already be loaded into your work environment, see source("A0.2_R_Starting Material_for_Advanced_Course/ggplot_CEA_functions.R")).

(5)**Multiple curves: different patient sub-groups**

Up to this point we have been working with a model where the patient characteristics have been set for a female aged 60 years. In this part of the exercise, we want you to repeatedly obtain results for Men and Women aged 40, 60 and 80 years and record the results and present them as multiple acceptability curves.

- Create an array that can store the simulation results for each subgroup defined above. Think about the output of model.THR() in terms of size and shape.
- Run the function on each of the subgroups, storing the results within the array. You may want to have larger increments in the WTP values this time to help with speed. Create a data.frame (CEAC.subgroups) with nrow=WTP values and ncol= number of subgroups + 1 column to record the probability of cost-effectiveness result.

To use the plotting functions we need to make the CEAC subgroup data.frame in a long format:

WIDE FORM EXAMPLE: (note this is just an example and doesn't inclue all subgroups used in this exercise)

```
##   WTP Male_40 Male_60 Female_40 Female_60
## 1   0   0.992   0.260     0.820     0.018
## 2  50   0.994   0.280     0.841     0.019
## 3 100   0.996   0.301     0.866     0.020
## 4 150   0.997   0.314     0.895     0.024
```

LONG FORM EXAMPLE:

```
##   WTP subgroup   pCE
## 1   0  Male_40 0.992
## 2  50  Male_40 0.994
## 3 100  Male_40 0.996
## 4 150  Male_40 0.997
```

We can do this using the melt() function within the reshape2 package (which should have been loaded in at the start of this exercise using library(reshape2), if not go back to the beginning of this script and load this in). Type in ?melt.data.frame into the console to see how to use this function.

Finally, we've used plot the resulting CEACs for the different patient characteristics using the pre-defined ggplot function plot.ceac.all().

You should find that the curves can be quite different for different patient characteristics. For example, the new prosthesis seems to be more cost-effective generally for men (due to the fact that men have higher failure rates on average and so the absolute benefits of reducing failure risk are greater) and that the new prosthesis is not so cost-effective for elderly patients (where death is an important competing risk for prosthesis failure).