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

This will require the following data files to be located in the same working-directory folder:

- hazardfunction.csv
- cov55.csv
- life-table.csv

And will require the following packages to be installed:

- data.table
- tidyr
- dplyr
- ggplot2
- reshape2

The final 2 packages mentioned above are needed as we have added some additional code at the bottom of the template and solution files that use the `reshape` and `ggplot2` packages to show you other potential ways to plot the simulation results. 'ggplot' is a graphics package that allows you to make very good graphs and plots, but requires the data to be structured in a specific way (long format), and tends to require more detailed code (which is why it allows for highly detailed plots). For more information on ggplot2, see this cheat sheet: https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf.

- Please ensure you have the "ggplot_CEA_functions.R" script stored in the same folder as your template/solution files (with the working directory set to that same folder)

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.

All of R packages listed above for Part 1 are also needed for Part 2. Additionally the following data files are needed:

- hazardfunction_NP2.csv : Note this is different to the "hazardfunction.csv" listed for Part 1, as this now gives data on an additional treatment pathway (NP2).
- cov55_NP2.csv : Note this is different to the "cov55.csv" listed for Part 1, as this now gives data on an additional treatment pathway (NP2).
- life-tables.csv : this is the same as used in Part 1.

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 labelled "A2.3.4_Additional_Information_THR_Model_with_simulation.R") 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, and allow for characteristics within the model to be easily changed whilst using the model function.

- First, 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 outlined in the template, 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. You should have similar results to the below (remember actual numbers will differ):

```
##     cost.SP0    qalys.SP0     cost.NP1    qalys.NP1     inc.cost    inc.qalys
## 552.95151080  14.94816867 652.42463948  14.97956472  99.47312868   0.03139605
```

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 the 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 `as.numeric(NA)`). You should also use the `rep()` function. We've laid out the structure for you to fill. The first 6 rows of `sim.runs` is shown below:

```
##   cost.SP0 qalys.SP0 cost.NP1 qalys.NP1 inc.cost inc.qalys
## 1       NA        NA       NA        NA       NA        NA
## 2       NA        NA       NA        NA       NA        NA
## 3       NA        NA       NA        NA       NA        NA
## 4       NA        NA       NA        NA       NA        NA
## 5       NA        NA       NA        NA       NA        NA
## 6       NA        NA       NA        NA       NA        NA
```

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.

(ii) Load in the "`ggplot_CEA_functions.R`" script within the "A0.2_R_Starting Material_for_Advanced_Course" 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
```

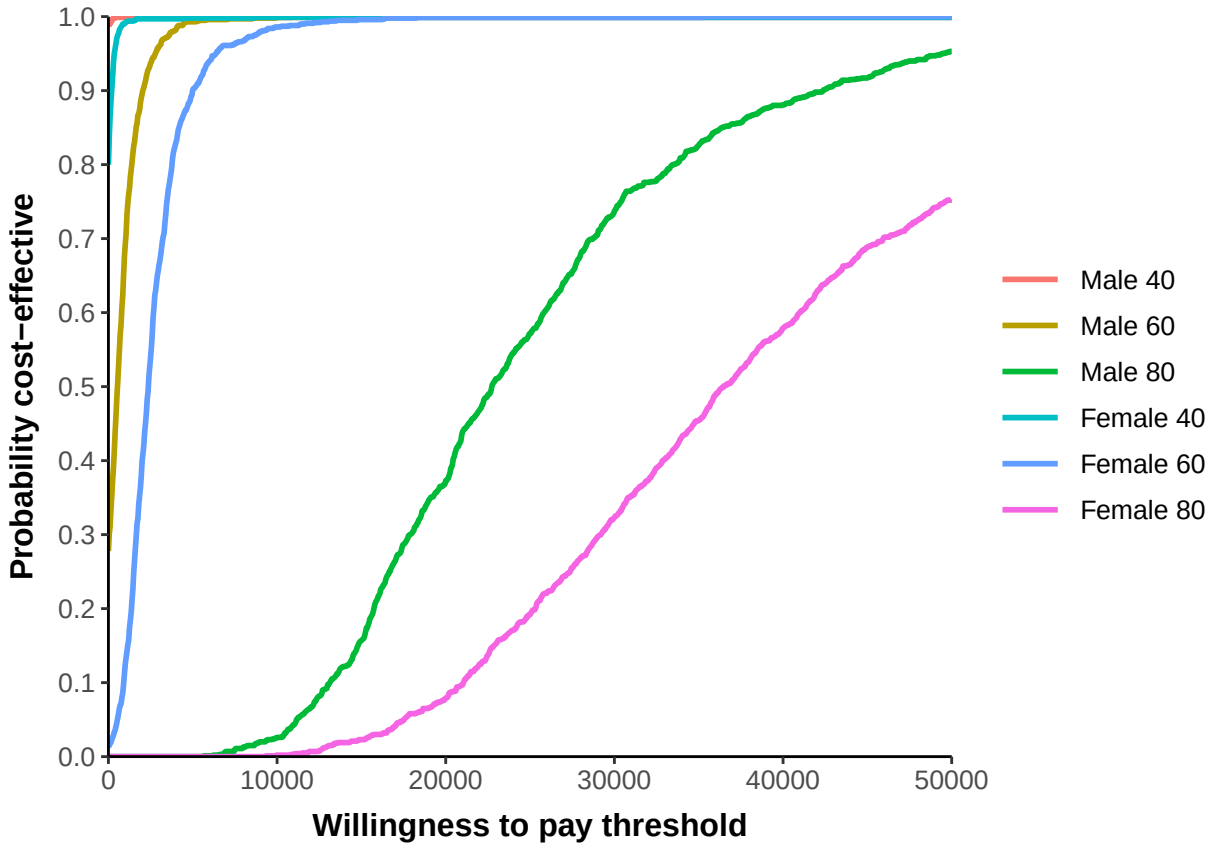
LONG FORM EXAMPLE:

```
##   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 (your final graph for Part 1 should look similar to the below). 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).



PART 2: Step-by-step guide

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 template for this exercise is 'A3.3.2b_Presenting_Simulation_Results_Part2_Template.R.R'.

This step-by-step guide covers two main tasks 1. Adding in a third prosthesis option to the function 2. Running a new round of simulations and storing these results 3. Creating multiple curves for mutually exclusive treatment options.

(1) Adding in a third prosthesis option

Although in the vast majority of trials just two treatment alternatives are compared, in modelling it is much more common to look at multiple treatment options. If you open A3.3.2b_Presenting_Simulation_Results_Part2_Template.R you will find that the read in for "cov55.csv" has been replaced by "cov55_NP2.csv", and "hazardfunction.csv" by "hazardfunction_NP2.csv". Take a moment to examine the new `hazards` and `cov.55` data, which contains additional information concerning this prosthesis in terms of its effectiveness and correlation with other parameters.

The aim in this section is to update the model so that the third prosthesis option is fully integrated. This involves updating the parameter, analysis and simulation sheets before re-running the probabilistic sensitivity analysis.

We start by defining the deterministic parameters that don't update for each run (refer to the work done in Part 1, this can be copied over). The main difference here is we need to add the the new treatment pathway

(NP2) to the Cholesky decomposition, so make sure this is updated from Part 1 (see template document for an outline of what to add).

We now need to set up a new function `model.THR.3b()`. Use `model.THR()` as the base and update/create the following objects within the new function (`model.THR.3b()`): * Relative risk of revision for new prosthesis 2 compared to standard: `r.NP2` and `RR.NP2` (within the hazard function part of the function) and `revision.risk.np2` (within the Markov model section, referring to the time dependent risk of revision for NP2) * Create transition and trace matrices for NP2 (`tm.NP2` and `trace.NP2`) * Estimate the discounted cost (`disc.cost.NP2`) and QALY (`disc.QALYs.NP2`) outcomes from NP2, using the code provided for NP1 as a guide. * An updated `output` matrix that includes estimated cost and qalys for NP2.

Hint: It might be worth testing each stage of the model by defining `age` *(e.g.* `age <- 60`*) and* `male` *(e.g.* `male <- 0`*) and work through each section of the function before running the whole thing. If you do this, the first 6 rows of the updated revision.risk.NP2 column in tdtps should look like the following:*

```
## [1] 4.929054e-05 9.052519e-05 1.174737e-04 1.393052e-04 1.581746e-04
## [6] 1.750431e-04
```

(2) Running and Storing Simulation Results

Now that the adaptations relating to the third option are complete the updated results can be produced. We can now store the simulation results, similar to the end of Part 1 of this exercise, work your way through the section of the template file. Be patient, it might take a few seconds for your simulations to run. Your simulation.results should have the same structure as the below:

```
##   cost.SP0 qalys.SP0 cost.NP1 qalys.NP1 cost.NP2 qalys.NP2
## 1 490.4874  15.17128 596.7246  15.21552 807.8559  15.21432
## 2 465.6587  13.32369 611.6788  13.33677 818.1266  13.33762
## 3 467.5967  14.66657 613.0696  14.69761 800.6712  14.71444
## 4 495.6129  14.81404 615.1812  14.84050 810.2097  14.84616
## 5 462.8061  14.45422 596.8012  14.49133 793.9923  14.49995
## 6 536.0078  14.50180 618.1423  14.53422 816.0646  14.53772
```

Note: in the solution file we've begun to add in code that gives you information on the progress of your simulation runs. You don't need to include it in your template file for this exercise, but take a look at how we've implemented it, as we will start to use it more in the next exercise.

We can calculate the mean results across the simulations and use these values to update the point estimates for the probabilistic analysis on the section. Use the `apply()` function to calculate these averages. *Hint: you want to use the* `mean` *function over columns, enter* `?apply` *into the console to see more information on this function*

(3) Multiple curves for mutually exclusive treatment options

Now that you've ran the simulations and stored the results, we can begin to plot the simulation results graphically. First let's plot the cost-effectiveness plane, go to the section. We've added the code in for you, run and take a look.

Note that one of the problems of looking at the three clouds of points on the CE plane is that we lose the perception of how the points are correlated between the three options.

The results from the probabilistic analysis of mutually exclusive options can be presented as multiple acceptability curves. However, in contrast to the independent sub-group analysis of the first part of this exercise, these multiple curves must sum to one. We must therefore repeat the analysis based on net-benefit, but this time for three options, which is where the use of average net-benefit becomes useful.

(i) Calculate the average net monetary benefit for each option by first defining a vector of WTP values from £0 to £50,000 per QALY gained, by increments of your chosing (`WTP.values`). Then create a data.frame (`CEAC`) which has the WTP values and empty columns for standard, NP1 and NP2 columns.

(ii) Now finish writing the function (`pCE.3b`), we've left a few definitions blank for you to complete. The aim of the function is to estimates the probability of the new intervention being cost-effective. You will need to use the `apply()` function to first select the maximum nmb value by row (`max.nmb`) and then to calculate the mean number of times that each treatment pathway was found to be the most cost-effective (i.e. the mean times each column within the `CE` data.frame was found to be the strategy with the maximum NMB).

(iii) Complete the for loop to fill in the CEAC table with the outputs of the `pCE.3b` function.

After running the for loop, you should have a CEAC with a 'head' and 'tail' similar to the below (differing values are to be expected, the main thing is to have a similar structure filled with numerics and a similar pattern - i.e. a shift from Standard being the most cost-effective to the NP1/NP2 pathways being more cost-effective as we move from lower WTP values to higher WTP values).

```
head(CEAC)
```

```
##   WTP Standard   NP1 NP2
## 1   0    0.988 0.012   0
## 2  10    0.988 0.012   0
## 3  20    0.988 0.012   0
## 4  30    0.987 0.013   0
## 5  40    0.987 0.013   0
## 6  50    0.987 0.013   0
```

```
tail(CEAC)
```

```
##        WTP Standard   NP1   NP2
## 4996 49950        0 0.523 0.477
## 4997 49960        0 0.523 0.477
## 4998 49970        0 0.523 0.477
## 4999 49980        0 0.523 0.477
## 5000 49990        0 0.523 0.477
## 5001 50000        0 0.523 0.477
```

Now run the final lines of code that uses our pre-defined `plot.ceac.all()` from the "ggplot_CEA_functions.R" script, first reshaping the data to match the format needed.

As was noted in the presentation, the interpretation of multiple acceptability curves can be tricky – especially when the model is non-linear. Fenwick et al (2001) have argued for the use of a cost-effectiveness acceptability frontier in these situations and an example of this applied to the Excel version of the model you have just constructed is available on the course website.