Decision Modelling for Health Economic Evaluation R Stream Reader - Advanced Course

Nichola Naylor & Jack Williams

2022 Course

Introduction

Welcome to the R Stream of the Decision Modelling for Health Economic Evaluation - Advanced Course!

Over the course of the next few modules we will cover how to code Advanced Markov Models (Module 1), Making Models Probabilistic (Module 2), Presenting Simulation Results (Module 3) and Value of Information Analyses (Module 4).

This course stream is not intended to teach you how to use R, but rather to teach potential ways of constructing health economic evaluation models within R. For more information on how to use R seek readily available resources such as:

- Free R books: http://www.cookbook-r.com
- Free forums online can solve *almost* any problem: http://www.stackoverflow.com

There are often many ways to do the same thing in R. Throughout this course we present one potential way of coding things, often opting for code which is easier to follow (for example, choosing a loop process over something more efficient but less intuitive), and tend to use base R functions wherever possible. This is not to say another way is "wrong" or that the solutions provided are the only way to code these models. We encourage you to play around with the code, checking whether the results align across the different methods. In some instances, we have also provided two different lines of code to produce the same output, to demonstrate the different approaches available.

R Exercise Structure

Each Module has the following:

- An instruction pdf (e.g. "A1.3.1_Advanced_Markov_Modelling_Instructions.pdf" which refers to Advanced Module 1, R stream, File 1): This provides a step-by-step guide for each exercise, providing important information on parameters and hints & tips on how to code your models. This should be used in conjunction with…
- A template R script (e.g. "A1.3.2_Advanced_Markov_Modelling_Template.R" which refers to Advanced Module 1, R stream, File 2): This provides an outline of the model code, with key variables, functions and/or loops to define - there are blank spaces to fill whilst following the Instructions pdf.
- A solutions R script (e.g. "A1.3.1_Advanced_Markov_Modelling_Solutions.pdf" which refers to Advanced Module 2, R stream, File 3) which provides a completed model code script, filled in for you. If you run this R script (having in mind the section below about directory set up), this will provide the fully functional models and results.

For Modules 3 and 4, the Template and Solution files are split into 'Part 1' and 'Part 2', and the Instructions pdf will walk you through when to use which.

We mention 'sections' within the R scripts (such as the 'Parameters' section) for this course, similar to how you might refer to sheets in Excel. If you are using RStudio, sections can be selected using the bottom left of the Source panel or alternatively using the document outline functionality (top right of the Source panel).

Additional Files and Scripts for the Advanced Course

For those of you that completed the Foundations course, everything previously was defined within each script, and so each script could be run on its own. However, for the Advanced course we have additional (i) data files that need to be loaded for the template and/or solution files to run and (ii) additional R scripts that are sourced within the template and/or solution files to use pre-written functions (such as plotting functions). These include:

(i) Data files:

1. "cov55.csv"
2. "cov55_NP2.csv"
3. "hazardfunction.csv"
4. "hazardfunction_NP2.csv"
5. "life-table.csv"

(ii) Additional R Scripts:

1. "A2.3.4_Additional_Information_THR_Model_with_simulation.R"
2. "A4.3.3b_VoI_Model_Script.R"
3. "ggplot_CEA_functions.R"
4. "cholesky_decomposition.R" (this final one is not necessary for read ins but useful for understanding).

Throughout the template and solution files csv files will be read in using the `read.csv()` function. For more information on this enter `?read.csv` into the console.For example:

```r
hazards <- read.csv("hazardfunction.csv", header=TRUE)
head(hazards)
```

```
##   explanatory.variables coefficient       se hazard.ratio
## 1               lngamma    0.374097 0.047450     1.453678
## 2                  cons   -5.490935 0.207892     0.004124
## 3                   age   -0.036702 0.005211     0.963963
## 4                  male    0.768536 0.109066     2.156607
## 5                   NP1   -1.344474 0.382582     0.260677
```

Throughout the template and solution files additional R script files will be read in using the `source()` function. For more information on this enter `?source` into the console.For example:

```r
source("ggplot_CEA_functions.R", local = knitr::knit_global())
```

```
## Loading required package: ggplot2
```

```r
## by sourcing in the file we can see and use previously defined functions, such as:

print(plot.ce.plane)
```

```
## function (results)
## {
##     xlabel = "Incremental QALYs"
##     ylabel = "Incremental costs"
##     plot = ggplot(results) + geom_point(shape = 21, size = 2,
##         colour = "black", fill = NA, alpha = 0.5, aes(x = inc.qalys,
##             y = inc.cost)) + labs(x = xlabel, text = element_text(size = 10)) +
##         labs(y = ylabel, text = element_text(size = 10)) + theme_classic() +
##         theme(legend.title = element_blank(), axis.title = element_text(face = "bold"),
##             axis.title.x = element_text(margin = margin(t = 7,
##                 r = 0, b = 3, l = 0)), axis.title.y = element_text(margin = margin(t = 0,
##                 r = 7, b = 0, l = 3)), panel.grid.major = element_blank(),
##             panel.grid.minor = element_blank(), legend.key.width = unit(1.8,
##                 "line"), text = element_text(size = 12), plot.margin = unit(c(1.2,
##                 0.5, 0, 1.2), "cm"))
##     return(plot)
## }
```

```r
## Don't worry you don't need to know what that function actually means yet!
```

Please check you have downloaded these additional files before attempting the exercises. Do not rename these files. We will now highlight how these files need to be stored in the following section…

Setting up your working directory

Once all of the above files have been downloaded from the course website, please make sure that you have a dedicated folder for the course code and data. This folder should become your working directory for the Advanced course.

If you want to manually set the working directory, then you can use the setwd() function, or alternatively, within RStudio use the “Set Working Directory” functionality available through the “Session” Tab.

It’s key that all of the scripts **and** the data files are all stored within the same folder (not in subfolders), and that folder is the working directory for all of your analyses.

Installing packages

Although within the course we try to mostly use base R functions, we will need to use additional packages for some exercises.

The following packages are used in the course and can be installed using the following code.

```r
install.packages("tidyr")
install.packages("dplyr")
install.packages("ggplot2")
install.packages("reshape2")
```

Some R Reminders

We will use many base R functions throughout the course. You can see a simple cheat sheet available under the “Base R” category within the RStudio cheat sheet repository: https://www.rstudio.com/resources/cheatsheets/

Key terms to be familiar with before starting: vectors, matrices, functions, loops.

Vector

```r
(vector <- c(1, 2, 3))
```

```
## [1] 1 2 3
```

Matrix

```r
matrix(1:9, nrow = 3, ncol = 3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

Array (a multi-dimensional matrix) - the dim (dimensions) aspects indicates rows, columns, matrix numbers

```r
array(1:18, dim = c(3, 3, 2)) ## dimensions = c(nrow, ncolumns, number of matrices in the 3D space)
```

```
## , , 1
##
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
##
## , , 2
##
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
```

```
array(1:6, dim = c(1, 2, 3))
```

```
## , , 1
##
##      [,1] [,2]
## [1,]    1    2
##
## , , 2
##
##      [,1] [,2]
## [1,]    3    4
##
## , , 3
##
##      [,1] [,2]
## [1,]    5    6
```

Functions

```
print.greeting <- function(x){
  if(x>=2){print("hello")}
  else{print("goodbye")}
}
print.greeting(3)
```

```
## [1] "hello"
```

```
print.greeting(1)
```

```
## [1] "goodbye"
```

For Loop

```
x <- c(1:5)
print(x)
```

```
## [1] 1 2 3 4 5
```

```
for(i in 1:5){
  x[i] <- x[i]*2
}
print(x)
```

```
## [1]  2  4  6  8 10
```

Subsetting and selecting data

Another important aspect of the course is selecting the appropriate data. For the various vectors, data frames and arrays, here are some ways to select data:

Vector

```
(vector <- c(10, 20, 30, 40))
```

```
## [1] 10 20 30 40
```

```
vector[2] # Takes the second element of the vector
```

```
## [1] 20
```

```
vector[3:4] # Takes the third and fourth element of the vector
```

```
## [1] 30 40
```

Matrix

```
x <- matrix(1:9, nrow = 3, ncol = 3)
x
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
## we can provide column and row names to matrices:
colnames(x) <- c("a","b","c")
rownames(x) <- c("d","e","f")
x
```

```
##   a b c
## d 1 4 7
## e 2 5 8
## f 3 6 9
```

```
## these can be used to call values
x["d","a"] ## remember: rowname first, then colname
```

```
## [1] 1
```

Data frame

```
df <- data.frame(cost=1:5, effect=100:104)
print(df)
```

```
##   cost effect
## 1    1    100
## 2    2    101
## 3    3    102
## 4    4    103
## 5    5    104
```

```
df[3,] # Third row (all columns)
```

```
##   cost effect
## 3    3    102
```

```
df[,2] # Second column (all rows)
```

```
## [1] 100 101 102 103 104
```

```
df[3,2]  # Third row and second column value
```

```
## [1] 102
```

```
df$cost # We can also call columns by their column name
```

```
## [1] 1 2 3 4 5
```

```
df[1,"cost"] ## 1st row, cost value
```

```
## [1] 1
```

Array

```
ar <- array(1:18, dim = c(3, 3, 2))
print(ar)
```

```
## , , 1
##
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
##
## , , 2
##
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
```

```
ar[1,,2] # first row, of second matrix
```

```
## [1] 10 13 16
```

```
ar[,,1] # All rows, all columns, first matrix
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

Matrix Multiplication

Throughout this course matrix multiplication will play an important role in our models. Below is a reminder of some useful information on matrix multiplication in R. (Some code based on https://www.programmingr.com/matrix-multiplication/, last accessed 19/05/2021)

```
a = matrix(c(1,3,5,7), ncol=2, nrow=2)
a
```

```
##      [,1] [,2]
## [1,]    1    5
## [2,]    3    7
```

```
b = matrix(c(2,4,6,8), ncol=2, nrow=2)
b
```

```
##      [,1] [,2]
## [1,]    2    6
## [2,]    4    8
```

```
c <- a*b ## this just multiplied values by the same positioned-value
## c[2,1] = a[2,1] * b[2,1]
c
```

```
##      [,1] [,2]
## [1,]    2   30
## [2,]   12   56
```

```
### However, if we want to perform matrix multiplication to get the matrix product
### we can use the %*% operator in R
c <- a %*% b
## c[2,1] = (a[2,1]*b[1,1])+(a[2,2]*b[1,2])
## c[2,1] = 6 + 28 = 34
c
```

```
##      [,1] [,2]
## [1,]   22   46
## [2,]   34   74
## Notice that the order of the matrices affects the results:
c <- b %*% a
## c[2,1] = (b[2,1]*a[1,1])+(b[2,2]*a[1,2])
## c[2,1] = 4 + 24 = 28
c
```

```
##      [,1] [,2]
## [1,]   20   52
## [2,]   28   76
```

```
## If we have a vector and want to multiply a matrix by the values in the vector
## Again, ordering matters. For example..
## if we want to multiply state costs (vector) by number of people in each state (matrix)

people <- matrix(c(100,0,0,
                   90,10,0,
                   80,10,10,
                   50,30,20), ncol=3, nrow=4, byrow=TRUE,
                 dimnames = list(c("year1","year2","year3","year4"),
                                 c("healthy","sick","verysick") ))
cost <- c(0, 200, 500)

people
```

```
##       healthy sick verysick
## year1     100    0        0
## year2      90   10        0
## year3      80   10       10
## year4      50   30       20
```

```
cost
```

```
## [1]   0 200 500
```

```
result <- people %*% cost
result
```

```
##        [,1]
## year1     0
## year2  2000
## year3  7000
## year4 16000
```

```
## results[1,1] = people[1,1]*cost[1]+people[2,1]*cost[2]+people[3,1]*cost[3]
```

Note that if we tried `cost %*% people` this would result in: `Error in cost %*% people : non-conformable arguments`.

This is because the number of columns/values in cost (n=3) does not equal the number of rows/values in people (n=4). We also wouldn't want the product of this calculation, even if these dimension lengths match, as that would multiply the wrong values together. For example . . .

```
people_3yr <- matrix(c(100,0,0,
                  90,10,0,
                  80,10,10), ncol=3, nrow=3, byrow=TRUE,
                dimnames = list(c("year1","year2","year3"),
                                c("healthy","sick","verysick") ))

wrong_result <- cost %*% people_3yr
wrong_result
```

```
##      healthy sick verysick
## [1,]   58000 7000     5000
```

```
## wrong_result[1,] = (100*0)+(200*90)+(500*80), which is wrongly multiplying healthy
## people in years 2 and 3 with the cost of sick and very sick respectively.
```

Therefore, be sure to be careful when using matrix multiplication throughout these exercises!

Probabilistic modelling and sampling from random draws

In Module 2 we will learn about probabilistic modelling, to account for parameter uncertainty in decision models.

We show some examples of sampling using the normal distribution, beta distribution, and gamma distribution. We run 6 draws of each example below, but this could be set to any number.

```
# Normal distribution with mean of 20 and standard deviation of 4
rnorm(n = 6, mean = 20, sd = 4)
```

```
## [1] 18.41229 13.47159 28.64538 16.49913 24.59113 21.44762
```

```
# Beta distribution of probability of 0.05 (1 event out of 20 subjects)
rbeta(n = 6, shape1 = 1, shape2 = 19)
```

```
## [1] 0.060598762 0.019828316 0.034120086 0.007695745 0.038775762 0.037112765
```

```
# Gamma distribution for mean of 1000 and SE of 200
rgamma(n = 6, shape = 25, scale = 40)
```

```
## [1]  968.3651  776.7968 1026.4392 1144.4084  912.2972  928.2232
```

There will be more information within the course on how to calculate shape and scale parameters for different distributions, this reader is simply to show how to code draws from the samples.

Plots

In the course we will use a mixture of base R plots and plots using a package called `ggplot2`.

The base R plots are simple to create and provide an easy way to look at the data.

`ggplot` is a much more flexible package that has many more options within plots. This allows the user to edit every aspects of the plot to get it exactly how they want it. Whilst this is really helpful for data visualisation and producing great plots, the number of options are extensive and we won't have time to cover them in any detail in this course.

For those interested in learning more about ggplot, there is an excellent chapter on data visualisation chapter in the 'R for Data Science' book, available here: https://r4ds.had.co.nz/data-visualisation.html

For an overview of ggplot codes see the cheat sheet available through the "Data Visualization Cheatsheet" within the RStudio cheat sheet repository: https://www.rstudio.com/resources/cheatsheets/

Other Information

Please do let us know (via the discussion tools available on the course website) whether there are any issues in the R exercise information provided.

For those interested in other approaches, packages and tutorials for health economic decision making online, please see the following resource:

- SHEPRD - Signposting Health Economic Packages in R for Decision Modelling: https://hermes-sheprd.netlify.app/ . This webpage aims to help you navigate you through the forest of R packages that can be used in health economics and signpost you to the most appropriate one, given your intended health economic analysis. *If you have any additional R packages you would like to contribute to the website, instructions of how to do so via Github are available on the homepage.*

Happy coding!