Health Economic Evaluation using Markov Models in R for Microsoft Excel Users: A Tutorial

Nathan Green[1]* (PhD), Felicity Lamrock*[2] (PhD), Nichola Naylor[3,4] (PhD), Jack Williams[4] (PhD), Andrew Briggs[4] (DPhil)

1 Department of Statistical Science, University College London - UK

2 Mathematical Sciences Research Centre, Queen's University Belfast - UK

3 HCAI, Fungal, AMR, AMU and Sepsis Division, UK Health Security Agency, London, UK

4 Department of Health Services Research and Policy, London School of Hygiene & Tropical Medicine - UK

*joint first author

Abstract

A health economic evaluation (HEE) is a comparative analysis of alternative courses of action in terms of both their costs and consequences. A cost-effectiveness analysis is a type of HEE that compares one intervention to one or more alternatives by estimating how much it costs to gain an additional unit of health outcome. Cost-effectiveness analyses are commonly performed using Microsoft (MS) Excel. However, there is current interest in using other software that is better suited to more complex problems, methods, and data, as well as improved reproducibility and transparency. That is, it is increasingly important to be able to repeat an analysis of a particular data set and obtain the same results, and access the analysis and results in a clearly and comprehensively available form as openly available. In this tutorial we provide a step-by-step guide on how to implement a mainstay model of HEE, namely a Markov model, in the statistical programming language R. The adoption of R for the purpose of cost-effectiveness analysis is highly dependent on the health economic modeller's ability to understand, learn, and apply programming-type skills. R is likely less familiar than MS Excel for many modellers and so coding a cost-effectiveness model in R can be a large jump. We describe the technical details from the perspective of a MS Excel user to help bridge the gap between software and reduce the learning curve by providing for the first time side-by-side comparisons of the Markov model example in MS Excel and R.

Keywords: software, education, cost-effectiveness analysis, multi-state model

Key points:

- In health economics modelling larger and more complex problems, methods, and data have made MS Excel less fit for purpose and programming languages such as R more attractive.
- However, the lack of familiarity with R might be impeding a move away from MS Excel.
- Using our technical details, guidance and templates will help bridge the gap transitioning between software and reduce the potentially steep learning curve.

Health Economic Evaluation using Markov Models in R for Microsoft Excel Users

1 Introduction

A health economic evaluation (HEE) is a comparative analysis of alternative courses of action in terms of both their costs and consequences [1]. A cost-effectiveness analysis is a type of HEE that examines and compares both the costs and health outcomes of one intervention to one or more alternatives by estimating how much it costs to gain an additional unit of a health outcome [2].

Cost-effectiveness analyses are very often performed using Microsoft (MS) Excel [3-4], but there are other bespoke HEE or general software available to perform this task, such as TreeAge (TreeAge Software, Inc, Williamstown, MA), WinBUGS [5] or statistical programming languages, such as R [6]. R is increasingly being used for cost-effectiveness modelling [7-8] along with its popular Integrated Development Environment (IDE) RStudio [9]. As a result, specific HEE tools are being written in R [10]. An overview of R functionality that is applicable to HEE is given by Jalal (2017) [11] and several tutorials already exist to help implement common HEE models in R [12-13].

However, the adoption of R for the purpose of cost-effectiveness analysis is highly dependent on the health economic modeller's ability to understand, learn, and apply programming-type skills [8]. For individuals who are normally used to working with MS Excel, coding a cost-effectiveness model in R can be a large jump [14].

This tutorial is designed for those who are familiar with HEEs built in MS Excel but are a beginner at using R. The aim of this tutorial is to expose what has previously been presented at workshops provided by the authors on this topic, while highlighting why this type of modelling using R is beneficial. In this tutorial a Markov model example from MS Excel will be used to demonstrate how to implement it in R. The reader will be guided through the necessary steps for coding in R, comparing with the equivalent MS Excel model. Throughout, base R is used where possible, as dependency on packages (such as *heemod* [15]) has been shown previously to lead to slightly different results when compared directly to equivalent models built in MS Excel [8]. Additionally, writing out necessary functions in base R allows for transparent step-by-step comparisons with MS Excel models and for greater flexibility in adapting the code for future use.

Tips for those new to R coding

- Setting up and the basics: R is a free, open-access software, downloadable from https://cran.r-project.org. There are lots of introductory tutorials for R available, for example see some of the introductory basic R manuals available under Documentation Manuals. (https://cran.r-project.org/manuals.html)

- Health economics: There are many different packages in R - while this tutorial uses base R, some useful packages for HEE can be found here: https://github.com/n8thangreen/health_economics_R_packages; https://hermes-sheprd.netlify.app/

- The standard R assignment operator <- will be used throughout but = would be equivalent.

- R has a comprehensive library of functions for generating random numbers from various statistical distributions. A full list of distributions can be seen by typing ?distribution in the console. When you generate a "random numbers" in R, you are actually generating pseudorandom numbers. These numbers are generated in a sequence with an algorithm that requires a seed to be set to initialize. The set.seed() function can be used when running simulations to ensure all results, figures, etc are reproducible.

- We will make use of existing and user-defined functions in R. MS Excel has many existing functions which can be called from within a cell. The syntax for creating these is similar in R and Visual Basic for Applications (VBA) in MS Excel. For an introduction in R see [16].
- Version control and collaborating: Code sharing and saving work via GitHub is recognised good practice for R users and helps maintain version control. GitHub.com is a popular provider of internet hosting for software development and version control.
- When repositories on GitHub are open source they will commonly have a license regarding how others are free to use, change, and distribute the software [17].

2 A time-dependent Markov model

A Markov model is arguably the most commonly used approach in HEEs. A Markov model is useful for repeated events over time such as modelling disease progression. It can be thought of as a special form of a more general state transition model. Each state is mutually exclusive, discrete and may be an absorbing or transient state [18]. Of particular focus for the following example, transition probabilities in a Markov model can either be fixed (time homogeneous) or depend on time (time inhomogeneous). Furthermore, we will consider a "clock forward" model, which means that time refers to time since entering the initial state. Conversely, in a "clock reset" (a semi-Markov) model, time refers to time since entering the current state, meaning that time resets to 0 each time a patient enters a new state. General details of Markov chains can be found elsewhere [19].

2.1 Real-world example

This tutorial focuses on providing a practical walk through of the MS Excel cost-effectiveness model from Briggs *et al* [20-21] while comparing with an R model equivalent. Code for the R model used in this tutorial can be found on GitHub at the following link https://github.com/Excel-R-tutorials/Markov_Intro. The corresponding original MS Excel model is available open-access and can be downloaded from the following link https://doi.org/10.17037/DATA.00002980 [20]. The model outlined in the MS Excel spreadsheet has been used to illustrate the principles of cost-effectiveness Markov modelling and probabilistic sensitivity analysis (PSA) [21-22]. It has been made available for non-commercial teaching purposes only.

For the purposes of more easily demonstrating the translation from the MS Excel model to R we have made some small changes to how the formulae in the original spreadsheet were written. These changes do not affect the results of the model. The GitHub repository contains both versions and the original version is in the 'original files' folder. From now on reference to the MS Excel model will refer to the latest version. In particular, in the updated version we adopt the @ notation which substitutes the value into a cell from another cell on the same line from a named column. This will be discussed in Section 2.2.

In this tutorial a three-state model is used to describe how patients transition between health states with a chronic disease. Patients begin in an asymptomatic health state, which means that the patient has developed the chronic disease but has no symptoms. Patients can stay in the asymptomatic health state or move to a progressive health state which means that the patient is experiencing symptoms of the disease. Patients can move from the asymptomatic health state to the dead health state at the same rate as the general population without the disease. Patients can stay in the progressive health state or move from the progressive health state to dead, but at an increased risk of death. The dead state is an absorbing state as a patient cannot change from being dead. Briggs *et al* [21] describe the model in more detail. The (**Fig. 1**) shows an example of the model diagram produced using R. Arrows represent possible transitions between health states (represented by circles) after each model cycle. Backward bending arrows returning to the state they left show that it is possible to remain in the same state as the previous cycle. Let us also define the variables used in the model. In terms of transition probabilities, *tpProg* is the probability of disease progression, *tpDn* is the probability of all-cause death, *tpDcm* is the excess probability of death and *effect* is the proportion reduction of the transition probability from asymptomatic to progressive disease due to the drug treatment. The cost of one cycle in the Asymptomatic and Progressive disease state was *cAsymp* and *cProg*, respectively. The unit cost of the drug was cDrug and the one-off cost of a death was *cDeath*. In terms of Quality of Life weight, in the Asymptomatic and Progressive disease states these were *uAsymp* and *uProg*, respectively. The values will be described in the next sections as well as demonstrating how to fully implement this model in R.

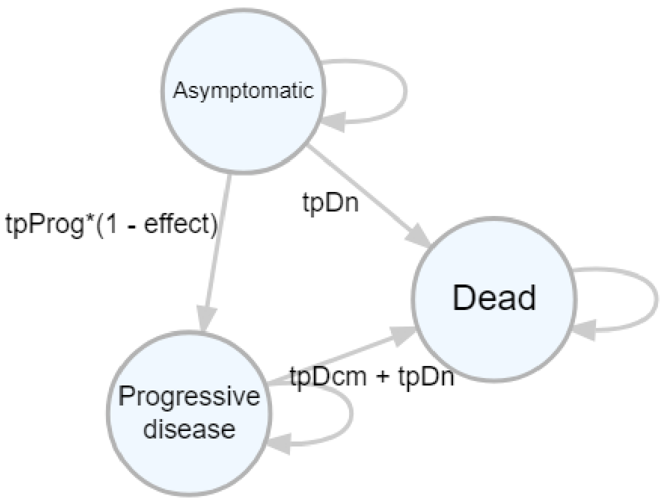

Fig. 1 Markov model diagram created using R. Nodes represent health states and directed edges represent transitions between states. Transition probabilities are shown on the edges. *tpProg* probability of disease progression, *tpDn* probability of all-cause death, *tpDcm* excess probability of death, *effect* effectiveness of drug treatment.

Markov model diagrams in R

There are many packages available in R for plotting edge-node type graphs, including *igraph* [29], *diagram* [30] and *DiagrammeR* [31]. More specific packages such as *markovchain* [32] and *heemod* [15] are used for Markov chain modelling. They build on these base packages and do the work of translating the model object into the graph.

In MS Excel the model figure is usually created by-hand and copy-and-pasted in a sheet as a picture. An alternative in R is to use packages to create a plot directly from the model code. This has the added advantage of literate programming, where the documentation and code are in the same place and the plot is always up to date with any changes to the model. Example R code to make the model diagram using *DiagrammeR* is available in **Online Resource 1** and this tutorial paper GitHub repository**.**

2.2 MS Excel implementation

The MS Excel cost-effectiveness model from Briggs *et al* [20-21] consists of several separate tabs each containing input parameters, computation, or output plots and tables. In particular, the *Parameters and Analysis* tab contains all of the cost, utility, death rate and other input point values and distribution paramaters. The *Markov model* tab calculates for both treatment scenarios all of the state population counts, summary statistics, and total costs and QALYs in adjacent columns. Rows correspond to a single cycle and each simulation total is calculated at the bottom of the sheet. The *Scenario Summary* and *Cohort* tabs give the output tables and plots, respectively.

We can make some direct comparisons between MS Excel and R. In MS Excel values are entered in a cell which can be named so that it can be referred to elsewhere in the workbook by its name as well as it cell location. In R the equivalent is to define a variable and assign it a value. The name of the variable can subsequently be used in the code in place of explicitly embedding the value directly (called "hard-coding"). Similarly, groups of values can be referred to by name in MS Excel by selecting a range of cells and naming these. These can then be handled as a single object e.g. as an input to a function. In R there are several ways of defining a group of values like this but for the purposes of this tutorial we shall use the matrix and array data structures which are two-dimensional and n-dimensional rectangular objects, respectively. See [26] for more details.

2.3 Translating an MS Excel model using loops

In this section we provide a demonstration on how to translate a MS Excel Markov model to an equivalent R model by defining the use of *for* loops which will be implemented in this tutorial. A *for* loop is a type of control flow statement which repeatedly executes a section of code for a defined number of iterations. The focus will be on the calculation of the state populations over time. Part of the MS Excel Markov model (Markov Model tab) is shown in **Fig. 2a**. Each row represents a cycle (cycle length is one year) so that discrete time steps go from top to bottom and the columns contain the state populations, time-dependent probabilities and (cumulative) functions of these, such as total quality-adjusted life years (QALYs) and total costs. The counts are shown to one decimal place to emphasize that fractions of the starting state population are simulated. In **Fig. 2a** there are 1000 patients in the asymptomatic health state (*Asymp)* and 0 in both the progressive health state (*Prog)* and *Dead* states. Probabilities, defined within the "Parameters and Analysis" sheet, are used to calculate the movements between health states. The all-cause mortality was dependent on age and we assumed that all the individuals in the cohort had the same age at the start of the simulation. Values were taken from standard life-tables and aggregated in to 10 year intervals. For example, by cycle 15, in row 20, the 1000 patients are distributed differently - 195 patients in the *Asymp* health state, 233 in the *Prog* health state and 572 who are *Dead*.

Fig. 2b outlines the corresponding formulae behind the same model. The *Asymp* and *Prog* health state formulas are shown for simplicity, but the same pattern applies to the rest of the sheet. Recall that the @ notation is used in this model which substitutes the value on the same line from the named column.

State populations
Outcome statistics
Time

Cycle	rDeath	Asymp	Prog	Death	LE	QALE	LYs	QALYs	StateCost	TransCost	TotalCost
0	0	1000.0	0.0	0.0							
1	0.0138	976.2	10.0	13.8	986.2	934.9	930.4	882.0	488774	0	488774
2	0.0138	943.2	27.9	28.9	971.1	917.0	864.3	816.1	494180	1415	495595
3	0.0138	901.9	51.6	46.5	953.5	895.5	800.6	751.9	508632	3723	512355
4	0.0138	853.4	79.2	67.4	932.6	870.1	738.7	689.2	526261	6500	532761
5	0.0138	798.9	108.9	92.1	907.9	840.7	678.4	628.2	542688	9414	552102
6	0.0138	740.0	139.0	121.0	879.0	807.2	619.6	569.1	554831	12209	567041
7	0.0138	678.0	168.0	154.0	846.0	770.1	562.6	512.2	560718	14700	575419
8	0.0138	614.4	194.8	190.9	809.1	729.7	507.7	457.8	559306	16764	576070
9	0.0138	550.6	218.1	231.3	768.7	686.7	455.0	406.4	550311	18329	568640
10	0.0138	487.9	237.5	274.6	725.4	641.6	405.1	358.3	534043	19368	553411
11	0.0379	415.8	246.5	337.7	662.3	579.9	348.9	305.5	499111	19891	519002
12	0.0379	350.1	250.1	399.8	600.2	520.2	298.3	258.5	459869	19480	479349
13	0.0379	291.3	248.6	460.0	540.0	463.2	253.2	217.2	417980	18643	436624
14	0.0379	239.5	242.7	517.8	482.2	409.5	213.3	181.1	374993	17484	392477
15	0.0379	194.5	233.0	572.5	427.5	359.5	178.4	150.0	332266	16101	348367
16	0.0379	156.0	220.4	623.6	376.4	313.5	148.2	123.4	290928	14584	305512

Fig. 2a MS Excel model cycles and population flow, costs and QALYs for cohort withoutdrug treatment, showing values. *LE* life expectancy, *QALE* quality-adjusted life expectancy, *LY* life-year *QALY* quality-adjusted life-year, *rDeath* probability of death each year, *StateCost* total state occupancy cost each year, *TransCost* total state transition cost each year, TotalCost sum of *StateCost* and *TransCost*.

Cycle	rDeath	Asymp	Asymp (Cycle-1)	Prog	Prog (Cycle-1)
0	0	1000		0	
1	0.0138	=n_cohort-@Prog-@Death	=D5	=@Prog_lag*(1-tpDcm-@rDeath)+@Asymp_lag*tpProg*@ncycle	=F5
2	0.0138	=n_cohort-@Prog-@Death	=D6	=@Prog_lag*(1-tpDcm-@rDeath)+@Asymp_lag*tpProg*@ncycle	=F6
3	0.0138	=n_cohort-@Prog-@Death	=D7	=@Prog_lag*(1-tpDcm-@rDeath)+@Asymp_lag*tpProg*@ncycle	=F7
4	0.0138	=n_cohort-@Prog-@Death	=D8	=@Prog_lag*(1-tpDcm-@rDeath)+@Asymp_lag*tpProg*@ncycle	=F8
5	0.0138	=n_cohort-@Prog-@Death	=D9	=@Prog_lag*(1-tpDcm-@rDeath)+@Asymp_lag*tpProg*@ncycle	=F9
6	0.0138	=n_cohort-@Prog-@Death	=D10	=@Prog_lag*(1-tpDcm-@rDeath)+@Asymp_lag*tpProg*@ncycle	=F10
7	0.0138	=n_cohort-@Prog-@Death	=D11	=@Prog_lag*(1-tpDcm-@rDeath)+@Asymp_lag*tpProg*@ncycle	=F11

Fig. 2b MS Excel model cycles and population flow, showing formulae. *rDeath* probability of death each year, *tpProg* probability of disease progression, *tpDcm* excess probability of death,

Notice that besides the initial row, the formulae in the *Asymp* and *Prog* columns are the same for all rows. When this pattern occurs in code it can be simplified and written as a loop. **Fig. 3** is an example of some pseudo-code (non-R code) which takes the MS Excel formulae and uses it in a *for* loop.

Define constants:

```
n_cohort = 1000
n_cycle = 45
tpProg = 0.01
tpDcm = 0.15
```

Define starting populations at *t*=0 i.e. first row:

```
@Asymp = n_cohort
@Prog = 0
@Death = 0
```

Iterate over cycles i.e. row by row:

```
loop from t = 1 to n_cycle
  @Prog = @Prog_lag*(1 - tpDcm - @rDeath) + @Asymp_lag*tpProg*n_cycle
  @Death = @Death_lag + @Prog_lag*tpDcm + (@Asymp_lag + @Prog_lag)*@rDeath
  @Asymp = n_cohort - @Prog - @Death
End
```

Fig. 3 Pseudo-code showing how the MS Excel formulae can be represented programmatically as a loop. *rDeath* probability of death each year, *tpProg* probability of disease progression, *tpDcm* excess probability of death.

The pseudo-code in **Fig. 3** outlines how the cohort begins with 1000 patients (*n_cohort)*, for 45 cycles (*n_cycle)*, and two probabilities (*tpProg* and *tpDcm*) influencing the movement between health states. Once the starting populations are defined (1000 patients in the *Asymp* health state and 0 in both the *Prog* and *Dead* states), it is then possible to loop over each cycle using the updated information from the cycle before. To loop from cycle 1 to cycle 45, the formulae for each health state is used.

Further, it is possible to rewrite the pseudo-code of the loop using a transition matrix and using matrix multiplication (**Fig. 4**) equivalently and succinctly. This mathematical formulation is common in Markov models.

```
loop from t = 1 to n_cycle
     (@Asymp, @Prog, @Death) = (@Asymp, @Prog, @Death) %*%
                               [ C   tpProg*n_cycle   @rDeath         ]
                               [ 0   C                tpDcm + @rDeath ]
                               [ 0   0                1               ]
End
```

Fig. 4 Pseudo-code showing an approach with matrix multiplication within a loop for the MS Excel population movement. *rDeath* probability of death each year, *tpProg* probability of disease progression, *tpDcm* excess probability of death, *C* one minus the sum of all other probabilities from a state.

The command %*% is called an *infix* operator and is used for matrix multiplication in R. The 1 by 3 matrix (@Asymp, @Prog, @Death) is the number of patients in the three health states. The 3 by 3 matrix represents the possible transitions between health states. For example, the top right entry is the value tpDn which describes the movement between *Asymp* to *Death*. The values C in the matrix represents 1 minus the probabilities in the remainder of the corresponding row. This idea of using loops is the basis for all the following R code throughout the rest of this tutorial.

2.4 Setting up the R model

The MS Excel model models a cohort of 1000 patients that receives a drug and compares to a cohort of 1000 patients that does not receive a drug to assess how patients move throughout the health states over time with a chronic disease. To set up the MS Excel model in R some definitions are needed first. **Fig. 5** outlines the R code used to define the number of treatments (prefixed with t_) and their names (prefixed with n_), the number of states (prefixed with s_) and their names. The number of cycles in the R model starts at 1 (not 0 as in spreadsheet models) and so the number of cycles is 46 rather than 45 as the cycle length is 1 year. The initial age of patients in the cohort begins at 55. The same variable names have been used as the MS Excel model where feasible to help with comparisons. The probabilities of death each year by age taken from standard life-tables where 0.0138 for 55-65, 0.0379 for 66-75, 0.0912 for 76-85 and 0.1958 for >85 years old.

```
t_names ← c("without_drug", "with_drug")
n_treatments ← length(t_names)

s_names  ← c("Asymptomatic_disease", "Progressive_disease", "Dead")
n_states ← length(s_names)

n_cohort ← 1000

n_cycles ← 46
Initial_age ← 55
```

Fig. 5 Defining the treatments, names, states, cycles and starting age in R

The unit costs and unit utilities associated with each of the health states as well as the cost of the drug need to be defined at the start and shown in **Fig. 6**. The utility of being in the dead health state is 0 so does not need to be defined here. Costs begin with a 'c' and utilities with a 'u'. The discount rate for costs and outcomes are also included at a rate of 6%. Other discounts rates may be preferred in other analyses but we remain consistent with Briggs (1998). The excess probability of dying from the disease in a single cycle (*tpDcm*) as above in Section 2.2 is now defined. The all-cause mortality *tpDm* is not defined here since it is a time-dependent variable and will be defined later.

```
cAsymp ← 500
cDeath ← 1000
cDrug ← 1000
cProg ← 3000
uAsymp ← 0.95
uProg ← 0.75
oDr ← 0.06
cDr ← 0.06
tpDcm ← 0.15
```

Fig. 6 Defining the costs, utilities, discount rates and probability of dying in R

This process of defining treatment names, states and cycles is similar to that in MS Excel. An additional step required in R is to create empty objects into which the outcomes of calculations will be entered. Using R this way has more flexibility for the structure of the objects we use for the computation and for saving the results, and not just a flat, 2D matrix. We shall define the same structure for the matrices for the cost of transition to a health state, and cost and QALYs accrued being in health state. **Fig. 7a** shows the matrix for the cost of transitioning to a state, trans_c_matrix. All costs are zero except for the cost of £1000 for transitioning to the *Dead* state from the *Progressive disease* state. The costs are the same for both cohort.

The first argument defines a vector of values for each health state similar to a column of values in a parameters sheet in MS Excel. The argument byrow = TRUE makes sure that the costs from the vector fill up the rows of the matrix from left to right before moving to the next row. Therefore, the resulting matrix will look like how we have arranged the vector, shown in

(**Fig. 7b**).

```
trans_c_matrix ←
  matrix(c(0, 0, 0,
           0, 0, cDeath,
           0, 0, 0),
         byrow = TRUE,
         nrow = n_states,
         dimnames = list(from = s_names,
                         to = s_names))
```

Fig. 7a Creating empty matrix space in R for the transition cost matrix

```
> trans_c_matrix
                      to
from                    Asymptomatic_disease Progressive_disease Dead
  Asymptomatic_disease                    0                   0    0
  Progressive_disease                     0                   0 1000
  Dead                                    0                   0    0
```

Fig. 7b The trans_c_matrix created in R

In a similar way, **Fig. 8a** describes the cost of being in a health state. The structure of state_c_matrix is treatment by state and is assigned non-zero costs to being in the *Asymp* and *Prog* health states. Again, the argument byrow = TRUE ensures that the without drug cohort costs are along the first row and the with drug cost along the second row(**Fig. 8b**).

```
state_c_matrix <-
  matrix(c(cAsymp, cProg, 0,
           cAsymp + cDrug, cProg, 0),
         byrow = TRUE,
         nrow = n_treatments,
         dimnames = list(t_names,
                         s_names))
```

Fig. 8a Creating a matrix for the costs of being in each health state in R

```
> state_c_matrix
             Asymptomatic_disease Progressive_disease Dead
without_drug                  500                3000    0
with_drug                    1500                3000    0
```

Fig. 8b The costs of being in each health state for each treatment

An input matrix for the QALYs accrued being in health state are created the same way (**Online Resource 2**).

The probability of dying from the disease in a single cycle (tpDcm) was defined above. Space is also needed to define the transition probabilities between health states. A matrix is created as above but with another dimension for the movements between health states (**Fig. 9a**). The resulting matrix is shown in **Fig. 9b** – it shows an empty space for the transitions between each health state dependent on the cohort being with or without a drug. **Online Resource 2** outlines how we can assign a cost to entering the health state as well as occupying it.

```
p_matrix <- array(data = 0,
                  dim = c(n_states, n_states, n_treatments),
                  dimnames = list(from = s_names,
                                  to = s_names,
                                  t_names))
```

Fig. 9a Creating a 3-dimensional matrix for probabilities in R

```
> p_matrix
, ,  = without_drug

                     to
from                  Asymptomatic_disease Progressive_disease Dead
  Asymptomatic_disease                    0                   0    0
  Progressive_disease                     0                   0    0
  Dead                                    0                   0    0

, ,  = with_drug

                     to
from                  Asymptomatic_disease Progressive_disease Dead
  Asymptomatic_disease                    0                   0    0
  Progressive_disease                     0                   0    0
  Dead                                    0                   0    0
```

Fig. 9b The probability transition matrix created in R The output shows two "slices" through the array, one without drug treatment and one with drug treatment.

2.5 Time-dependent components

One addition in R relative to MS Excel is that we should prepare for an analysis by creating objects with empty space for when calculations are performed. It is like creating the cells in MS Excel. Of course, in MS Excel this step is unnecessary since the spreadsheet cells are in some sense already defined and available to assign values to. By creating matrices and arrays in R, empty space for population counts, costs and utilities in each health state for patients with or without the drug is specified. Appending values to other objects at run time in R is possible but ill advised because it is slower and more error prone. The labelling of cells and groups of cells in MS Excel could be thought of as a similar process as creating a matrix in R.. A population matrix (pop) and transition matrix (trans) are created in the same way so there is blank space for each health state, with or without a drug, for each of the 46 cycles, known as an "array". **Fig. 10a** and **Fig. 10b** below show this R code for a generic array (cycle_empty_array), see **Online Resource 3** for how this applied to transition arrays.

```
cycle_empty_array <-
  array(NA,
        dim = c(n_treatments, n_cycles),
        dimnames = list(treatment = t_names,
                        cycle = NULL))
```

Fig. 10a Creating empty space for each treatment and cycle in R

```
> cycle_empty_array
              cycle
treatment      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17]
  without_drug   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA    NA    NA    NA    NA
  with_drug      NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA    NA    NA    NA    NA
              cycle
treatment      [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33]
  without_drug    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
  with_drug       NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
              cycle
treatment      [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46]
  without_drug    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
  with_drug       NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
```

Fig. 10b The resulting empty space for 46 cycles and both treatments

Transition probabilities in a Markov model can either be time-independent, such as tpDcm, or time-dependent (like @rDeath in the MS Excel model). In this model, the transition probabilities from *Asymp* to *Dead* are time dependent, as they depend on the age of the cohort.Therefore the transition probabilities depend on the cyclewhich is why we need to have an extra cycle dimension in the empty space for probabilities, costs and QALYs.

To account for the time dependency, the representation by a hard-coded array in MS Excel was replaced with a function in R named p_matrix_cycle which is called at each cycle iteration, similar to the logic in Section 2.2. By simply replacing a fixed valued array with a function, this decouples the calculation of the transition matrix and the higher-level cost-effectiveness calculations. This makes changes and testing to either part easier and more reliable. This function is defined and broken into three components. **Fig. 11a** outlines the first part of the function which defined the fixed transition probabilities tpProg and tpDcm as input arguments to the function with default values as well as the proportion reduction of the transition probability from asymptomatic to progressive disease due to the drug (effect = 0.5). Note that, strictly speaking, in R this is an array data structure but we have named the data structure p_matrix to emphasise that it provides what is known in Markov modelling as the transition probability matrix. The time dependent probabilities are those that depend on the age of the cohort. A lookup function is used to describe the transition probability from *Asymp* to *Dead* using 6 age group categories as in the MS Excel model.

```
p_matrix_cycle <- function(p_matrix, age, cycle,
                           tpProg = 0.01,
                           tpDcm = 0.15,
                           effect = 0.5) {
  tpDn_lookup <-
    c("(34,44]" = 0.0017,
      "(44,54]" = 0.0044,
      "(54,64]" = 0.0138,
      "(64,74]" = 0.0379,
      "(74,84]" = 0.0912,
      "(84,100]" = 0.1958)

  age_grp <- cut(age, breaks = c(34,44,54,64,74,84,100))
  tpDn <- tpDn_lookup[age_grp]
```

Fig. 11a The first part of the probability transition matrix function p_matrix_cycle which assigns age-dependent probabilities in R

```
p_matrix["Asymptomatic_disease", "Progressive_disease", "without_drug"] <- tpProg*cycle
p_matrix["Asymptomatic_disease", "Dead", "without_drug"] <- tpDn
p_matrix["Asymptomatic_disease", "Asymptomatic_disease", "without_drug"] <- 1 - tpProg*cycle - tpDn
p_matrix["Progressive_disease", "Dead", "without_drug"] <- tpDcm + tpDn
p_matrix["Progressive_disease", "Progressive_disease", "without_drug"] <- 1 - tpDcm - tpDn
p_matrix["Dead", "Dead", "without_drug"] <- 1
```

Fig. 11b The second part of the function p_matrix_cycle calculating transition probabilities for the cohort without a drug, where the array dimensions are called in R by array[row, columns, dimension]

```
  p_matrix["Asymptomatic_disease", "Progressive_disease", "with_drug"] <- tpProg*(1 - effect)*cycle
  p_matrix["Asymptomatic_disease", "Dead", "with_drug"] <- tpDn
  p_matrix["Asymptomatic_disease", "Asymptomatic_disease", "with_drug"] <-
    1 - tpProg*(1 - effect)*cycle - tpDn
  p_matrix["Progressive_disease", "Dead", "with_drug"] <- tpDcm + tpDn
  p_matrix["Progressive_disease", "Progressive_disease", "with_drug"] <- 1 - tpDcm - tpDn
  p_matrix["Dead", "Dead", "with_drug"] <- 1

  return(p_matrix)
}
```

Fig. 11c: The third part of the function p_matrix_cycle calculating transition probabilities for the cohort with a drug

Fig. 11b and **Fig. 11c** outline the specific calculations for the transition probabilities including those dependent on age linked to the age specific values, as well as the fixed values. This p_matrix_cycle function can now be used when calculation the costs and QALYs over time as it will calculate the corresponding p_matrix each cycle.

So the filled array of transition probabilities (p_matrix), matrices representing state costs and rewards of the different health states (state_c_matrix, state_q_matrix) and transition costs and rewards (trans_c_matrix, trans_q_matrix) have been constructed based on the parameters defined in the beginning of the model code. Now, we run the model by combining these objects, much like the process of multiplication performed across multiple MS Excel sheets.

2.6 Running the model

To run the model and combine all the information and code from the previous sections, an algorithm will be created. The steps in an algorithm must be 'flattened-out' in MS Excel. That is, one must copy-and-paste the same formula across several cells in order to repeat a calculation for different input values. Ubiquitous in programming languages, the *for* loop means that we do not have to do this.

A loop is first created using the p_matrix function. **Fig. 12** gives the R code used with the starting age of the cohort, and subsequently updating the age and the corresponding pop and trans matrices. This loop repeats from cycle number 2 to cycle 46 (n_cycles), as cycle 1 was already defined above, equivalent to cycle 0 rows in MS Excel models. There is also an element i in this loop, which we shall see is from another encapsulating loop for the two treatments which incorporates this loop. Recall the matrix multiplication operator %*%, used here to calculate the state population at the next time step from the current time population (pop) and the number of individuals who transition between states (trans).

Notice the connection with the discussion in Section 2.2. This is the equivalent R code to the MS Excel pseudocode of **Fig. 4**.

```
for (j in 2:n_cycles) {

  p_matrix <- p_matrix_cycle(p_matrix, age, j - 1)

  pop[, cycle = j, treatment = i] <-
    pop[, cycle = j - 1, treatment = i] %*% p_matrix[, , treatment = i]

  trans[, cycle = j, treatment = i] <-
    pop[, cycle = j - 1, treatment = i] %*% (trans_c_matrix * p_matrix[, , treatment = i])

  age <- age + 1
}
```

Fig. 12 The first, inner loop in running the model calculating population counts at each cycle in R

If we execute this code snippet for a single treatment of without drug (i=1) and the initial age fixed at age=50, then the first 10 time steps of the pop array are shown in **Fig. 13**.

```
> pop
, , treatment = without_drug

                       cycle
state                   [,1]  [,2]      [,3]      [,4]      [,5]      [,6]     [,7]     [,8]     [,9]    [,10]
  Asymptomatic_disease 1000 985.6 961.55136 928.47399 887.24975 838.98336 777.0664 711.9482 645.1675 578.1991
  Progressive_disease     0  10.0  28.16800  52.66540  81.67282 113.42503 145.1850 175.7984 203.9584 228.6151
  Dead                    0   4.4  10.28064  18.86061  31.07743  47.59161  77.7486 112.2534 150.8741 193.1858
```

Fig. 13 The first 10 time steps of the pop array in R

To complete the simulation code we now also present the outer loop which uses the state population counts to calculate the cost-effectiveness analysis statistics in **Fig. 14**. To simplify this code we substitute where the code snippet in Fig. 12 would be with a function call to sim_pop() which contains equivalent code. (full code for running the model is given in the Supplementary Material). The cost, QALY, LE, LY and QALE at each cycle are calculated as the p_matrix is updated at each iteration of the loop. The discount rate is also incorporated into the model here easily as each cycle's costs and QALYs will depend on the cycle number. These repeated steps are performed for each of the two treatments (1:n treatments) and the total costs and QALYs over the lifetime of the model can then be calculated for each treatment.

```
for (i in 1:n_treatments) {

  # simulate state populations
  sim_res <-
    sim_pop(n_cycles, Initial_age,
            trans_c_matrix,
            p_matrix, pop, trans, i)

  trans[, , i] <- sim_res$trans
  pop[, , i] <- sim_res$pop

  cycle_state_costs[i, ] <-
    (state_c_matrix[treatment = i, ] %*% pop[, , treatment = i]) * 1/(1 + cDr)^(1:n_cycles - 1)

  # discounting at _previous_ cycle
  cycle_trans_costs[i, ] <-
    (c(1,1,1) %*% trans[, , treatment = i]) * 1/(1 + cDr)^(1:n_cycles - 2)

  cycle_costs[i, ] <- cycle_state_costs[i, ] + cycle_trans_costs[i, ]

  # life expectancy
  LE[i, ] <- c(1,1,0) %*% pop[, , treatment = i]

  # life-years
  LYs[i, ] <- LE[i, ] * 1/(1 + oDr)^(1:n_cycles - 1)

  # quality-adjusted life expectancy
  cycle_QALE[i, ] <-
    state_q_matrix[treatment = i, ] %*% pop[, , treatment = i]

  # quality-adjusted life-years
  cycle_QALYs[i, ] <- cycle_QALE[i, ] * 1/(1 + oDr)^(1:n_cycles - 1)

  total_costs[i] <- sum(cycle_costs[treatment = i, -1])
  total_QALYs[i] <- sum(cycle_QALYs[treatment = i, -1])
}
```

Fig. 14 The simulation code including the second, outer loop of the main model which calculates cost-effectiveness analysis summary statistics in R

2.7 Results

Displaying the results after running the model is very easy in R. As the incremental cost-effectiveness ratio (ICER) requires the incremental costs and incremental QALYs, **Fig. 15** outlines the two lines of R code needed to subtract the total costs of the *without_drug* cohort from the *with_drug* cohort. These results will therefore assume that the strategy where the cohort are without a drug is the standard of care or base case analysis. Swapping with and without the drug will change this around. It is important to note that the calculation of the ICER is relevant here as there are two strategies being compared. If there are more than two strategies, more than one ICER may be required to be calculated. In the case of more than two strategies and one is dominated, an ICER would not be relevant for that strategy compared to the others.

```
c_incr <- total_costs["with_drug"] - total_costs["without_drug"]
q_incr <- total_QALYs["with_drug"] - total_QALYs["without_drug"]

ICER <- c_incr/q_incr
```

Fig. 15 The R code to calculate the ICER. *ICER* incremental cost-effectiveness ratio

The code to plot the results on a cost-effectiveness plane is given in **Fig. 16**. The diagonal line indicates the willingness to pay threshold of £20,000 per QALY and the ICER is indicated by the black point. We can see that the drug treatment is cost-effective against the no drug option. The x-axis and y-axis limit are set with xlim and ylim arguments. These can be omitted to allow R to calculate the axes limits is desired. Note that the plot in **Fig. 17** is a graph created using base R. There are many packages that can display more sophisticated plots, e.g with a range of colours and other details, that are not covered in this tutorial, including BCEA [10].

```
wtp <- 20000
plot(x = q_incr/n_cohort, y = c_incr/n_cohort,
     xlim = c(0, 2),
     ylim = c(0, 15e3),
     pch = 16, cex = 1.5,
     xlab = "QALY difference",
     ylab = paste0("Cost difference (", enc2utf8("\u00A3"), ")"),
     frame.plot = FALSE)
abline(a = 0, b = wtp) # willingness-to-pay threshold
```

Fig. 16 R code to plot the cost-effectiveness plane

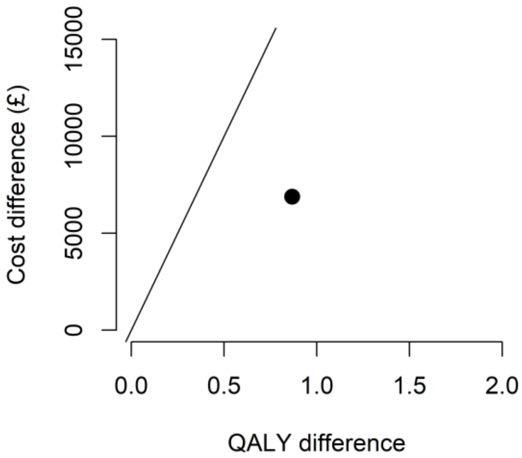

Fig. 17 Results of the analysis per person on a cost-effectiveness plane comparing drug treatment against no drug cohorts, with an ICER of £7931/QALY represented by the black piont. The willingness to pay threshold for £20,000 is represented by the diagonal line. *ICER* incremental cost-effectiveness ratio, *QALY* quality-adjusted life-years

3 Probabilistic Sensitivity Analysis (PSA)

The formulation in the previous section can be extended to include uncertainty about one or more of the parameters. Briggs (2000) [18] describe this for the current model by repeating the analytical solution of the model employing different values for the underlying parameters sampled from specified ranges and distributions. The MS Excel implementation of the example in [18] is also provided in the GitHub repository for this tutorial paper. Sensitivity analyses can be performed one at a time (one-way) or for multiple parameters simultaneously (multi-way). This section presents a multi-way probabilistic sensitivity analysis.

In MS Excel random numbers can be generated most simply for a uniform distribution using the RAND() function. Samples from some other distribution can be made using an inverse cumulative density transformation e.g. for the gamma distribution with GAMMA.INV(p,a,b). However, a full PSA is arguably not practical without resorting to additional Visual Basic for Applications (VBA) coding due to the number of simulations run. We recommend that all calculations should remain in the spreadsheet and accessible to the user without the need to understand or interpret VBA coding, which is only there to facilitate the repeated application of the results. At the point where the user needs to write substantial code inside of MS Excel then there is an especially strong argument to transfer to a software that is more specifically designed for this purpose, such as R.

Performing a PSA analysis can be done by inputting random draws from the unit costs and QALY distributions as inputs to the existing model function. In R, there are numerous ways of implementing a PSA. Following from the R code presented in the previous sections, we can wrap this model code in a function, e.g. called ce_markov(), which we can then repeatedly call with different parameter values. To this we will need to pass the starting conditions: population (start_pop), age (init_age) and number of cycles (n_cycle) (in our case, if not defined then age and number of cycles are assigned default values). We will also need the probability transition matrix (p_matrix), state cost and QALY matrices (state_c_matrix, state_q_matrix, trans_c_matrix).

In extension to the first analysis, the unit values have distributions rather than point values. To sample from a base R distribution the function name syntax is a short form version of the distribution name preceded by an 'r' (for *r*andom or *r*ealisation). For example, to sample a single value from a standard normal distribution then call rnorm(1). All of the random numbers could be sampled before running the model which would allow us to save them to use again and improve run time because this would only be performed once outside of the main loop. Alternatively, we can sample the random variables at runtime, within the Markov model function. This is arguably neater and if we wish to replicate a particular run, in the same way as prerunning the sampling, then the random seed can be set beforehand with set.seed(). We will demonstrate how to implement a simple version when sampling at runtime. We will not assume any correlation between parameters but if multivariate sampling is required then this is another reason to use R. Because we will want to sample more than once, rather than just once at the start of the simulation, we can wrap the random sampling statements in a function so that they are called newly every time the Markov model is run. So, using the same names as we used for the point values in the previous analysis.

```
cAsymp <- function() rnorm(1, 500, 127.55)
cDeath <- function() rnorm(1, 1000, 255.11)
cDrug  <- function() rnorm(1, 1000, 102.04)
cProg  <- function() rnorm(1, 3000, 510.21)
effect <- function() rnorm(1, 0.5, 0.051)
tpDcm  <- function() rbeta(1, 29, 167)
tpProg <- function() rbeta(1, 15, 1506)
uAsymp <- function() rbeta(1, 69, 4)
uProg  <- function() rbeta(1, 24, 8)
```

Fig. 18 R code for random sampling input values in probability sensitivity analysis (PSA)

Similarly, rather than using fixed state_c_matrix, trans_c_matrix and state_q_matrix, if we define these as functions, we can sample newly their component values each time they are called. In practice, the code looks the same as previously but now the unit values are function calls so are followed by open and closed brackets. The example for `state_c_matrix` is shown in **Fig.19** and the full code is given in Supplementary Material.

```
state_c_matrix <- function() {
  matrix(c(cAsymp(), cProg(), 0,             # without drug
           cAsymp() + cDrug(), cProg(), 0), # with drug
           byrow = TRUE,
           nrow = n_treatments,
           dimnames = list(t_names,
                           s_names))
}
```

Fig. 19 R code for state cost matrix in probability sensitivity analysis (PSA)

To finally obtain the PSA output, loop over ce_markov() remembering to record the cost and QALYs outputs each time.

```
n_trials ← 500

costs ← matrix(NA, nrow = n_trials, ncol = n_treatments,
                dimnames = list(NULL, t_names))
qalys ← matrix(NA, nrow = n_trials, ncol = n_treatments,
                dimnames = list(NULL, t_names))

for (i in 1:n_trials) {
  ce_res ← ce_markov(start_pop = c(n_cohort, 0, 0),
                     p_matrix,
                     state_c_matrix(),
                     trans_c_matrix(),
                     state_q_matrix())

  costs[i, ] ← ce_res$total_costs
  qalys[i, ] ← ce_res$total_QALYs
}
```

Fig. 20 R code for full analysis model in probability sensitivity analysis (PSA)

The output of this gives the cost-effectiveness plane in **Fig. 21**. Code to produce this plot is given in the Supplementary Material.

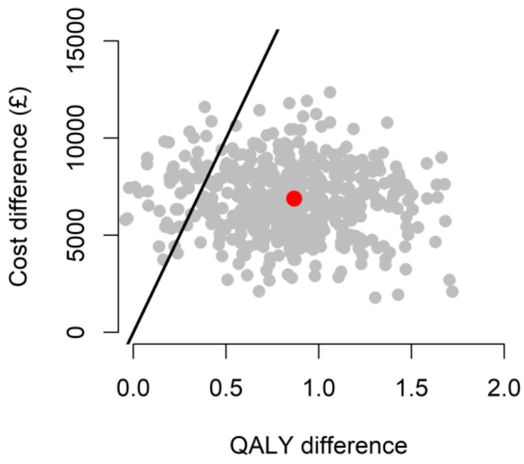

Fig. 21 PSA cost-effectiveness plane per person comparing drug treatment against no drug cohorts. The diagonal line indicates the willingness to pay threshold of £20,000 per QALY. The red point represents the ICER. *PSA* probability sensitivity analysis, *ICER* incremental cost-effectiveness ratio, *QALY* quality-adjusted life-years

4 Discussion

This tutorial paper has detailed a step-by-step tutorial to implement a Markov model in R for MS Excel users. This was based on a widely used original MS Excel model so that users can move more easily from using one software to the other by following the steps. Both the R code and the MS Excel workbook are freely available online at https://github.com/Excel-R-tutorials/Markov_Intro.

The gap between MS Excel and R can be bridged by not "throwing the baby out with the bathwater" when difficult modelling approaches and assumptions arise for beginner users of R. Through practical experience and discussions, the authors have learned that there are several drawbacks to creating HEE models in MS Excel, and to promote the use of R these need to be addressed. Many individuals are keen to learn new tools if they feel it is worthwhile to commit the extra time and effort and talking their "Excel language" is an important part of this. It is also important to note that performing a probabilistic sensitivity analysis in MS Excel requires the use of VBA code, and so these general programming skills will be highly beneficial when moving to R.

Several packages have been written specifically for HEE modelling in R such *heemod* [15], *hesim* [24] and *BCEA* [10]. The increasing numbers of available HEE R packages can be considered in light of an 'open source modelling' movement in the health economics community (see the R for Health Technology Assessment group (https://r-hta.org/) and the International Society for Pharmacoeconomics and Outcomes Research (ISPOR) Open Source Models Special Interest Group (https://www.ispor.org/member-groups/special-interest-groups/open-source-models). These packages make it easier and more robust to implement and analyse a wide variety of models in R. In many practical cases, these packages may be a preferred solution to coding up a HEE model from scratch. They are open source, meaning that the code can be inspected, modified, and enhanced by anyone. They are also used by hundreds or thousands of people so errors can be identified and corrected that might be missed in someone's de novo code. However, we wish to highlight that a rationale for using standard MS Excel for modelling, in contrast to add-ins with additional functionality (such as @Risk) is to help the modeller to understand and question the assumptions and principles of constructing a model rather than simply how to operate a piece of software. We propose that the same principle applies to this tutorial. Further, building models in third party software will never be as transparent or bespoke as a model written by the user in the original language.

Special Interest Group previous papers have highlighted the increased speed and flexibility in using R over other software for HEE [11-13]. Additionally, utilising R allows for the fairly easy conversion of R code into more user-friendly models through RShiny apps [3,8]. The intention of this tutorial paper was not to show the "best" way of creating cost-effectiveness models in R, but the code has been deliberately simplified so that the barrier to entry for people new to R is lowered. Using explicit loops rather than using the mapping functions (e.g. the base R apply family of functions) show an easier and more explicit flow of the program and reasoning. In R there are often numerous ways of solving the same problem, depending on personal preference and experience (see base R vs *tidyverse* discussions [23]). This could include use of a completely different programming paradigm, such as object oriented programming [25]. We created code that is intuitive and simple and not necessarily the shortest, fastest or most elegant. Additionally, for improved reproducability in R, analyses can be documented with literate programming notebooks such as R Markdown [26] or Quarto [27]. These interleave the code and accompanying text to provide details about procedures and data so the same analysis could be repeatedThrough open-source, documented, clean code we hope to promote the increased transparency and re-use of future health economic models.

Once users that are new to R have grasped the basic implementation of HEEs in R, the next stage is to move on to other R tutorials [12-13] and develop more general R skills [25,28]. This can allow integration of statistical analyses and economic analyses in the same software. In addition, further HEE analyses such as Value of Information which is difficult to perform in MS Excel can be performed relatively easily in R to provide a more robust HEE in addition to all of the other benefits of performing a HEE in R.

5. Abbreviations

HEE; health economic evaluation
ICER; incremental cost-effectiveness ratio
IDE; Integrated Development Environment
LE; life expectancy
LY; life-year
MM; multistate model
MS; Microsoft
PSA; probabilistic sensitivity analysis
QALE; Quality-adjusted life expectancy
QALY; Quality-adjusted life-year
VBA; Visual Basic for Applications

6. Statements and Declarations

1. Competing Interests

The authors declare they have no conflicts of interest.

2. Authorship statement

N Green: Conceptualization, writing original draft, validation, writing review and editing**. F Lamrock:** Conceptualization, writing original draft, validation, writing review and editing. **N Naylor:** validation, writing review and editing. **J Williams:** validation, writing review and editing. **A Briggs:** Resources, writing review and editing.

3. Data Availability

All data used for this tutorial paper are available at https://github.com/Excel-R-tutorials/Markov-model-introduction

4. Funding

Financial support for this study was provided by a Medical Research Council Centre pump-priming award. The funding agreement ensured the authors' independence in designing the study, interpreting the data, writing, and publishing the report. The views expressed are those of the author(s) and are not necessarily those of author-affiliated institutions, including the National Institute for Health Research, the UK Health Security Agency or the Department of Health and Social Care.

5. Ethics

Not applicable.

5 7. References

1. Drummond MF, O'Brien B, Stoddart GL, et al. Methods for the economic evaluation of health care programmes. Oxford University Press; 1997; 978-0198529453

2. Weinstein MC, Stason WB. Foundations of cost-effectiveness analysis for health and medical practices. New England journal of medicine. 1977;296(13):716-721.

3. Hart R, Burns D, Ramaekers B, et al. R and Shiny for Cost-Effectiveness Analyses: Why and When? A Hypothetical Case Study. PharmacoEconomics. 2020; 38:765–776.

4. Edlin R, McCabe C, Hulme C, et al. Cost Effectiveness Modelling for Health Technology Assessment, A Practical Course. Springer; 2015; https://doi.org/10.1007/978-3-319-15744-3

5. Lunn DJ, Thomas A, Best N et al. WinBUGS - A Bayesian modelling framework: Concepts, structure, and extensibility. Statistics and Computing. 2000;10:325–337. https://doi.org/10.1023/A:1008929526011

6. R Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. 2020. Available from https://www.R-project.org/.

7. Baio G, Heath A. When simple becomes complicated: why Excel should lose its place at the top table. Glob Reg Health Technol Assess. 2017;4(1):e3-e6.

8 Xin Y, Gray E, Robles-Zurita JA, Haghpanahan H, Heggie R, Kohli-Lynch C, Briggs A, McAllister DA, Lawson KD, Lewsey J. From Spreadsheets to Script: Experiences From Converting a Scottish Cardiovascular Disease Policy Model into R. Applied Health Economics and Health Policy. 2021;0123456789. https://doi.org/10.1007/s40258-021-00684-y

9. RStudio Team. RStudio: Integrated Development for R. RStudio, PBC, Boston, MA URL. 2020. Available from http://www.rstudio.com/.

10. Baio G, Berardi A, Heath A. Bayesian Cost-Effectiveness Analysis with the R package BCEA. Springer; 2017; https://doi.org/10.1007/978-3-319-55718-2

11. Jalal H, Pechlivanoglou P, Krijkamp E, et al. An Overview of R in Health Decision Sciences. Med Decis Making. 2017;37(7), 735–746. Available from https://doi.org/10.1177/0272989X16686559

12. Krijkamp EM, Alarid-Escudero F, Enns EA, et al. Microsimulation Modeling for Health Decision Sciences Using R: A Tutorial. *Med Decis Making.* 2018;38(3):400–422. Available from https://doi.org/10.1177/0272989X18754513

13. Williams C, Lewsey JD, Briggs AH, et al. Cost-Effectiveness Analysis in R Using a Multi-state Modeling Survival Analysis Framework: A Tutorial. Med Decis Making. 2016;1–13. Available from https://doi.org/10.1177/0272989X16651869.

14. Taveras JL. R for Excel Users: An Introduction to R for Excel Analysts. CreateSpace Independent Publishing Platform; 2016.

15 Filipovic-Pierucci A, Zarca KID-Z., Markov Models For Health Economic Evaluation Modelling In R With The heemod Package, Value in Health, Volume 19, Issue 7, A369. Available from: https://arxiv.org/abs/1702.03252.

16. Grolemund G. Hands-On Programming with R, O'Reilly, 2014; 978-1449359010; https://rstudio-education.github.io/hopr/index.html

17. https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/licensing-a-repository

18. Hunink M, Weinstein M, Wittenberg E, et al. Decision Making in Health and Medicine: Integrating Evidence and Values. Cambridge University Press; 2014. doi:10.1017/CBO9781139506779.

19. Norris JR. Markov chains. No. 2. Cambridge university press; 1998; https://doi.org/10.1017/CBO9780511810633

20. Briggs, A. Decision Modelling for Health Economic Evaluation - Exercises. London School of Hygiene & Tropical Medicine, London, United Kingdom; 2022. https://doi.org/10.17037/DATA.00002980

21. Briggs A, Sculpher M. Introducing Markov models for economic evaluation. PharmacoEconomics 1998; 13(4): 397-409.

22. Briggs AH. Handling uncertainty in cost-effectiveness models. PharmacoEconomics 2000 May;17(5):479-500.

23. dplyr <-> base R. Tidyverse vignette. Package developed by Wickham H, François R, Henry L, Müller K, Rstudio. Available from https://dplyr.tidyverse.org/articles/base.html.

24. Incerti D, Jansen JP. hesim: Health Economic, Simulation Modeling and Decision Analysis. 2021. Available from https://arxiv.org/abs/2102.09437.

25. Wickham H. Advanced R. Chapman & Hall/CRC The R Series 2nd Edition; 2019; https://doi.org/10.1201/9781351201315

26. Allaire J, Xie Y, McPherson J, Luraschi J, Ushey K, Atkins A, Wickham H, Cheng J, Chang W, Iannone R. rmarkdown: Dynamic Documents for R. R package version 2.14. 2022. https://rmarkdown.rstudio.com.

27. Allaire J. quarto: R Interface to 'Quarto' Markdown Publishing System. R package version 1.2, 2022 https://CRAN.R-project.org/package=quarto.

28. Grolemund G, Wickham H. R Programming for Data Science. O'Reilly Media 1st Edition; 2016; 978-1491910399

29. Csardi G, Nepusz T. The igraph software package for complex network research, InterJournal, Complex Systems. 2006; 1695. https://igraph.org

30. Soetaert K. diagram: Functions for Visualising Simple Graphs (Networks), Plotting Flow Diagrams. R package version 1.6.5. 2020; Available from https://CRAN.R-project.org/package=diagram

31. Iannone R. DiagrammeR: Graph/Network Visualization. R package version. 2020;1.0.6.1. Available from https://CRAN.R-project.org/package=DiagrammeR

32. Spedicato GA. Discrete Time Markov Chains with R, The R Journal. 2017;9/2:84-104

6 Supplementary Material

Health Economic Evaluation using Markov Models in R for Microsoft Excel Users: A Tutorial

Nathan Green[1]* (PhD), Felicity Lamrock*[2] (PhD), Nichola Naylor[3,4] (PhD), Jack Williams[4] (PhD), Andrew Briggs[4] (DPhil)

1 Department of Statistical Science, University College London - UK

2 Mathematical Sciences Research Centre, Queen's University Belfast - UK

3 HCAI, Fungal, AMR, AMU and Sepsis Division, UK Health Security Agency, London, UK

4 Department of Health Services Research and Policy, London School of Hygiene & Tropical Medicine - UK

*joint first author

6.1.1 Online Resource 1 Markov model diagram R code using the *DiagrammeR* package

```
library(DiagrammeR)
library(diagram)

a_graph <-
  create_graph() %>%
  add_node(label = "Asymptomatic",
           node_aes = node_aes(fontsize = 5)) %>% # 1
  add_node(label = "Progressive \ndisease", from = 1,
           node_aes = node_aes(fontsize = 6),
           edge_aes = edge_aes(label = "tpProg*(1 - effect)", fontsize = 5, len = 0.5)) %>%
  add_node(label = "Dead", from = c(1,2),
           node_aes = node_aes(fontsize = 8),
           edge_aes = edge_aes(label = c("tpDn", "tpDcm + tpDn"), fontsize = 5, len = 0.5)) %>%
  add_edge(from = 1, to = 1, edge_aes = edge_aes(label = " ")) %>%
  add_edge(from = 2, to = 2) %>%
  add_edge(from = 3, to = 3)

render_graph(a_graph, layout = "nicely")
render_graph(a_graph, layout = "circle")
```

6.1.2 Online Resource 2 Basic Model R code

```
############################
# Markov model: real world #
############################

## model set-up ----

t_names <- c("without_drug", "with_drug")
n_treatments <- length(t_names)

s_names  <- c("Asymptomatic_disease", "Progressive_disease", "Dead")
n_states <- length(s_names)

n_cohort <- 1000

n_cycles <- 46
Initial_age <- 55

cAsymp <- 500
cDeath <- 1000
cDrug <- 1000
cProg <- 3000
uAsymp <- 0.95
uProg <- 0.75
oDr <- 0.06
cDr <- 0.06
tpDcm <- 0.15

# cost of staying in state
state_c_matrix <-
  matrix(c(cAsymp, cProg, 0,
           cAsymp + cDrug, cProg, 0),
         byrow = TRUE,
         nrow = n_treatments,
         dimnames = list(t_names,
                         s_names))

# qaly when staying in state
state_q_matrix <-
  matrix(c(uAsymp, uProg, 0,
           uAsymp, uProg, 0),
         byrow = TRUE,
         nrow = n_treatments,
         dimnames = list(t_names,
                         s_names))

# cost of moving to a state
# same for both treatments
trans_c_matrix <-
  matrix(c(0, 0, 0,
           0, 0, cDeath,
           0, 0, 0),
         byrow = TRUE,
         nrow = n_states,
         dimnames = list(from = s_names,
                         to = s_names))

# Transition probabilities ----

# Transition probabilities
```

```
p_matrix <- array(data = 0,
                  dim = c(n_states, n_states, n_treatments),
                  dimnames = list(from = s_names,
                                  to = s_names,
                                  t_names))

# Store population output for each cycle

# state populations
pop <- array(data = NA,
             dim = c(n_states, n_cycles, n_treatments),
             dimnames = list(state = s_names,
                             cycle = NULL,
                             treatment = t_names))

pop["Asymptomatic_disease", cycle = 1, ] <- n_cohort
pop["Progressive_disease", cycle = 1, ] <- 0
pop["Dead", cycle = 1, ] <- 0

# _arrived_ state populations
trans <- array(data = NA,
               dim = c(n_states, n_cycles, n_treatments),
               dimnames = list(state = s_names,
                               cycle = NULL,
                               treatment = t_names))

trans[, cycle = 1, ] <- 0


# Sum costs and QALYs for each cycle at a time for each drug

cycle_empty_array <-
  array(NA,
        dim = c(n_treatments, n_cycles),
        dimnames = list(treatment = t_names,
                        cycle = NULL))

cycle_state_costs <- cycle_trans_costs <- cycle_empty_array
cycle_costs <- cycle_QALYs <- cycle_empty_array
LE <- LYs <- cycle_empty_array    # life-expectancy; life-years
cycle_QALE <- cycle_empty_array   # qaly-adjusted life-years

total_costs <- setNames(c(NA, NA), t_names)
total_QALYs <- setNames(c(NA, NA), t_names)


# Time-dependent probability matrix ----

p_matrix_cycle <- function(p_matrix, age, cycle,
                           tpProg = 0.01,
                           tpDcm = 0.15,
                           effect = 0.5) {

  tpDn_lookup <-
    c("(34,44]" = 0.0017,
      "(44,54]" = 0.0044,
      "(54,64]" = 0.0138,
      "(64,74]" = 0.0379,
      "(74,84]" = 0.0912,
      "(84,100]" = 0.1958)
```

```
  age_grp <- cut(age, breaks = c(34,44,54,64,74,84,100))

  tpDn <- tpDn_lookup[age_grp]

  # Matrix containing transition probabilities for without_drug

  p_matrix["Asymptomatic_disease", "Progressive_disease", "without_drug"]
<- tpProg*cycle

  p_matrix["Asymptomatic_disease", "Dead", "without_drug"] <- tpDn

  p_matrix["Asymptomatic_disease", "Asymptomatic_disease", "without_drug"]
<- 1 - tpProg*cycle - tpDn

  p_matrix["Progressive_disease", "Dead", "without_drug"] <- tpDcm + tpDn

  p_matrix["Progressive_disease", "Progressive_disease", "without_drug"] <-
1 - tpDcm - tpDn

  p_matrix["Dead", "Dead", "without_drug"] <- 1

  # Matrix containing transition probabilities for with_drug

  p_matrix["Asymptomatic_disease", "Progressive_disease", "with_drug"] <-
tpProg*(1 - effect)*cycle

  p_matrix["Asymptomatic_disease", "Dead", "with_drug"] <- tpDn

  p_matrix["Asymptomatic_disease", "Asymptomatic_disease", "with_drug"] <-
    1 - tpProg*(1 - effect)*cycle - tpDn

  p_matrix["Progressive_disease", "Dead", "with_drug"] <- tpDcm + tpDn

  p_matrix["Progressive_disease", "Progressive_disease", "with_drug"] <- 1
- tpDcm - tpDn

  p_matrix["Dead", "Dead", "with_drug"] <- 1

  return(p_matrix)
}


## Run model ----

for (i in 1:n_treatments) {

  age <- Initial_age

  for (j in 2:n_cycles) {

    p_matrix <- p_matrix_cycle(p_matrix, age, j - 1)

    pop[, cycle = j, treatment = i] <-
      pop[, cycle = j - 1, treatment = i] %*% p_matrix[, , treatment = i]

    trans[, cycle = j, treatment = i] <-
      pop[, cycle = j - 1, treatment = i] %*% (trans_c_matrix * p_matrix[,
, treatment = i])

    age <- age + 1
```

```
  }

  cycle_state_costs[i, ] <-
    (state_c_matrix[treatment = i, ] %*% pop[, , treatment = i]) * 1/(1 +
cDr)^(1:n_cycles - 1)

  # discounting at _previous_ cycle
  cycle_trans_costs[i, ] <-
    (c(1,1,1) %*% trans[, , treatment = i]) * 1/(1 + cDr)^(1:n_cycles - 2)

  cycle_costs[i, ] <- cycle_state_costs[i, ] + cycle_trans_costs[i, ]

  LE[i, ] <- c(1,1,0) %*% pop[, , treatment = i]

  LYs[i, ] <- LE[i, ] * 1/(1 + oDr)^(1:n_cycles - 1)

  cycle_QALE[i, ] <-
    state_q_matrix[treatment = i, ] %*% pop[, , treatment = i]

  cycle_QALYs[i, ] <- cycle_QALE[i, ] * 1/(1 + oDr)^(1:n_cycles - 1)

  total_costs[i] <- sum(cycle_costs[treatment = i, -1])
  total_QALYs[i] <- sum(cycle_QALYs[treatment = i, -1])
}


## Plot results ----

# Incremental costs and QALYs of with_drug vs to without_drug
c_incr <- total_costs["with_drug"] - total_costs["without_drug"]
q_incr <- total_QALYs["with_drug"] - total_QALYs["without_drug"]

# Incremental cost-effectiveness ratio
ICER <- c_incr/q_incr

plot(x = q_incr/n_cohort, y = c_incr/n_cohort,
     xlim = c(0, 1100/n_cohort),
     ylim = c(0, 10e6/n_cohort),
     pch = 16, cex = 1.5,
     xlab = "QALY difference",
     ylab = paste0("Cost difference (", enc2utf8("\u00A3"), ")"),
     frame.plot = FALSE)
abline(a = 0, b = 30000) # Willingness-to-pay threshold
```

Commented [FL1]: I added a - to cost-effectiveness to be in line with the text

Commented [FL2]: Is this right? £?

Commented [FL3]: I was going to change this to 20000?

6.1.3 Online Resource 3 PSA R code

```
#############################################
# Probability Sensitivity Analysis (PSA)

ce_markov <- function(start_pop,
                      p_matrix,
                      state_c_matrix,
                      trans_c_matrix,
                      state_q_matrix,
                      n_cycles = 46,
                      init_age = 55,
                      s_names = NULL,
                      t_names = NULL) {

  n_states <- length(start_pop)
  n_treat <- dim(p_matrix)[3]

  pop <- array(data = NA,
               dim = c(n_states, n_cycles, n_treat),
               dimnames = list(state = s_names,
                               cycle = NULL,
                               treatment = t_names))
  trans <- array(data = NA,
                 dim = c(n_states, n_cycles, n_treat),
                 dimnames = list(state = s_names,
                                 cycle = NULL,
                                 treatment = t_names))

  for (i in 1:n_states) {
    pop[i, cycle = 1, ] <- start_pop[i]
  }

  cycle_empty_array <-
    array(NA,
          dim = c(n_treat, n_cycles),
          dimnames = list(treatment = t_names,
                          cycle = NULL))

  cycle_state_costs <- cycle_trans_costs <- cycle_empty_array
  cycle_costs <- cycle_QALYs <- cycle_empty_array
  LE <- LYs <- cycle_empty_array    # life-expectancy; life-years
  cycle_QALE <- cycle_empty_array   # qaly-adjusted life-years

  total_costs <- setNames(rep(NA, n_treat), t_names)
  total_QALYs <- setNames(rep(NA, n_treat), t_names)

  for (i in 1:n_treat) {

    age <- init_age

    for (j in 2:n_cycles) {

      # difference from point estimate case
      # pass in functions for random sample
      # rather than fixed values
      p_matrix <- p_matrix_cycle(p_matrix, age, j - 1,
                                 tpProg = tpProg(),
                                 tpDcm = tpDcm(),
                                 effect = effect())

      # Matrix multiplication
```

```
      pop[, cycle = j, treatment = i] <-
        pop[, cycle = j - 1, treatment = i] %*% p_matrix[, , treatment = i]

      trans[, cycle = j, treatment = i] <-
        pop[, cycle = j - 1, treatment = i] %*% (trans_c_matrix *
p_matrix[, , treatment = i])

      age <- age + 1
    }

    cycle_state_costs[i, ] <-
      (state_c_matrix[treatment = i, ] %*% pop[, , treatment = i]) * 1/(1 +
cDr)^(1:n_cycles - 1)

    cycle_trans_costs[i, ] <-
      (c(1,1,1) %*% trans[, , treatment = i]) * 1/(1 + cDr)^(1:n_cycles -
2)

    cycle_costs[i, ] <- cycle_state_costs[i, ] + cycle_trans_costs[i, ]

    LE[i, ] <- c(1,1,0) %*% pop[, , treatment = i]

    LYs[i, ] <- LE[i, ] * 1/(1 + oDr)^(1:n_cycles - 1)

    cycle_QALE[i, ] <-
     state_q_matrix[treatment = i, ] %*%  pop[, , treatment = i]

    cycle_QALYs[i, ] <- cycle_QALE[i, ] * 1/(1 + oDr)^(1:n_cycles - 1)

    total_costs[i] <- sum(cycle_costs[treatment = i, -1])
    total_QALYs[i] <- sum(cycle_QALYs[treatment = i, -1])
  }

  list(pop = pop,
       cycle_costs = cycle_costs,
       cycle_QALYs = cycle_QALYs,
       total_costs = total_costs,
       total_QALYs = total_QALYs)
}


# replace point values with functions to random sample

cAsymp <- function() rnorm(1, 500, 127.55)
cDeath <- function() rnorm(1, 1000, 255.11)
cDrug  <- function() rnorm(1, 1000, 102.04)
cProg  <- function() rnorm(1, 3000, 510.21)
effect <- function() rnorm(1, 0.5, 0.051)
tpDcm  <- function() rbeta(1, 29, 167)
tpProg <- function() rbeta(1, 15, 1506)
uAsymp <- function() rbeta(1, 69, 4)
uProg  <- function() rbeta(1, 24, 8)


# Define cost and QALYs as functions

state_c_matrix <- function() {
  matrix(c(cAsymp(), cProg(), 0,            # without drug
           cAsymp() + cDrug(), cProg(), 0), # with drug
           byrow = TRUE,
           nrow = n_treatments,
```

```
           dimnames = list(t_names,
                           s_names))
}

state_q_matrix <- function() {
  matrix(c(uAsymp(), uProg(), 0,  # without drug
           uAsymp(), uProg(), 0), # with drug
         byrow = TRUE,
         nrow = n_treatments,
         dimnames = list(t_names,
                         s_names))
}

trans_c_matrix <- function() {
  matrix(c(0, 0, 0,          # Asymptomatic_disease
           0, 0, cDeath(),  # Progressive_disease
           0, 0, 0),         # Dead
         byrow = TRUE,
         nrow = n_states,
         dimnames = list(from = s_names,
                         to = s_names))
}


## Run PSA analysis ----

n_trials <- 500

costs <- matrix(NA, nrow = n_trials, ncol = n_treatments,
                dimnames = list(NULL, t_names))
qalys <- matrix(NA, nrow = n_trials, ncol = n_treatments,
                dimnames = list(NULL, t_names))

for (i in 1:n_trials) {
  ce_res <- ce_markov(start_pop = c(n_cohort, 0, 0),
                      p_matrix,
                      state_c_matrix(),
                      trans_c_matrix(),
                      state_q_matrix())

  costs[i, ] <- ce_res$total_costs
  qalys[i, ] <- ce_res$total_QALYs
}


## Plot results ----

# incremental costs and QALYs of with_drug vs to without_drug
c_incr_psa <- costs[, "with_drug"] - costs[, "without_drug"]
q_incr_psa <- qalys[, "with_drug"] - qalys[, "without_drug"]

plot(x = q_incr_psa/n_cohort, y = c_incr_psa/n_cohort,
     xlim = c(0, 2),
     ylim = c(0, 15e3),
     pch = 16, cex = 1.2,
     col = "grey",
     xlab = "QALY difference",
     ylab = paste0("Cost difference (", enc2utf8("\u00A3"), ")"),
     frame.plot = FALSE)
abline(a = 0, b = 30000, lwd = 2) # Willingness-to-pay threshold
?30,000/QALY
```

```
points(x = q_incr/n_cohort, y = c_incr/n_cohort,
       col = "red",
       pch = 16, cex = 1.5)
```

Input table

Name	**Value**	**Variable description**
cAsymp	500.00	*Cost of one cycle in the asymptomatic disease state*
cDeath	1000	*Cost associated with transition to the dead state*
cDrug	1000	*Cost of drug for one cycle*
tpDcm	0.15	*Probability of dying from the disease in a single cycle*
cProg	3000	*Cost of one cycle in the progressive disease state*
effect	50%	*Effectiveness of drug in terms of reducing disease progression*
tpProg	0.010	*Coefficent of increase for probability of entering the progressive disease state*
uAsymp	0.95	*Quality of life weight for one cycle in the asymptomatic disease state*
uProg	0.75	*Quality of life weight for one cycle in the progressive disease state*
cycle	1	*Length in years of one cycle*
ini_age	55	*The initial age at which patients are deeemed to start the model*
cDR	6%	*Discount rate for costs*
oDR	6%	*Discount rate for outcomes*
nD35	0.0017	*Natural death risk for over 35's (from standard life-tables)*
nD45	0.0044	*Natural death risk for over 45's (from standard life-tables)*
nD55	0.0138	*Natural death risk for over 55's (from standard life-tables)*
nD65	0.0379	*Natural death risk for over 65's (from standard life-tables)*
nD75	0.0912	*Natural death risk for over 75's (from standard life-tables)*
nD85	0.1958	*Natural death risk for over 85's (from standard life-tables)*

Output table

Strategy	**Eff**	**Cost**	**ΔEff**	**ΔCost**	**ICER**
No drug	7.76	£ 9,265			
Drug	8.62	£ 16,155	0.87	£ 6,891	£ 7,931