R-to-Excel conversion proof-of-concept using a cohort Markov model

Howard Thom, Queena Wang, Edna Keeney, Chris Fawsitt, Michael O'Donnell

Clifton Insight, United Kingdom

Background

The R programming language has become increasingly prominent in health economic cost-effectiveness modelling.[1-3] It has substantial advantages over spreadsheet software, such as Microsoft Excel, in transparency, efficiency and flexibility. These are further boosted by the RStudio interface for R programming, which allows the creation of 'Projects' grouping scripts and data together; these projects can be hosted on the GitHub repository to ease sharing and reuse of code. The RMarkdown package for R enables the automatic creation of reports as PDF, Microsoft Word or HTML documents, further improving the transparency and flexibility of R. Finally, the RShiny package enables the creation of graphical user interfaces (GUI) for R models which can be hosted online and/or accessed by non-R users.[4 5]

The R for Health Technology Assessment (HTA) and Decision Analysis in R for Technologies in Health (DARTH) organisations have created websites filled with tutorials and example code on the use of R in HTA. The R for HTA organisation runs annual workshops on the latest developments in R for HTA. Tutorials and packages have been published for Bayesian cost-effectiveness analysis, cohort Markov models, semi-Markov models and state-transition microsimulation models.[6-9] The DARTH group have published guidelines on transparent design of RStudio projects for cost-effectiveness models, separating data and code scripts into sensible categories.[10] The TidyVerse Style Guide has also developed standards for spacing, tabs, variable names, and function names which have been encouraged by the R for HTA organisation to improve transparency and reusability of R analyses.[11]

The Zorginstituutnederland (ZIN) in the Netherlands have also organised a working group of HTA agencies interested in the use of R in their manufacturer submissions. The group includes representatives from Canada (CADTH), Sweden (TLV), UK (NICE), Germany (IQWiG), Ireland (NCPE), Hungary (OGYEI), and Slovakia (NIHO). Together, they proposed guidelines on the use of R in submissions.[12] These included recommended directory structure, coding style, documentation, and functionality. They also recommended the use of the 'testthat' package for unit testing of model components and structures, which is effectively implemented through R packages. Although RShiny interfaces were permitted, they were not required. Finally, they did not make it mandatory to submit models in R but only aimed to encourage such submissions.

Despite these advantages and the establishment of standards for R in HTA, many agencies do not yet accept R models or have only limited capacity for assessing R models. This paper explains a methodology for implementing R cohort Markov models in such a way that automatic generation of an Excel version is feasible. This would allow country affiliates in jurisdications that do not currently accept R models to still have an Excel version for submissions. It would furthermore improve transparancy as users familiar with Excel but not R could explore the technical functioning of the model.

Methods

Markov models

The models explored in this proof-of-concept are discrete-time cohort Markov models, which are among the most commonly used models in economic evaluation of healthcare interventions.[13 14] These characterise possible patient health status with a set of discrete and mutually exclusive states. This focus on states makes them particularly suited to modelling chronic diseases such as psoriatic arthritis, multiple sclerosis or Alzheimer's.[15-17] They are a special case of multistate models, with a restriction of event times to the end of discrete cycles, simulation of a cohort instead of individuals, and making the Markov assumption that future progression depends only on current state occupancy. [18 19] We only consider fully probabilistic models where probabilistic sensitivity analysis is the default analysis.[20 21]

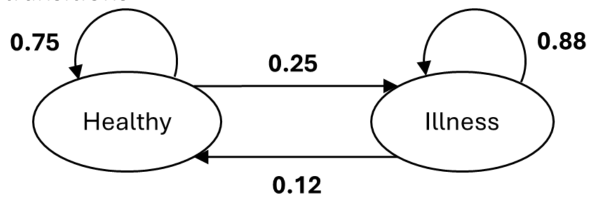
To implement the core of a Markov model in R it is essential to understand that it relies on matrix multiplication. The state occupancy at time t is represented by a cohort vector π_t, with each element representing proportion of the cohort in the corresponding state. This cohort vector is calculated by multiplying the vector at the previous time point (π_{t-1}) by the probability transition matrix P.

$$\pi_t = \pi_{t-1}P$$

The elements of the transition matrix represent the probability of transitioning from one state to another, as illustrated in Figure 1. Transition probabilities can be made time dependent, for example to capture changes in treatment effect or mortality, by adding a time index to the matrix P_t, giving a time-inhomogeneous Markov model.

Figure 1 Representation of Markov model using transition matrix

The Markov model with transitions



Is represented by the transition matrix

$$\begin{pmatrix} 0.75 & 0.25 \\ 0.12 & 0.88 \end{pmatrix}$$

The first row represents the probability of transitioning from healthy to healthy (0.75) or illness (0.25); the second row represents the probability of transitioning from illness to healthy (0.12) or illness (0.88).

This matrix multiplication is implemented in R using base R matrices and vectors, as illustrated with sample code for a probabilistic time-inhomogeneous Markov model in Code block 1. Base R is preferred over the package ‘heemod’ as it has greater efficiency, and over ‘hesim’ to allow for greater flexibility and transparency.[8 22] Matrices and arrays are preferred over data frames, data tables, or Tibbles due to much faster access times and the ability to use the efficient %*% matrix multiplication operator of R; the advantages of multidimensional arrays have been described in the literature.[6] The approach follows the R implementation of open-source models in atrial fibrillation, coeliac disease, and surgical procedure modelling.[23-26] The NICE Atrial Fibrillation guidelines model utilised this same approach. [27] It furthermore aligns with the approach outlined in recent tutorials for Markov modelling in R.[28 29] This matrix multiplication is distinct from implementation in Excel which typically writes out the underlying calculations cell-by-cell, which greatly reduces readability and ease of adaptation.[30]

*Code block 1 R code for the core Markov loop including probabilistic sensitivity analysis and a time-inhomogeneous transition matrix indexed by the cycle length i_cycle.**

```
# Loop over the treatment options
for (i_treatment in 1:self$n_treatments)
{
        # Loop over the PSA samples
        for (i_sample in 1:self$n_samples)
        {
                # Loop over the cycles
                # Cycle 1 is already defined so only need to update cycles 2:n_cycles
                for (i_cycle in 2:self$n_cycles)
                {
                        # Multiply previous cycle's cohort vector by transition matrix
                                a_cohort_array[i_treatment, i_sample, i_cycle, ] <-
                                a_cohort_array[i_treatment, i_sample, i_cycle-1, ] %*%
                                a_transition_matrices[i_treatment, i_sample, i_cycle, , ]
                }               }
                # Calculate cycle costs and QALYs for this PSA sample
                # Calculate total costs and QALYs for this PSA sample
        }
}
self$a_cohort_array <- a_cohort_array
self$a_transition_matrices <- a_transition_matrices
```

*Implemented within the markov_model R6 class. The 'self' object accesses the contents of the instantiated object, thus giving the object's number of treatments, samples and cycles.

Structuring the implementation

An R package named "R-to-Excel-POC" is used instead of simply an RStudio project due to the standardisation of structure, documentation and testing.[31] The structure of an R package was furthermore extended by DARTH to a framework for organising economic evaluations, including models, in R.[10] The proof-of-concept follows this DARTH framework with specific folders for data, functions and analysis, as explained in Table 1.

Table 1 Structure of the proof-of-concept package following the DARTH Framework for cost-effectiveness models

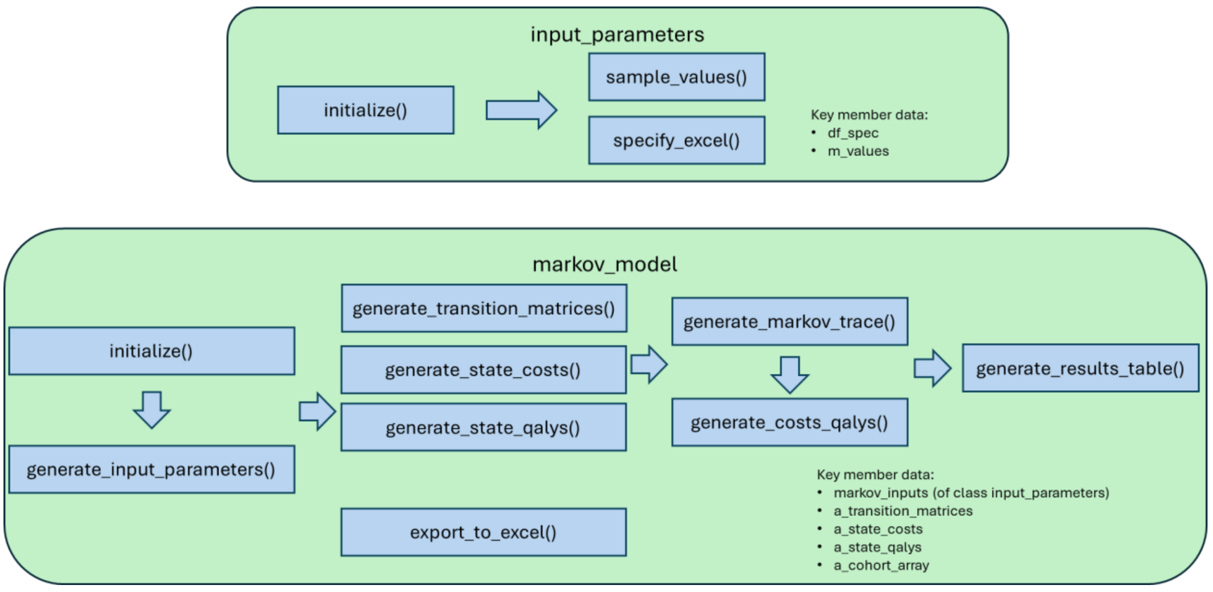
Folder	Description
R	Functions, such as those to generate input parameters or survival curves, with their uncompiled ROxygen2 documentation
Analysis	Scripts that call functions in the 'R' folder to analyse data in the 'data' folder, including a main.R script
Output	Interim and final output of the model generated by scripts in the 'analysis' folder, including PSA samples or CEACs
Man	Additional folder storing .Rd files generated by the functions in 'R' and R2oxygen*

The model is furthermore implemented in the Object Oriented Programming paradigm using R6 classes. Each model is an object with member functions and data which are called by the user. An advantage is increased ease of usage as users could call (say) markov_model::new() to create a new Markov model. The primary advantage for our purposes is ease of organisation as all member functions of a class have access to a shared set of member data. OOP has been adopted with success in the 'hesim' R package.[8]

Two classes are included in the package and they are illustrated in Figure 2. The first is the input_parameters class, which stores specification of random parameters and sampled values. The second is markov_model, which includes a markov_inputs member object of class input_parameters. The markov_model class includes member data describing the model and member functions allowing for generation of input parameters, transition matrices, state costs, state QALYs, the Markov trace, total costs and QALYs, and an overall results table. A final member function export_to_excel is described below.

The members functions of the two classes are illustrated in Figure 2. Each function is documented using ROxygen2 generated help files. Note that documentation is stored in the 'man' folder of the R package, so this folder is added to the DARTH framework of Table 1. The initialize() function is required by each class and is called when using input_parameters$new() or markov_model$new() to create new objects of each class. The arrows indicate that outputs of the earlier functions are used by later functions. Outputs used by later functions are saved as member data of the classes. For example, in markov_model objects the markov_inputs object is created by the generate_input_parameters() function, the a_transition_matrices array is created by the generate_transition_matrices() function, and the a_cohort_array array is created by the generate_markov_trace() function using the a_transition_matrices array and code similar to Code block 1.

Figure 2 Illustration of the two classes, member functions, and key member data for the Markov model implementation



Finally, a script named smoking_main.R in the 'analysis' folder runs the functions in the R-to-Excel-POC package, the outputs of which could be analysed with other packages such as BCEA.[9]

Methods for conversion from R to Excel

The key requirements of this process are:

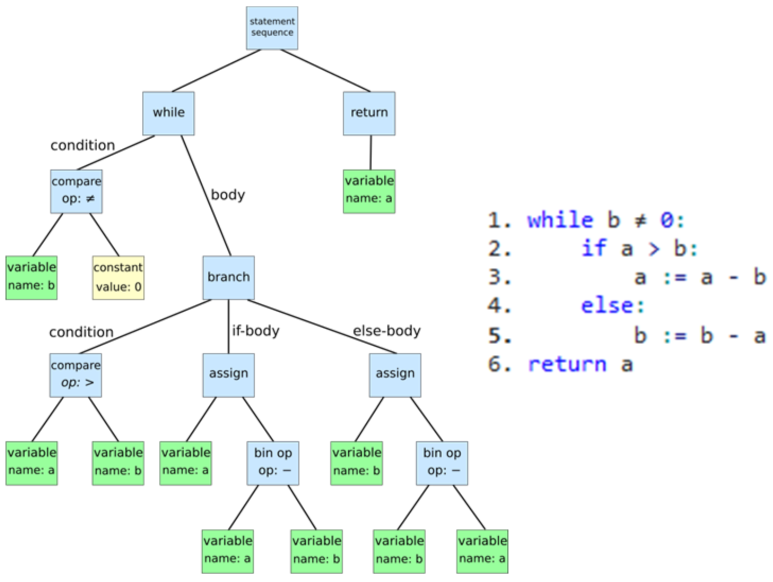
- Generalisable to any Markov model
- Generates a human-readable Excel file suitable for experimentation and for HTA submissions
- Avoids “double programming” the model in both Excel and R

The reason to avoid “doublet programming” is that it duplicates work as all model structures (e.g., number of states, allowed transitions, Markov trace calculations, parameter distributions for PSA) would need to be implemented in both templates. This would be error prone and labour intensive.

General approach through abstract syntax tree

At the University of Bristol Dr Thom is leading the REEEVR grant (UK Medical Research Grant MR/W029855/1) which is developing a tool to automatically convert models from Excel into R. This is being completed with Dr Michael O’Donnell. The REEEVR tool has successfully converted probabilistic Markov models from Excel into R. The converted models run up to 100 times faster than the Excel code and reproduce the exact results. The REEEVR tool is a fully general solution for code conversion as it utilises an abstract syntax tree (AST). This is a language-agnostic representation of the meaning of a computer program and is illustrated for a pseudocode while loop in Figure 3. The REEEVR converted traverses the Excel sheet, breaks code into components, and then represents each component with an AST. Each ‘leaf’ of the AST is an elementary component of the program. The final step is for REEEVR to run through the JSON file and select R equivalents of each Excel function.

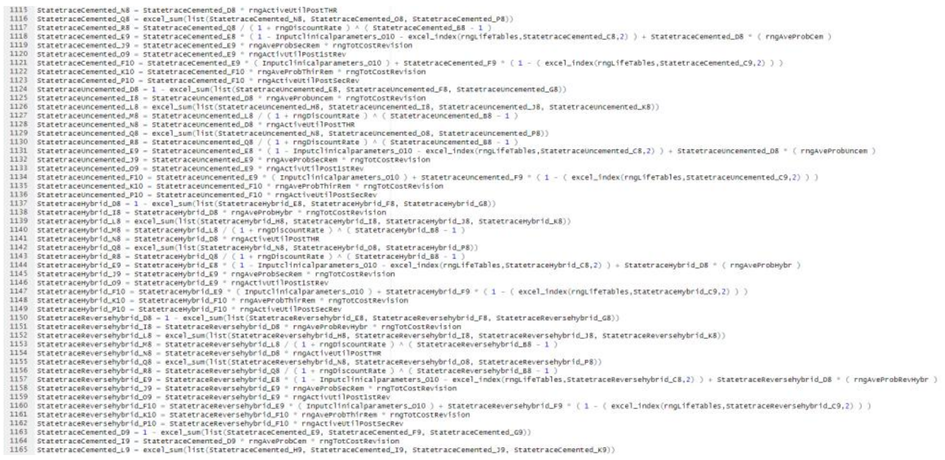
Figure 3 Illustration of abstract syntax tree representation of a programming language



```
while b ≠ 0:
    if a > b:
        a := a - b
    else:
        b := b - a
return a
```

A sample of the output R code of REEEVR is illustrated in Figure 4. The immediate difficulty with this output is that the code is not human readable. The objective of REEEVR was to accelerate Excel models rather than to provide an R model that could be adapted further by users.

*Figure 4 Output R code of the REEEVR tool representing a 4-state probabilistic Markov model**



```
StatetraceCemented_N8 = StatetraceCemented_D8 * rngActiveUtilPostTHR
StatetraceCemented_Q8 = excel_sum(list(StatetraceCemented_N8, StatetraceCemented_O8, StatetraceCemented_P8))
StatetraceCemented_R8 = StatetraceCemented_Q8 / ( 1 + rngDiscountRate ) ^ ( StatetraceCemented_B8 - 1 )
StatetraceCemented_E9 = StatetraceCemented_E8 * ( 1 - Inputclinicalparameters_O10 - excel_index(rngLifeTables,StatetraceCemented_C8,2) ) + StatetraceCemented_D8 * ( rngAveProbCem )
StatetraceCemented_J9 = StatetraceCemented_E9 * rngAveProbSecRem * rngTotCostRevision
StatetraceCemented_O9 = StatetraceCemented_E9 * rngActivUtilPost1stRev
StatetraceCemented_F10 = StatetraceCemented_E9 * ( Inputclinicalparameters_O10 ) + StatetraceCemented_F9 * ( 1 - ( excel_index(rngLifeTables,StatetraceCemented_C9,2) ) )
StatetraceCemented_K10 = StatetraceCemented_F10 * rngAveProbThirRem * rngTotCostRevision
StatetraceCemented_P10 = StatetraceCemented_F10 * rngActiveUtilPostSecRev
StatetraceUncemented_D8 = 1 - excel_sum(list(StatetraceUncemented_E8, StatetraceUncemented_F8, StatetraceUncemented_G8))
StatetraceUncemented_I8 = StatetraceUncemented_D8 * rngAveProbUncem * rngTotCostRevision
StatetraceUncemented_L8 = excel_sum(list(StatetraceUncemented_H8, StatetraceUncemented_I8, StatetraceUncemented_J8, StatetraceUncemented_K8))
StatetraceUncemented_M8 = StatetraceUncemented_L8 / ( 1 + rngDiscountRate ) ^ ( StatetraceUncemented_B8 - 1 )
StatetraceUncemented_N8 = StatetraceUncemented_D8 * rngActiveUtilPostTHR
StatetraceUncemented_Q8 = excel_sum(list(StatetraceUncemented_N8, StatetraceUncemented_O8, StatetraceUncemented_P8))
StatetraceUncemented_R8 = StatetraceUncemented_Q8 / ( 1 + rngDiscountRate ) ^ ( StatetraceUncemented_B8 - 1 )
StatetraceUncemented_E9 = StatetraceUncemented_E8 * ( 1 - Inputclinicalparameters_O10 - excel_index(rngLifeTables,StatetraceUncemented_C8,2) ) + StatetraceUncemented_D8 * ( rngAveProbUncem )
StatetraceUncemented_J9 = StatetraceUncemented_E9 * rngAveProbSecRem * rngTotCostRevision
StatetraceUncemented_O9 = StatetraceUncemented_E9 * rngActivUtilPost1stRev
StatetraceUncemented_F10 = StatetraceUncemented_E9 * ( Inputclinicalparameters_O10 ) + StatetraceUncemented_F9 * ( 1 - ( excel_index(rngLifeTables,StatetraceUncemented_C9,2) ) )
StatetraceUncemented_K10 = StatetraceUncemented_F10 * rngAveProbThirRem * rngTotCostRevision
StatetraceUncemented_P10 = StatetraceUncemented_F10 * rngActiveUtilPostSecRev
StatetraceHybrid_D8 = 1 - excel_sum(list(StatetraceHybrid_E8, StatetraceHybrid_F8, StatetraceHybrid_G8))
StatetraceHybrid_I8 = StatetraceHybrid_D8 * rngAveProbHybr * rngTotCostRevision
StatetraceHybrid_L8 = excel_sum(list(StatetraceHybrid_H8, StatetraceHybrid_I8, StatetraceHybrid_J8, StatetraceHybrid_K8))
StatetraceHybrid_M8 = StatetraceHybrid_L8 / ( 1 + rngDiscountRate ) ^ ( StatetraceHybrid_B8 - 1 )
StatetraceHybrid_N8 = StatetraceHybrid_D8 * rngActiveUtilPostTHR
StatetraceHybrid_Q8 = excel_sum(list(StatetraceHybrid_N8, StatetraceHybrid_O8, StatetraceHybrid_P8))
StatetraceHybrid_R8 = StatetraceHybrid_Q8 / ( 1 + rngDiscountRate ) ^ ( StatetraceHybrid_B8 - 1 )
StatetraceHybrid_E9 = StatetraceHybrid_E8 * ( 1 - Inputclinicalparameters_O10 - excel_index(rngLifeTables,StatetraceHybrid_C8,2) ) + StatetraceHybrid_D8 * ( rngAveProbHybr )
StatetraceHybrid_J9 = StatetraceHybrid_E9 * rngAveProbSecRem * rngTotCostRevision
StatetraceHybrid_O9 = StatetraceHybrid_E9 * rngActivUtilPost1stRev
StatetraceHybrid_F10 = StatetraceHybrid_E9 * ( Inputclinicalparameters_O10 ) + StatetraceHybrid_F9 * ( 1 - ( excel_index(rngLifeTables,StatetraceHybrid_C9,2) ) )
StatetraceHybrid_K10 = StatetraceHybrid_F10 * rngAveProbThirRem * rngTotCostRevision
StatetraceHybrid_P10 = StatetraceHybrid_F10 * rngActiveUtilPostSecRev
StatetraceReversehybrid_D8 = 1 - excel_sum(list(StatetraceReversehybrid_E8, StatetraceReversehybrid_F8, StatetraceReversehybrid_G8))
StatetraceReversehybrid_I8 = StatetraceReversehybrid_D8 * rngAveProbRevHybr * rngTotCostRevision
StatetraceReversehybrid_L8 = excel_sum(list(StatetraceReversehybrid_H8, StatetraceReversehybrid_I8, StatetraceReversehybrid_J8, StatetraceReversehybrid_K8))
StatetraceReversehybrid_M8 = StatetraceReversehybrid_L8 / ( 1 + rngDiscountRate ) ^ ( StatetraceReversehybrid_B8 - 1 )
StatetraceReversehybrid_N8 = StatetraceReversehybrid_D8 * rngActiveUtilPostTHR
StatetraceReversehybrid_Q8 = excel_sum(list(StatetraceReversehybrid_N8, StatetraceReversehybrid_O8, StatetraceReversehybrid_P8))
StatetraceReversehybrid_R8 = StatetraceReversehybrid_Q8 / ( 1 + rngDiscountRate ) ^ ( StatetraceReversehybrid_B8 - 1 )
StatetraceReversehybrid_E9 = StatetraceReversehybrid_E8 * ( 1 - Inputclinicalparameters_O10 - excel_index(rngLifeTables,StatetraceReversehybrid_C8,2) ) + StatetraceReversehybrid_D8 * ( rngAveProbRevHybr )
StatetraceReversehybrid_J9 = StatetraceReversehybrid_E9 * rngAveProbSecRem * rngTotCostRevision
StatetraceReversehybrid_O9 = StatetraceReversehybrid_E9 * rngActivUtilPost1stRev
StatetraceReversehybrid_F10 = StatetraceReversehybrid_E9 * ( Inputclinicalparameters_O10 ) + StatetraceReversehybrid_F9 * ( 1 - ( excel_index(rngLifeTables,StatetraceReversehybrid_C9,2) ) )
StatetraceReversehybrid_K10 = StatetraceReversehybrid_F10 * rngAveProbThirRem * rngTotCostRevision
StatetraceReversehybrid_P10 = StatetraceReversehybrid_F10 * rngActiveUtilPostSecRev
StatetraceCemented_D9 = 1 - excel_sum(list(StatetraceCemented_E9, StatetraceCemented_F9, StatetraceCemented_G9))
StatetraceCemented_I9 = StatetraceCemented_D9 * rngAveProbCem * rngTotCostRevision
StatetraceCemented_L9 = excel_sum(list(StatetraceCemented_H9, StatetraceCemented_I9, StatetraceCemented_J9, StatetraceCemented_K9))
```

*Full scrip consists of 2614 lines of R code

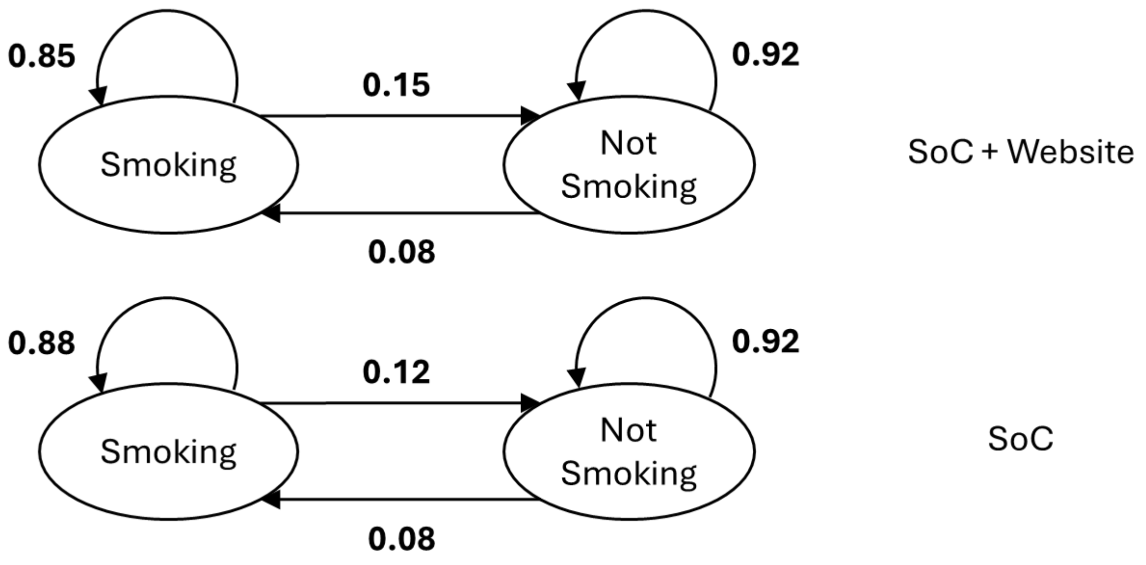
While this process of AST and code conversion could be applied to translate an R model into an Excel sheet the output would be similarly unreadable to Figure 4. Each line of R code would be stored in a separate Excel cell. The lack of for or apply loops in Excel would mean that every such statement would be unrolled in the Excel sheet. This would certainly not provide an Excel version of the R code that could be submitted to HTA agencies or used by even experienced Excel modellers.

We therefore do not recommend the general AST approach and instead opt for direct generation of Excel formulae through R.

Direct Excel formula generation

Our method builds Excel generation into the R model from the beginning, rather than adding this as functionality at a later stage. Our illustrative example is a probabilistic 2-state Markov model comparing a website to standard of care (SoC) to assist patients to quit smoking (Figure 5).

*Figure 5 Markov smoking example with average probabilities that are captured by Beta distributions in the code**



*SoC=Standard of Care

The input parameters object (Figure 2) includes a df_spec dataframe that fully specifies the distributions used in either R or Excel. The contents of this dataframe, exported by the openxlsx2::wb_add_data() function into an Excel sheet, are presented in Table 2. Distributions for transition probabilities, costs, utilities are fully specified by the v_distribution, hp_1 and hp_2 columns, which represent the distribution type, first hyperparameter and second hyperparameter, respectively. In the case of a Normal distribution hyperparameters would be the mean and SD. For a beta they would be the alpha and beta parameters. It will be possible to include alternative parameterisations (e.g., a v_distribution value of beta_mean_sd where hp_1 and hp_2 would be the mean and SD and the actual alpha and beta would be calculated using a method of moments).

The most important element in this data frame is the excel_formulae column that is generated by code similar to that in Code block 2. This code uses the paste0() function to concatenate string operations on Excel cells for the appropriate distribution (if random) or single cell (if fixed).

*Table 2 The df_spec dataframe of the input_parameters object in the Markov smoking application, exported to Excel**

	E	F	G	H	I	J	K	L	M	N	O	P
8	v_names	v_descriptions	v_type	v_distributions	hp_1	hp_2	from	to	v_treatment	v_state	excel_formulae	excel_value_location
9	Probability quit smoking website	Probability of quitting if on smoking website, follows a beta distribution	transition_probability	beta	15	85	1	2	2	1	=BETAINV(RAND(), I9, J9)	input_parameters!O9
10	Probability quit smoking SoC	Probability of quitting smoking if on SoC, follows a beta distribution	transition_probability	beta	12	88	1	2	1	1	=BETAINV(RAND(), I10, J10)	input_parameters!O10
11	Probability relapse	Probability relapse, which is same across treatments and follows a beta distribution	transition_probability	beta	8	92	2	1		2	=BETAINV(RAND(), I11, J11)	input_parameters!O11
12	Utility smoking	Utility smoking, follows a Normal distribution	utility	normal	0.95	0.02				1	=NORMINV(RAND(), I12, J12)	input_parameters!O12
13	Utility not smoking	Utility not smoking, follows a Normal distribution	utility	fixed	1					2	=I13	input_parameters!O13
14	Cost website	Cost website, fixed value and model assumes no cost of SoC and no state costs	one_off_cost	fixed	50				2		=I14	input_parameters!O14
15	Cost GP smoking	Cost of 6-monthly, on average, GP visit (£49 from PRSSU) for smoking related illness, follows Normal distribution	cost	normal	49	2				1	=NORMINV(RAND(), I15, J15)	input_parameters!O15
16	Cost statin smoking	Cost of roughly 20% of smokers taking statins (pravastatin at £3.45 per month), follows Normal distribution	cost	normal	0.69	0.069				1	=NORMINV(RAND(), I16, J16)	input_parameters!O16

*The states used in from and to columns a 1=Smoking and 2=Not smoking; the treatments are 1=SoC and 2=SoC+Website. Column O is omitted and stores the live value used in the model.

*The states used in from and to columns a 1=Smoking and 2=Not smoking; the treatments are 1=SoC and 2=SoC+Website. Column O is omitted and stores the live value used in the model.

Code block 2 Code to automatically generate the formula for distribution on an input parameter

Code

```
# First parameter
i_parameter <- 1
# Starting position in Excel sheet
startRow = 5
startCol = 8

# In which column letters would the hyper parameters be stored
hp_1_col <- openxlsx2::int2col(startCol + 1)
hp_2_col <- openxlsx2::int2col(startCol + 2)

paste0("NORMINV(RAND(), ",
     paste0(hp_1_col, startRow + i_parameter),
     ", ",
     paste0(hp_2_col, startRow + i_parameter),
     ")")
```

Output

```
"NORMINV(RAND(), I6, J6)"
```

With the input parameters specified, the next step is to construct the Markov trace. As has been noted in the literature, Markov models in Excel are implemented quite differently to Markov models in R.[30] R models use the matrix multiplication formulation of Figure 1 while Excel models write out the matrix multiplication in components. A simplified example of this is provided in Table 3.

Our proof of concept generates the state occupancy probability formulae (e.g., "=F9 * (1 - input_parameters!M9) + G9 * input_parameters!M11") in the Markov trace by using the df_spec data frame of Table 2.

This is made possible by associating each parameter with its source ('from') and destination ('to') states, as well as the treatment to which it corresponds. The R code is set up to ignore parameters without 'from' or 'to' settings as it would then recognise that these are costs or utilities, or perhaps adverse event rates. The code to accomplish this is provided in Code block 3.

*Table 3 First and second rows of Markov trace in standard Excel implementation of a Markov model**

E	F	G	H	I	J
8	cycle	SoC_Smoking	SoC_Not smoking	SoC with website_Smoking	SoC with website_Not smoking
9	1	1	0	1	0
10	2	=F9 * (1 - input_parameters!M9) + G9 * input_parameters!M11	=G9 * (1 - input_parameters!M11) + F9 * input_parameters!M9	=H9 * (1 - input_parameters!M10) + I9 * input_parameters!M11	=I9 * (1 - input_parameters!M11) + H9 * input_parameters!M10

*The input_parameters elements: M9=Smoking to not smoking on SoC+Website; M10=Smoking to not smoking on SoC; M11=Not smoking to smoking regardless of treatment

Code block 3 R code to automatically generate the Excel Markov trace formulae

```
# Generate the Markov trace using transition probabilities from markov_inputs
  df_markov_trace <- data.frame(cycle = c(1:n_cycles))

  cell_formula_temp <- rep("", n_cycles)

  for(i_treatment in 1:n_treatments) {
    for(i_state in 1:n_states) {
```

```
    cell_formula_temp[1] <- v_init_cohort[i_state]

    # Which parameters give probabilities from this state and are relevant
    # to this treatment (or to all treatments)
    from_indices <- which(markov_inputs$df_spec$from == i_state &
                  (markov_inputs$df_spec$v_treatment == i_treatment |
                   is.na(markov_inputs$df_spec$v_treatment)))
    # Create the sum of probabilities of exiting current state
    sum_probabilities_from <- ifelse(length(from_indices) == 1,
                        markov_inputs$df_spec$excel_value_location[from_indices],
                        paste0(markov_inputs$df_spec$excel_value_location[from_indices], collapse = "+"))

     for(i_cycle in 2:n_cycles) {
     # Numeric for row with previous cohort probabilities
     previous_row <- startRow + i_cycle - 1

     # Cell with probability of being in state at previous cycle
  cell_formula_temp[i_cycle] <- paste0(openxlsx2::int2col(startCol +
                                   (i_treatment - 1) * n_states +
                                   i_state), previous_row,
                              " * (1 - ",
                              if (sum_probabilities_from == "") 0 else sum_probabilities_from, ")")
    # Now append the probabilities of entering the state
     from_prob_formulae <- c()
     for(j_state in c(1:n_states)[-i_state]) {
      # Find the parameter storing transition probabilities from j to i
      from_j_to_i_index <- markov_inputs$df_spec$from == j_state &
       markov_inputs$df_spec$to == i_state &
       (markov_inputs$df_spec$v_treatment == i_treatment |
         is.na(markov_inputs$df_spec$v_treatment))
      # Check that there is a transition from j to i
      if(sum(from_j_to_i_index, na.rm = TRUE) == 1) {
       # Add probability of being in j multiplied by probability of going to i
       from_prob_formulae <- c(from_prob_formulae,
                    paste0(openxlsx2::int2col(startCol +
                              (i_treatment - 1) * n_states +
                              j_state), previous_row,
                         " * ",
                         markov_inputs$df_spec$excel_value_location[which(from_j_to_i_index)]))

      } # End if there are transitions from j to i
     } # End loop over j_state
     # Create the sum of probabilities of entering current state
     # Account for possibility that none of cohort makes this transition
    sum_probabilities_to <- ifelse(length(from_prob_formulae) == 0, "",
                        ifelse(length(from_prob_formulae) == 1,
                           from_prob_formulae,
                           paste0(from_prob_formulae, collapse = "+")))
     cell_formula_temp[i_cycle] <- paste0(c(cell_formula_temp[i_cycle], if (sum_probabilities_to == "") NULL else
sum_probabilities_to), collapse = " + ")
   } # End loop over i_cycle
    # Append these formulae to the Markov trace
    class(cell_formula_temp) <- c(class(cell_formula_temp), "formula")
    df_markov_trace <- cbind(df_markov_trace, cell_formula_temp)
    # Give column a sensible header
    names(df_markov_trace)[(i_treatment - 1) * n_states + i_state + 1] <- paste0(self$v_treatment_names[i_treatment],
                                                "_",
                                                self$v_state_names[i_state])
   } # End loop over states
  } # End loop over treatments
```

A similar approach to writing out Excel formulae is to used to generate treatment costs, state costs and state QALYs, and to add total costs and QALYs, with and without discounting, to the Excel workbook.

The Excel logic is written to a workbook “test_output_1_1.xlsm” which includes a Visual Basic for Applications (VBA) macro to run the PSA. This simply calculates the whole sheet n_samples times and saves the output to the PSA sheet. The same macro can be used regardless of the model implemented in R, so does not need to be modified.

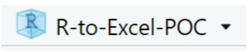
The final output of the Excel generation is compared to the R estimation for 1000 PSA samples in Table 4. The PSA macro is built into a dummy Excel sheet (named test_output_1_1.xlsm in the proof of concept) which simply loops through the Excel model. We see that the results are very close, with only some differences due to different random number generators.

Table 4 Comparison of results generated by R and Excel

	R		**Automatically generated Excel**	
	SoC	**SoC with website**	**SoC**	**SoC with website**
Total costs	312.453 (248.579, 374.685)	337.756 (276.242, 400.320)	307.72	331.60
Total QALYs	4.476 (4.346, 4.606)	4.488 (4.373, 4.609)	4.40	4.41
Total costs undiscounted	333.344 (263.673, 401.680)	356.240 (289.118, 424.459)	333.99	354.83
Total QALYs undiscounted	4.832 (4.692, 4.971)	4.846 (4.722, 4.975)	4.83	4.85
ICER	NaN	2029.36		1823.06

Instructions to test R-to-Excel-POC

To test the code the following steps should be followed:

- From within RStudio click on “File” and then "Open Project..." and then navigate to “R-to-Excel-POC.Rproj”
- Ensure you see the text "R-to-Excel-POC" in the top right corner of RStudio:
  
  R-to-Excel-POC
- In the ‘analysis’ subfolder open the “smoking_main.R” script
- Run all. This loads the Markov smoking model into the template, generates an Excel equivalent in “output/test_output_1_1.xlsm”, and generates a results matrix with total costs, QALYs and other summaries.
- Load the Excel sheet “output/test_output_1_1.xlsm” in Excel and navigate to the “PSA” tab.
- The exported input parameters are stored in the “input_parameters” tab and the reconstructed Markov trace in “markov_trace”
- Click the “Run PSA” button and look at the totals for each treatment that are saved in cells J3:U3.
- Comparing these verifies that the R and Excel models are in agreement.
- The settings in the “input_parameters” can be changed freely, as if the model had been built in Excel in the first place.

- You can change the settings (e.g., number of treatments, states, transition probability distributions) in smoking_main.R (note: we recommend saving as a new script so that original is preserved) allows other Markov models to be designed and exported.
- To export a new model create a copy of the "model_output_template.xlsm" Excel output template with your desired model name.
- Refer to help files using help() or ? to see documentation of individual member functions of the input_parameters and markov_model classes.

The input_parameters$specify_excel() function currently only has implementations for parameters that are fixed, follow beta distributions, or follow Normal distributions. Gamma, lognormal, uniform and other common distributions would need to be added if needed for applications.

Scientific references

1. Incerti D, Thom H, Baio G, et al. R You Still Using Excel? The Advantages of Modern Software Tools for Health Technology Assessment. *Value Health* 2019;22(5):575-79. doi: 10.1016/j.jval.2019.01.003
2. Jalal H, Pechlivanoglou P, Krijkamp E, et al. An Overview of R in Health Decision Sciences. *Med Decis Making* 2017;37(7):735-46. doi: 10.1177/0272989X16686559
3. R Core Team. R: A Language and Environment for Statistical Computing. https://www.R-project.org/. Vienna, Austria: R Foundation for Statistical Computing 2019.
4. 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(7):765-76. doi: 10.1007/s40273-020-00903-9
5. Chang W, Cheng J, Allaire J, et al. shiny:Web application Framework for R. R package version 0.13.2. 2016
6. Krijkamp EM, Alarid-Escudero F, Enns EA, et al. A Multidimensional Array Representation of State-Transition Model Dynamics. *Med Decis Making* 2020;40(2):242-48. doi: 10.1177/0272989X19893973
7. 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* 2017;37(4):340-52. doi: 10.1177/0272989X16651869
8. Incerti D, Jansen JP. Health-Economic Simulation Modeling and Decision Analysis: https://cran.r-project.org/web/packages/hesim/hesim.pdf. *CRAN* 2019
9. Baio G, Berardi A, Heath A. Bayesian Cost-Effectiveness Analysis with the R package BCEA: Springer International Publishing, 2017.
10. Alarid-Escudero F, Krijkamp EM, Pechlivanoglou P, et al. A Need for Change! A Coding Framework for Improving Transparency in Decision Modeling. *Pharmacoeconomics* 2019;37(11):1329-39. doi: 10.1007/s40273-019-00837-x
11. Wickham H. The tidyverse style guide https://style.tidyverse.org/ (Accessed 15-June-2022). 2022
12. Zorginstituutnederland. Guideline for building cost-effectiveness models in R https://www.zorginstituutnederland.nl/publicaties/publicatie/2022/10/04/richtlijn-kosteneffectiviteitsmodellen-in-r (Accessed 8-Dec-2022). 2022

13. Siebert U, Alagoz O, Bayoumi AM, et al. State-Transition Modelling: A report of the ISPOR-SMDM Modelling Good Research Practices Task Force-3. *Value in Health* 2012;15:812-20.
14. Briggs AH, Claxton K, Sculpher MJ. Decision modelling for health economic evaluation. Oxford: Oxford University Press 2006.
15. Thom HH, Jackson CH, Commenges D, et al. State selection in Markov models for panel data with application to psoriatic arthritis. *Stat Med* 2015;34(16):2456-75. doi: 10.1002/sim.6460
16. Baharnoori M, Bhan V, Clift F, et al. Cost-Effectiveness Analysis of Ofatumumab for the Treatment of Relapsing-Remitting Multiple Sclerosis in Canada. *Pharmacoecon Open* 2022;6(6):859-70. doi: 10.1007/s41669-022-00363-1
17. Handels RLH, Green C, Gustavsson A, et al. Cost-effectiveness models for Alzheimer's disease and related dementias: IPECAD modeling workshop cross-comparison challenge. *Alzheimers Dement* 2023;19(5):1800-20. doi: 10.1002/alz.12811
18. Beyersmann J, Allignol A, Schumacher M. Competing Risks and Multistate Models with R: Springer 2012.
19. Jackson C. Multi-State Models for Panel Data: The msm Package for R. *Journal of Statistical Software* 2011;38(8) doi: 10.18637/jss.v038.i08
20. Thom H. Deterministic and Probabilistic Analysis of a Simple Markov Model: How Different Could They Be? *Appl Health Econ Health Policy* 2022 doi: 10.1007/s40258-021-00700-1
21. Wilson ECF. Methodological Note: Reporting Deterministic versus Probabilistic Results of Markov, Partitioned Survival and Other Non-Linear Models. *Appl Health Econ Health Policy* 2021;19(6):789-95. doi: 10.1007/s40258-021-00664-2
22. Zarca K, Filipovic-Pierucci A, Wiener M, et al. Markov Models for Health Economic Evaluation: The R Package heemod. *CRAN* 2017
23. Elwenspoek MM, Thom H, Sheppard AL, et al. Defining the optimum strategy for identifying adults and children with coeliac disease: systematic review and economic modelling. *Health Technol Assess* 2022;26(44):1-310. doi: 10.3310/ZUCE8371
24. Thom HHZ, Hollingworth W, Sofat R, et al. Directly Acting Oral Anticoagulants for the Prevention of Stroke in Atrial Fibrillation in England and Wales: Cost-Effectiveness Model and Value of Information Analysis. *MDM Policy Pract* 2019;4(2):2381468319866828. doi: 10.1177/2381468319866828
25. Thom H, Visan AC, Keeney E, et al. Clinical and cost-effectiveness of the Ross procedure versus conventional aortic valve replacement in young adults. *Open Heart* 2019 doi: http://dx.doi.org/10.1136/openhrt-2019-001047
26. Sterne JA, Bodalia PN, Bryden PA, et al. Oral anticoagulants for primary prevention, treatment and secondary prevention of venous thromboembolic disease, and for prevention of stroke in atrial fibrillation: systematic review, network meta-analysis and cost-effectiveness analysis. *Health Technol Assess* 2017;21(9):1-386. doi: 10.3310/hta21090
27. NICE. Atrial fibrillation: diagnosis and management. The National Institute for Health and Care Excellence Guideline [NG196]. Available from: https://www.nice.org.uk/guidance/ng196 Site accessed 18/10/22. 2021
28. Alarid-Escudero F, Krijkamp E, Enns EA, et al. An Introductory Tutorial on Cohort State-Transition Models in R Using a Cost-Effectiveness Analysis Example. *Med Decis Making* 2023;43(1):3-20. doi: 10.1177/0272989X221103163
29. Alarid-Escudero F, Krijkamp E, Enns EA, et al. A Tutorial on Time-Dependent Cohort State-Transition Models in R Using a Cost-Effectiveness Analysis Example. *Med Decis Making* 2023;43(1):21-41. doi: 10.1177/0272989x221121747 [published Online First: 2022/09/17]
30. Green N, Lamrock F, Naylor N, et al. Health Economic Evaluation Using Markov Models in R for Microsoft Excel Users: A Tutorial. *Pharmacoeconomics* 2023;41(1):5-19. doi: 10.1007/s40273-022-01199-7 [published Online First: 2022/11/07]
31. Smith R, Mohammed W, Schneider P. Packaging cost-effectiveness models in R: A tutorial. *Wellcome Open Res* 2023;8:419. doi: https://doi.org/10.12688/wellcomeopenres.19656.1