Proof-of-concept for automatic export of Excel versions of R health economic models

Howard Thom[1,2], Michael O'Donnell[1], Queena Wang[2], Edna Keeney[2]

1 University of Bristol, UK

2 Clifton Insight, Bristol, UK

Funder acknowledgement

- This work was partly funded by Clifton Insight and the Reliable and Efficient Estimation of the Economic Value of medical Research (REEEVR) project at University of Bristol, itself funded by UK Medical Research Council grant MR/W029855/1.

- All code for today’s presentation is available at:

https://github.com/Bogdasayen/R-to-Excel-POC

Why R-to-Excel conversion?

- **UK NICE, Irish NCPE, Dutch Zin and other forward-thinking agencies accept R models, but others do not.**
- Companies may want to leverage efficiency, flexibility, and transparency advantages of R but will still need an Excel version.
- Double programming models can lead to errors and make it difficult to push updates across international models.
- **Instead use R models that can export Excel versions of themselves.**

https://github.com/Bogdasayen/R-to-Excel-POC

Requirements of R-to-Excel conversion

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

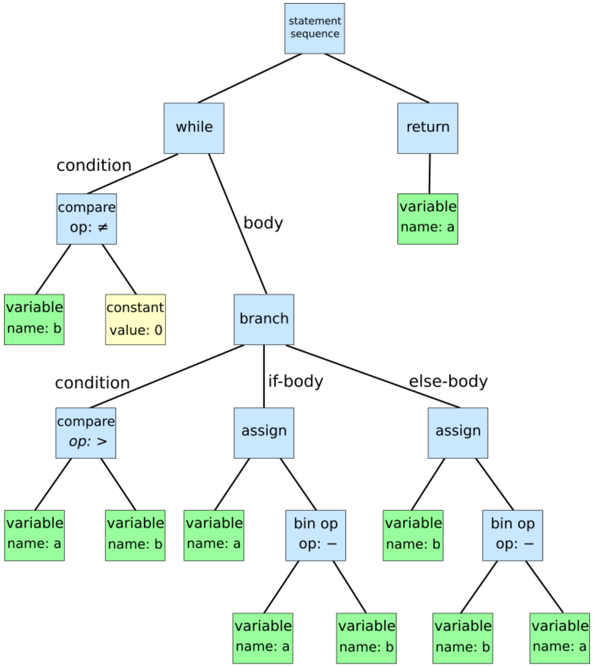
https://github.com/Bogdasayen/R-to-Excel-POC

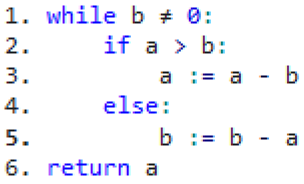
Reverse REEEVR?

- At University of Bristol, we are developing the REEEVR tool to automatically convert Excel models into R.
- This gains benefits of speed (100x improvement) at loss of readability.
- Approach theoretically works on any Excel model (though needs an R equivalent of each Excel function to be implemented)
- **Why not do this in reverse to convert R into Excel?**

https://github.com/Bogdasayen/R-to-Excel-POC

Abstract Syntax Tree - General





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

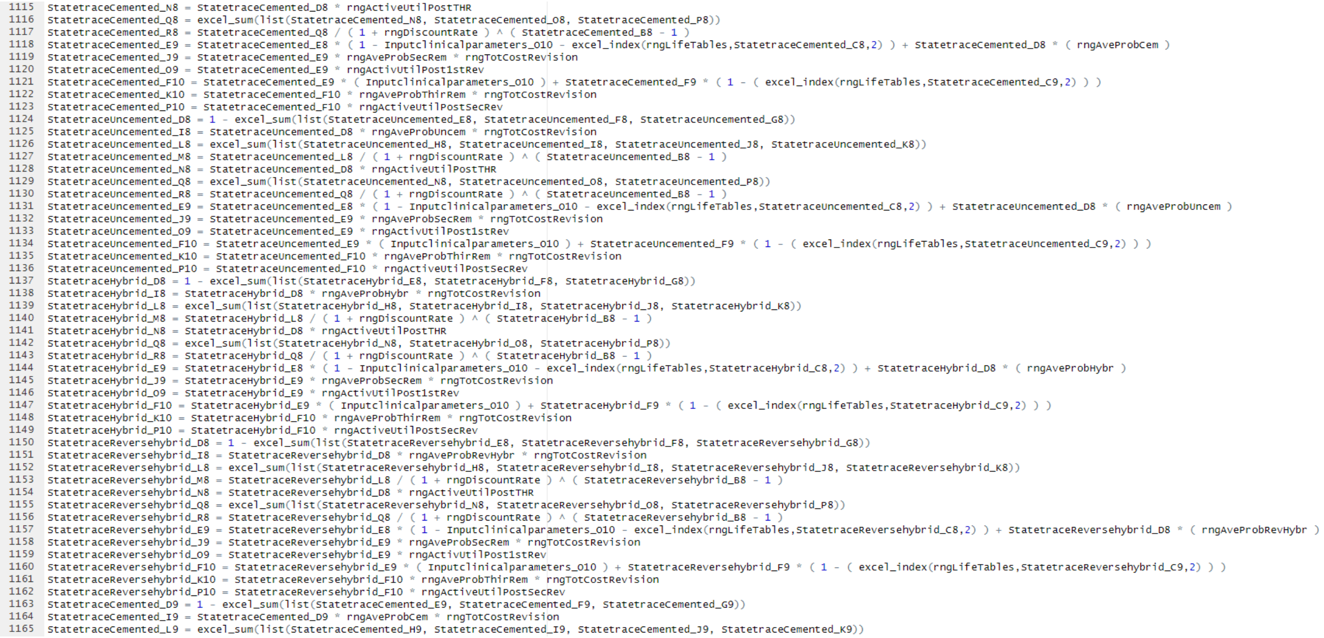
- Walks the program
- Breaks code into components
- Builds a tree where each leaf is an elementary component of the program
- The tree is language agnostic
- **Could be applied to R code**

bristol.ac.uk

https://github.com/Bogdasayen/R-to-Excel-POC

Clifton Insight

Unreadable and slow!



```
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))
```

- A simple 4-state Markov model with 4 interventions results in 2614 lines of **unreadable** R code.
- Applying in reverse would create either very many **unreadable** cell calculations, or a small number of very long **unreadable** cell calculations.
- Result would also likely be **very slow** as loops and vector operations (e.g., for PSA) would be duplicated Excel cells, rather than VBA macros.

bristol.ac.uk

https://github.com/Bogdasayen/R-to-Excel-POC

Clifton Insight

Using R to write Excel formulae

- We instead use the powerful **openxlsx R package** (maintained by Jan Marvin Garbuszus)
- This allows us to write data, formulae and formatting from R to Excel
  - Example usage below

```
output_wb <- loadWorkbook(file = wb_filename, isUnzipped = FALSE)
addWorksheet(output_wb, "Results", tabColour = "orange")
df_results <- Some formulae see next slide
class(df_results[, 1] <- c(class(df_results[, 1]), "formula")
names(df_results) <- c("Result", self$v_treatment_names)
writeData(output_wb, sheet = "Results",  x = df_results,
          startCol = 8, startRow = 5)
saveWorkbook(output_wb, file = wb_filename,
                    overwrite = TRUE, returnValue = TRUE)
```

bristol.ac.uk

https://github.com/Bogdasayen/R-to-Excel-POC

Clifton Insight

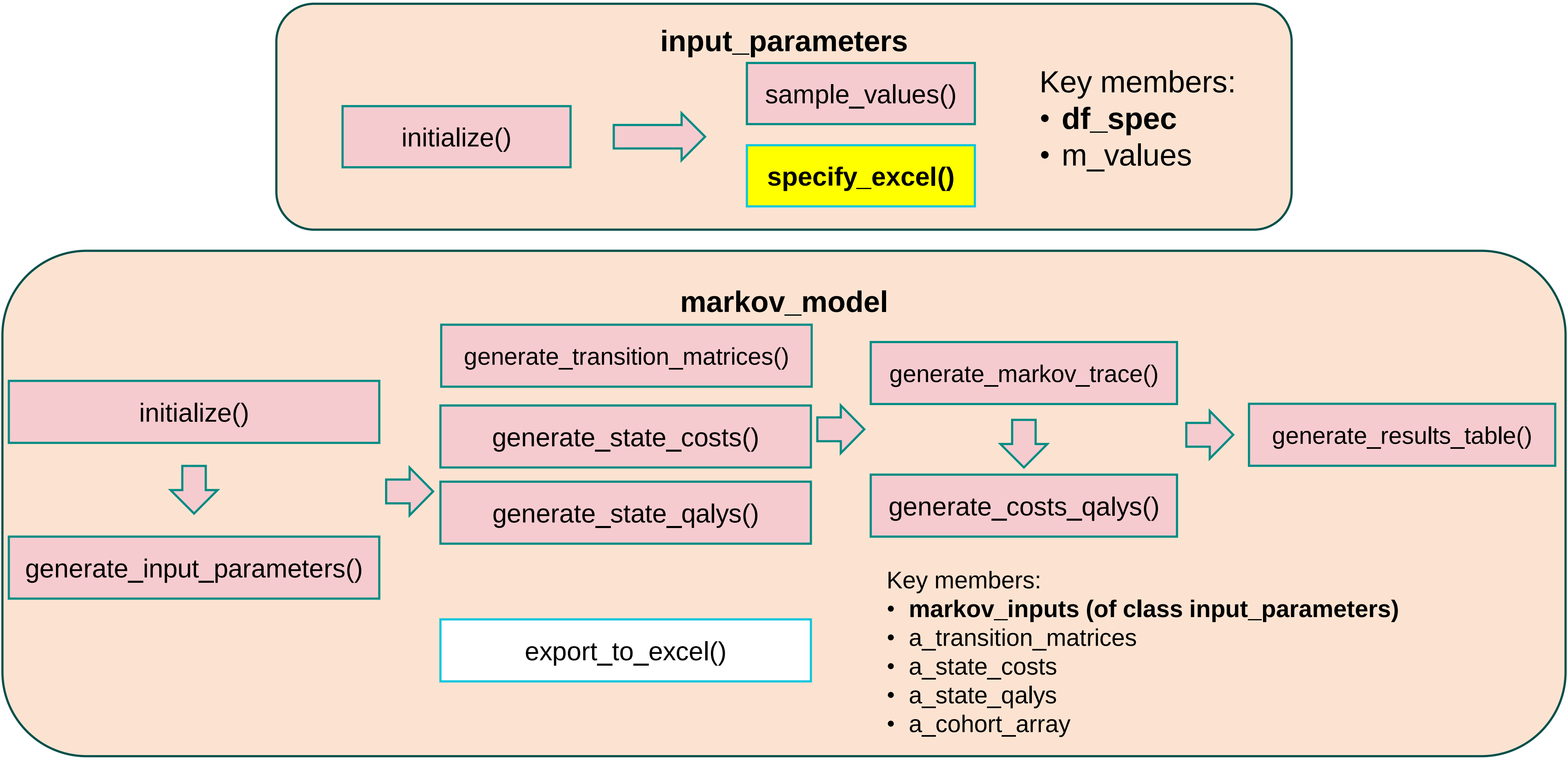
Object Oriented Programming

- Process kept manageable through R6 classes for OOP
- Each model is an object with member functions and data
- The model object maps relationships between input parameters, transition matrices, costs and QALYs

- **This mapping of data relationships is helpful for R-to-Excel conversion**

https://github.com/Bogdasayen/R-to-Excel-POC

Two key R6 objects with two key functions



input_parameters

initialize()

sample_values()

specify_excel()

Key members:
- **df_spec**
- m_values

markov_model

initialize()

generate_input_parameters()

generate_transition_matrices()

generate_state_costs()

generate_state_qalys()

export_to_excel()

generate_markov_trace()

generate_costs_qalys()

generate_results_table()

Key members:
- **markov_inputs (of class input_parameters)**
- a_transition_matrices
- a_state_costs
- a_state_qalys
- a_cohort_array

- **We’ll start with input_parameters$specify_excel()**

bristol.ac.uk

https://github.com/Bogdasayen/R-to-Excel-POC

Clifton Insight

Data structure and description

- Data structure and complete description of relationships is key.
- The dataframe **df_spec** fully specifies distributions used in model
- The contents of this dataframe are written by **specify_excel()**
- Column excel_formulae is generated automatically based on inputs

v_names	v_descriptions	v_distributions	hp_1	hp_2	from	to	v_treatment	excel_formulae	excel_value_location
Probability quit smoking website	Probability of quitting if on smoking website, follows a beta distribution	beta	15	85	1	2	1	**=BETAINV(RAND(), H9, I9)**	input_parameters!M9

https://github.com/Bogdasayen/R-to-Excel-POC

Example Excel random formula generation

- Cell formulae generated using paste0 statements and saved cell locations

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 <- LETTERS[startCol + 1]
	hp_2_col <- LETTERS[startCol + 2]
	
	paste0("NORMINV(RAND(), ",
	        paste0(hp_1_col, startRow + i_parameter),
	        ", ",
	        paste0(hp_2_col, startRow + i_parameter),
	        ")")

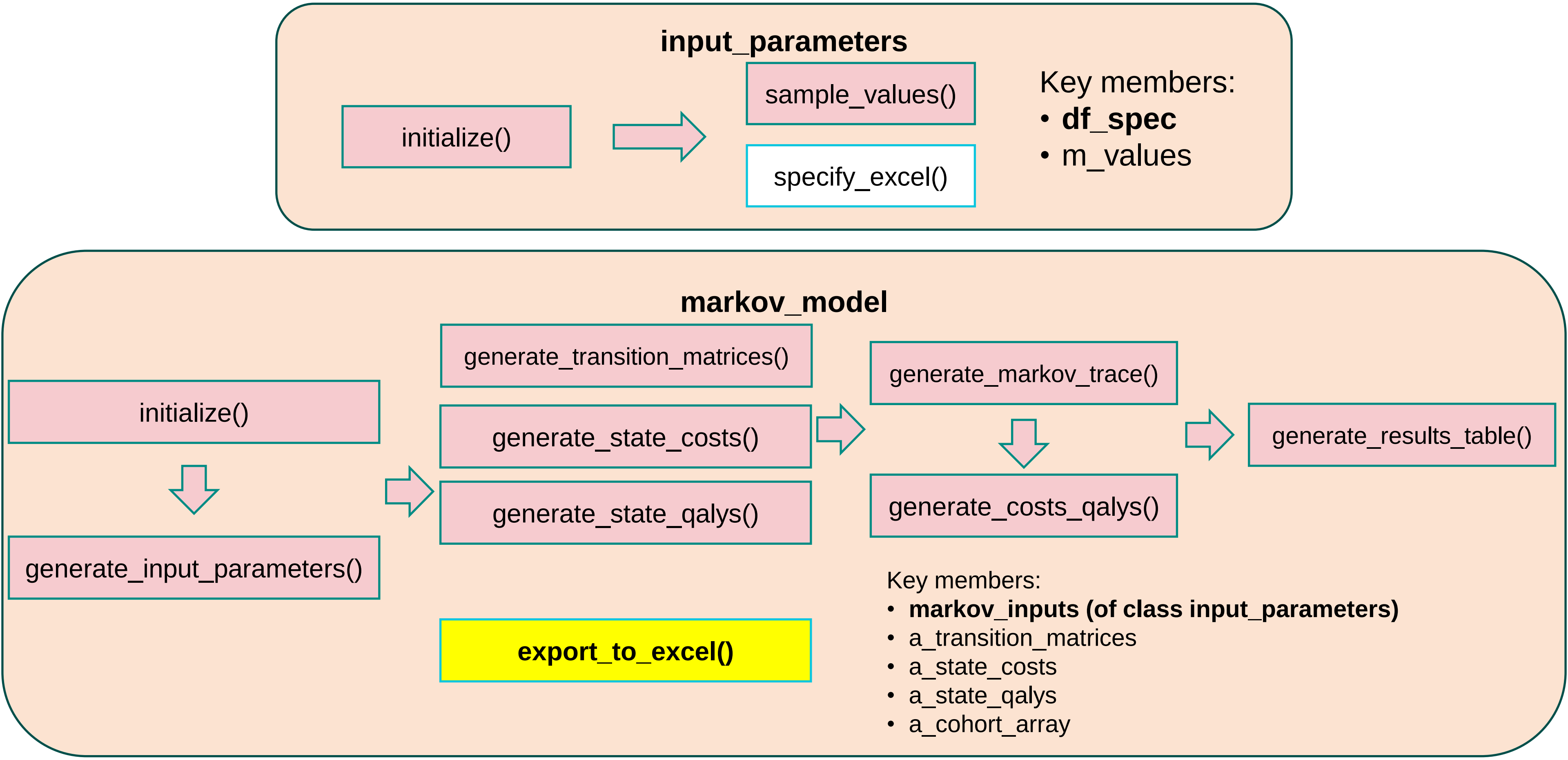
Output	"NORMINV(RAND(), I6, J6)"

bristol.ac.uk

https://github.com/Bogdasayen/R-to-Excel-POC

Clifton Insight

Two key R6 objects and two key functions



input_parameters

initialize()

sample_values()

specify_excel()

Key members:
- **df_spec**
- m_values

markov_model

initialize()

generate_input_parameters()

generate_transition_matrices()

generate_state_costs()

generate_state_qalys()

generate_markov_trace()

generate_costs_qalys()

generate_results_table()

export_to_excel()

Key members:
- **markov_inputs (of class input_parameters)**
- a_transition_matrices
- a_state_costs
- a_state_qalys
- a_cohort_array

- **Next is markov_model$export_to_excel()**

bristol.ac.uk

https://github.com/Bogdasayen/R-to-Excel-POC

Clifton Insight

Markov trace generation

- Generates state occupancy probabilities using df_spec
- Associates each parameter with its source (‘from’) and destination (‘to’)
- Ignores parameters without ‘from’ or ‘to’ as these are costs or utilities
- First two rows of generated Markov trace (With Excel locations) are provided below.

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

https://github.com/Bogdasayen/R-to-Excel-POC

Markov trace generation – part 1

```
# Generate the Markov trace using transition probabilities from markov_inputs
  df_markov_trace <- data.frame(cycle = c(1:self$n_cycles))
  cell_formula_temp <- rep("", n_cycles)
  for(i_treatment in 1:self$n_treatments) {
   for(i_state in 1:self$n_states) {
    cell_formula_temp[1] <- self$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(self$markov_inputs$df_spec$from == i_state &
                (self$markov_inputs$df_spec$v_treatment == i_treatment |
                 is.na(self$markov_inputs$df_spec$v_treatment)))

    # Create the sum of probabilities of exiting current state
    sum_probabilities_from <- ifelse(length(from_indices) == 1,
                     self$markov_inputs$df_spec$excel_value_location[from_indices],
paste0(self$markov_inputs$df_spec$excel_value_location[from_indices], sep = "+"))
```

Markov trace generation – part 2

```
for(i_cycle in 2:self$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(LETTERS[startCol +
                                         (i_treatment - 1) * n_states +
                                         i_state], previous_row,
                                     " * (1 - ", sum_probabilities_from,")")

    # Now append the probabilities of entering the state
    from_prob_formulae <- c()
    for(j_state in c(1:self$n_states)[-i_state]) {
     # Find the parameter storing transition probabilities from j to i
     from_j_to_i_index <- self$markov_inputs$df_spec$from == j_state &
      self$markov_inputs$df_spec$to == i_state &
      (self$markov_inputs$df_spec$v_treatment == i_treatment |
        is.na(self$markov_inputs$df_spec$v_treatment))
```

https://github.com/Bogdasayen/R-to-Excel-POC

Markov trace generation – part 3

```
# 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(LETTERS[startCol +
                        (i_treatment - 1) * n_states +
                        j_state], previous_row,
                    " * ",
self$markov_inputs$df_spec$excel_value_location[which(from_j_to_i_inde
x)]))

      } # End if there are transitions from j to i
     } # End loop over j_state
```

https://github.com/Bogdasayen/R-to-Excel-POC

Markov trace generation – part 4

```
 # 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, sep = "+")))
        cell_formula_temp[i_cycle] <- paste0(cell_formula_temp[i_cycle], " + ",
sum_probabilities_to)

      } # 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)
```

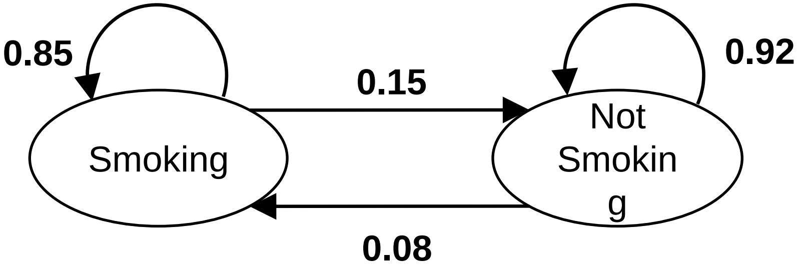
Costs, QALYs, PSA

- Similar approach is taken to costs and QALYs
- Output model is probabilistic but only for 1 sample
- Pre-built VBA macro used to run probabilistic sensitivity analysis
- So long as total cost and QALY cells are written in a specific format the VBA macro will work
  - e.g., starting from row 8 and column 5 in markov_trace sheet

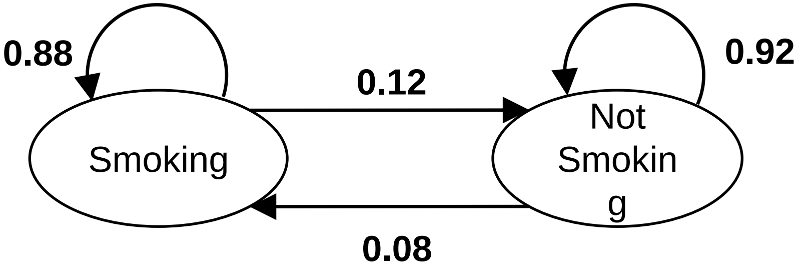
Now let's test it!

https://github.com/Bogdasayen/R-to-Excel-POC

Markov smoking



SoC + Website



SoC

- Teaching model used for Bristol University short courses and MSc teaching
- Two states and two treatments
- SoC + Website increases chance of quitting smoking but costs more money

bristol.ac.uk

https://github.com/Bogdasayen/R-to-Excel-POC

Clifton Insight

df_spec for utilities

v_names	v_descriptions	v_type	v_distributions	hp_1	hp_2	v_treatment	v_state	excel_value_location
Utility smoking	Utility smoking, follows a Normal distribution	utility	normal	0.95	0.02		1	input_parameters!O12
Utility not smoking	Utility not smoking, follows a Normal distribution	utility	fixed	1			2	input_parameters!O13

- Treatment column left blank, so these utilities apply to both treatments

https://github.com/Bogdasayen/R-to-Excel-POC

df_spec for costs

v_names	v_descriptions	v_type	v_distributions	hp_1	hp_2	v_treatment	v_state	excel_value_location
Cost website	Cost website, fixed value and model assumes no cost of SoC and no state costs	one_off_cost	fixed	50		2		input_parameters!O14
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	input_parameters!O15
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	input_parameters!O16

- Costs can be one-off or ongoing.

df_spec for transition probabilities

v_names	v_descriptions	v_type	v_distributions	hp_1	hp_2	from	to	v_treatment	v_state	excel_value_location
Probability quit smoking website	Probability of quitting if on smoking website, follows a beta distribution	transition_probability	beta	15	85	1	2	2	1	input_parameters!O9
Probability quit smoking SoC	Probability of quitting smoking if on SoC, follows a beta distribution	transition_probability	beta	12	88	1	2	1	1	input_parameters!O10
Probability relapse	Probability relapse, which is same across treatments and follows a beta distribution	transition_probability	beta	8	92	2	1		2	input_parameters!O11

- Probability of relapse is the same for both treatments

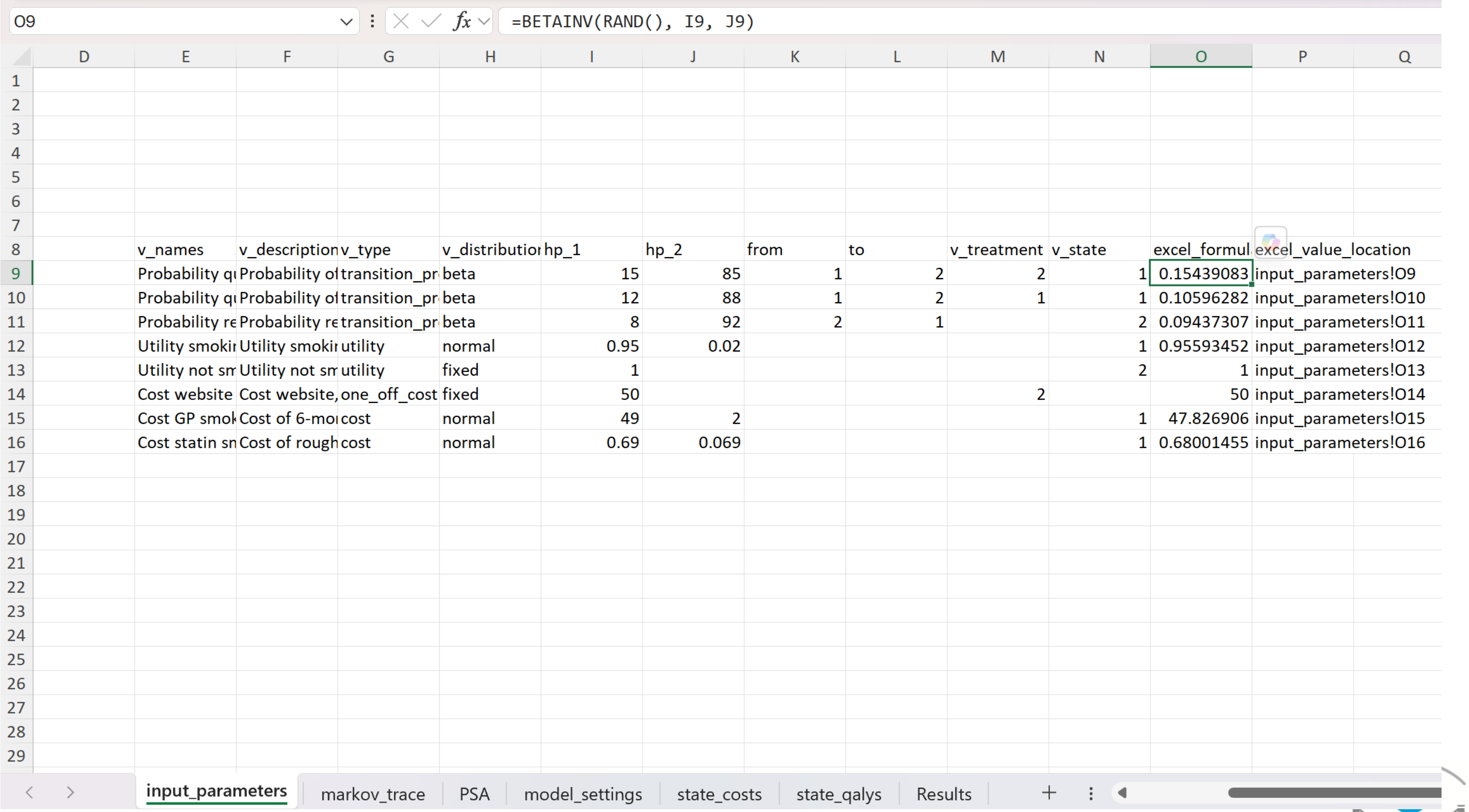
If v_treatment is missing it's the same for all treatments. The states used in from and to columns are 1=Smoking and 2=Not smoking; the treatments are 1=SoC and 2=SoC+Website. Column O is omitted and stores the live value used in the model.

bristol.ac.uk

Clifton Insight

https://github.com/Bogdasayen/R-to-Excel-POC

Output – note formula bar



O9 =BETAINV(RAND(), I9, J9)

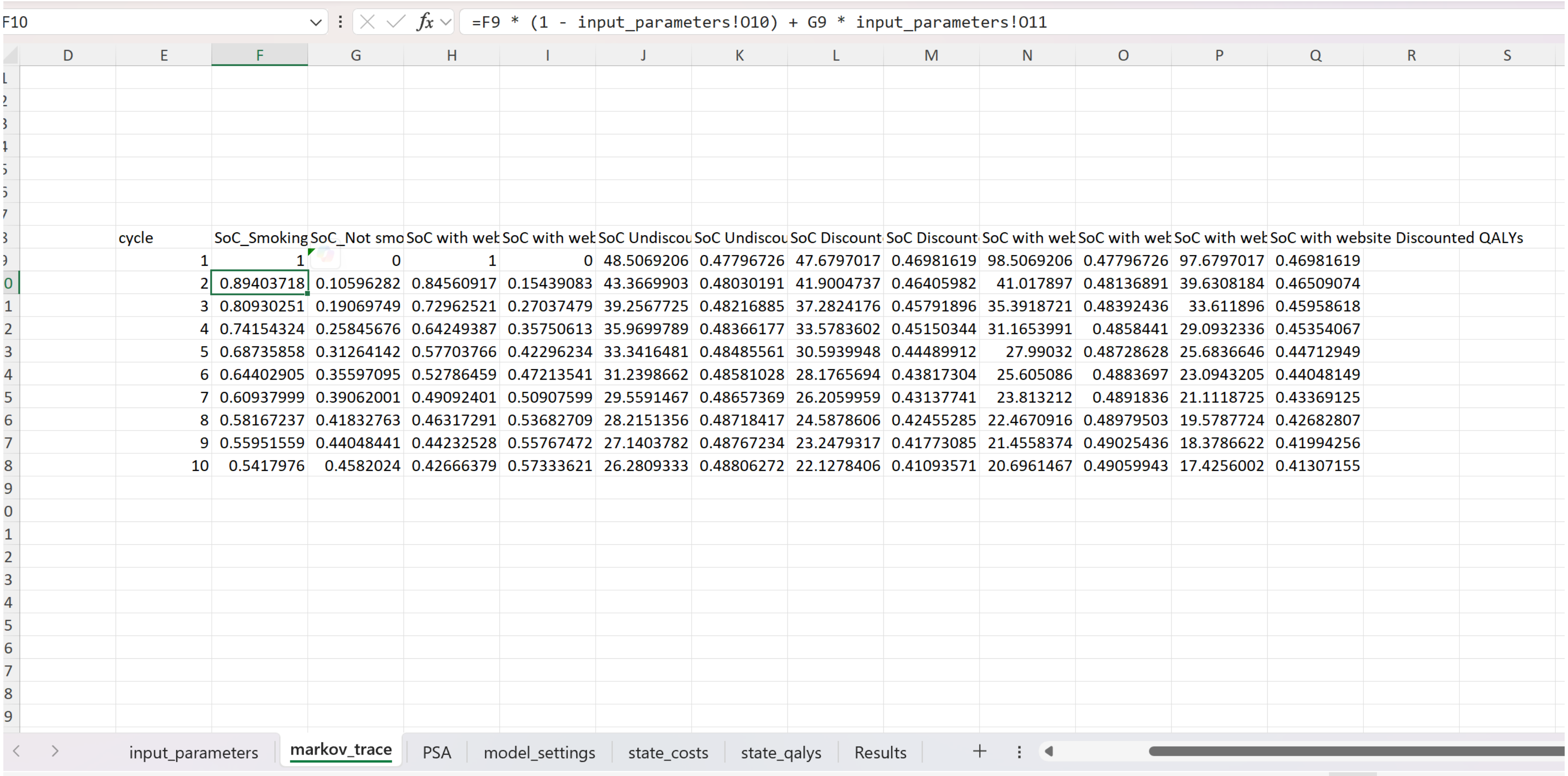
	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1														
2														
3														
4														
5														
6														
7														
8		v_names	v_description	v_type	v_distributior	hp_1	hp_2	from	to	v_treatment	v_state	excel_formul	excel_value_location	
9		Probability q	Probability o	transition_pr	beta	15	85	1	2	2	1	0.15439083	input_parameters!O9	
10		Probability q	Probability o	transition_pr	beta	12	88	1	2	1	1	0.10596282	input_parameters!O10	
11		Probability re	Probability re	transition_pr	beta	8	92	2	1		2	0.09437307	input_parameters!O11	
12		Utility smokir	Utility smokir	utility	normal	0.95	0.02				1	0.95593452	input_parameters!O12	
13		Utility not sm	Utility not sm	utility	fixed	1					2	1	input_parameters!O13	
14		Cost website	Cost website,	one_off_cost	fixed	50				2		50	input_parameters!O14	
15		Cost GP smok	Cost of 6-mo	cost	normal	49	2				1	47.826906	input_parameters!O15	
16		Cost statin sn	Cost of rough	cost	normal	0.69	0.069				1	0.68001455	input_parameters!O16	
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														

bristol.ac.uk

https://github.com/Bogdasayen/R-to-Excel-POC

Clifton Insight

Output – note formula bar



```
=F9 * (1 - input_parameters!O10) + G9 * input_parameters!O11
```

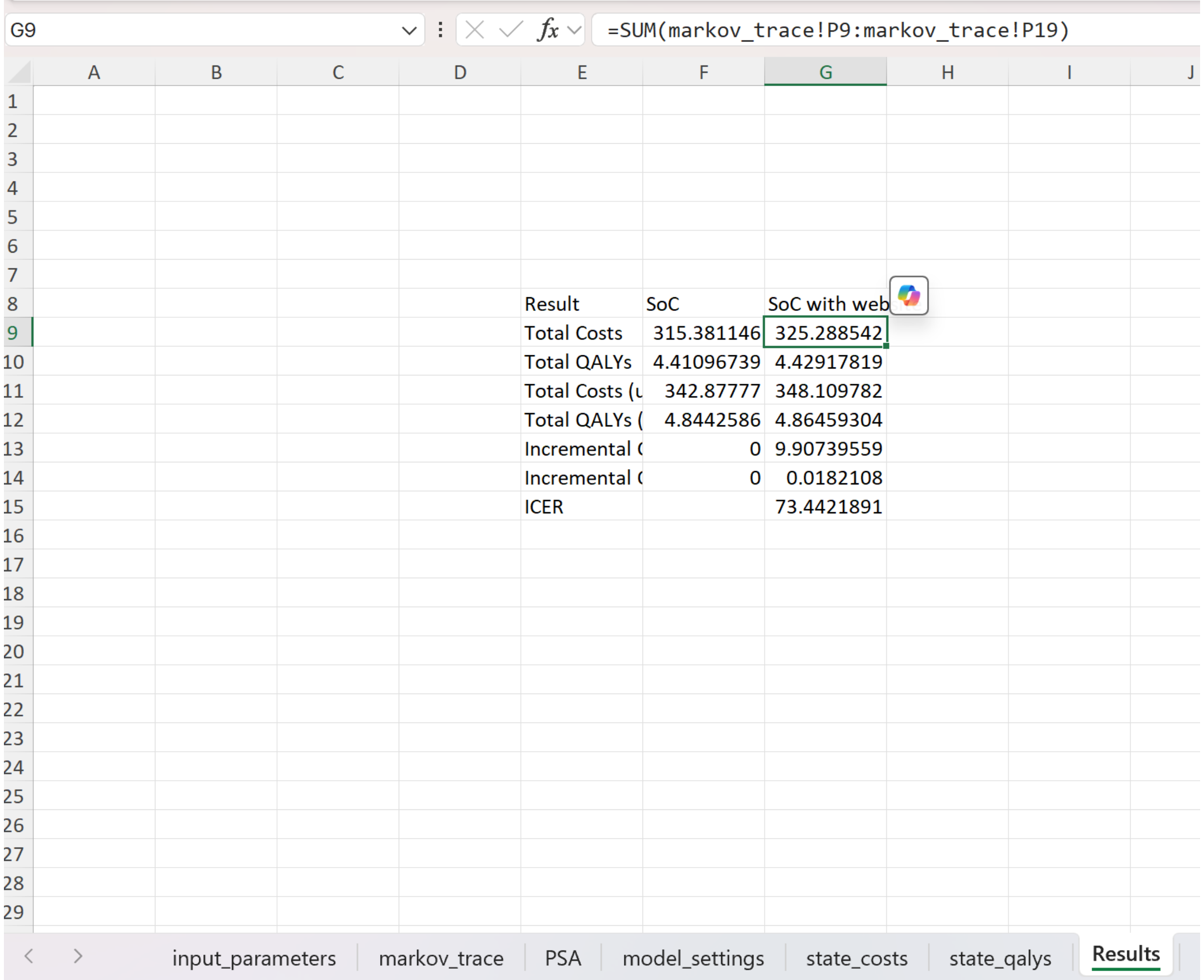
cycle	SoC_Smoking	SoC_Not smo	SoC with web	SoC with web	SoC Undiscou	SoC Undiscou	SoC Discount	SoC Discount	SoC with web	SoC with web	SoC with web	SoC with website Discounted QALYs
1	1	0	1	0	48.5069206	0.47796726	47.6797017	0.46981619	98.5069206	0.47796726	97.6797017	0.46981619
2	0.89403718	0.10596282	0.84560917	0.15439083	43.3669903	0.48030191	41.9004737	0.46405982	41.017897	0.48136891	39.6308184	0.46509074
3	0.80930251	0.19069749	0.72962521	0.27037479	39.2567725	0.48216885	37.2824176	0.45791896	35.3918721	0.48392436	33.611896	0.45958618
4	0.74154324	0.25845676	0.64249387	0.35750613	35.9699789	0.48366177	33.5783602	0.45150344	31.1653991	0.4858441	29.0932336	0.45354067
5	0.68735858	0.31264142	0.57703766	0.42296234	33.3416481	0.48485561	30.5939948	0.44489912	27.99032	0.48728628	25.6836646	0.44712949
6	0.64402905	0.35597095	0.52786459	0.47213541	31.2398662	0.48581028	28.1765694	0.43817304	25.605086	0.4883697	23.0943205	0.44048149
7	0.60937999	0.39062001	0.49092401	0.50907599	29.5591467	0.48657369	26.2059959	0.43137741	23.813212	0.4891836	21.1118725	0.43369125
8	0.58167237	0.41832763	0.46317291	0.53682709	28.2151356	0.48718417	24.5878606	0.42455285	22.4670916	0.48979503	19.5787724	0.42682807
9	0.55951559	0.44048441	0.44232528	0.55767472	27.1403782	0.48767234	23.2479317	0.41773085	21.4558374	0.49025436	18.3786622	0.41994256
10	0.5417976	0.4582024	0.42666379	0.57333621	26.2809333	0.48806272	22.1278406	0.41093571	20.6961467	0.49059943	17.4256002	0.41307155

bristol.ac.uk

https://github.com/Bogdasayen/R-to-Excel-POC

Clifton Insight

Output – note formula bar



G9

=SUM(markov_trace!P9:markov_trace!P19)

Result	SoC	SoC with web
Total Costs	315.381146	325.288542
Total QALYs	4.41096739	4.42917819
Total Costs (u	342.87777	348.109782
Total QALYs (	4.8442586	4.86459304
Incremental (	0	9.90739559
Incremental (	0	0.0182108
ICER		73.4421891

bristol.ac.uk

https://github.com/Bogdasayen/R-to-Excel-POC

Clifton Insight

Comparison of results

	R		Automatically generated Excel	
Treatment	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

- Results based on 1000 random samples
- Only numerical variation due to differences in random number generation

bristol.ac.uk

https://github.com/Bogdasayen/R-to-Excel-POC

Clifton Insight

Next steps

- The supplied proof-of-concept code works for any time-homogeneous Markov model
- Allow transition matrices to be time-inhomogeneous
- Implement formatting of the Excel format using the options of the openxlsx package
- Export IF and INDEX statements to allow switching between scenarios
- Validation of the conversion on multiple toy and real-world applications

https://github.com/Bogdasayen/R-to-Excel-POC

Thank you!

bristol.ac.uk

https://github.com/Bogdasayen/R-to-Excel-POC

Clifton Insight