REEEVR - Automated Conversion of Excel Cost-Effectiveness Models to R Models

Presenting Author: Michael J. O'Donnell

Authors:

Michael J. O'Donnell [1]

Qian Xin [1]

Gabriel Rogers [2]

Abdul-Lateef Haji-Ali [3]

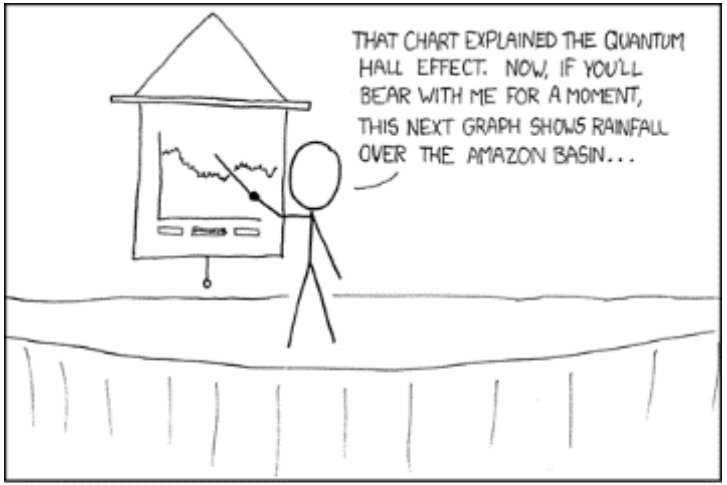
Howard Thom [1]

[1]*University of Bristol, UK;* [2]*University of Manchester, UK;* [3]*Heriot-Watt University, UK*

bristol.ac.uk

The presentation

- Previous work
- Why a conversion process
- The Technical Bit
- Examples
- Demonstration



IF YOU KEEP SAYING "BEAR WITH ME FOR A MOMENT", PEOPLE TAKE A WHILE TO FIGURE OUT THAT YOU'RE JUST SHOWING THEM RANDOM SLIDES.

https://xkcd.com/365/

michael.jon.odonnell@bristol.ac.uk

Previous improvements to HTA

Excel focused:

- SAVI
  - Generates CE/CEAC automatically
  - EVPI
  - EVPPI
  - Requires PSA already performed

R focused

- BCEA
  - Encourages use of Bayesian modelling
  - Generates CE/CEAC
  - EVPPI
- MLMC
  - R Package based on C++ package.
  - Utilises MLMC to sample the parameter space for VOI calculations

michael.jon.odonnell@bristol.ac.uk

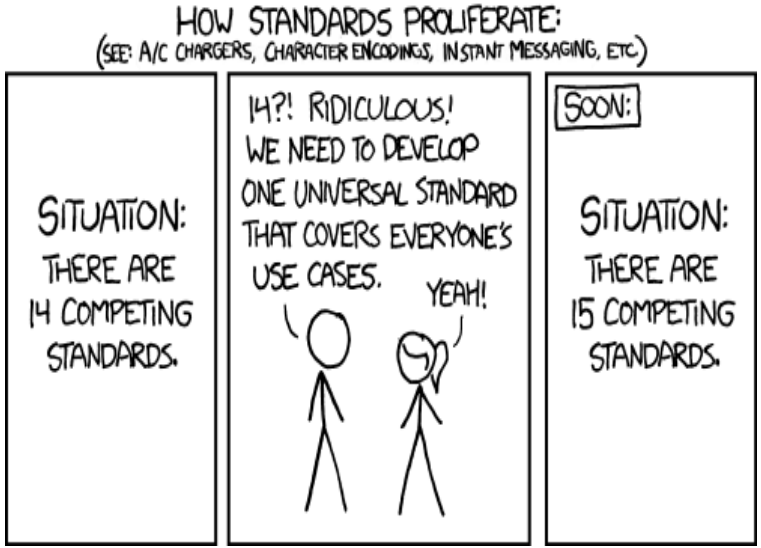
Advantages of pure R

- Can utilise various VOI methods
  - Types: EVPI, EVPPI
  - Premade R libraries: MLMC, BCEA
- Can take advantage of high performance computing techniques
  - In hardware: multithreading, SIMD, multi-nodal (MPI)
  - In Software: vectorisation, pre-compilation
- Reduces effect of HTA PSA runtime restrictions (e.g. CADTH)
- It's free

michael.jon.odonnell@bristol.ac.uk

Disadvantages of pure R

- Has a higher technical bar to entry
  - Need to learn a programming language
  - Generally requires stronger mathematical knowledge to take full advantage
- Is not the current industry standard
  - Many in the industry unfamiliar
  - May not have time to learn
  - Is not accepted by most HTA bodies

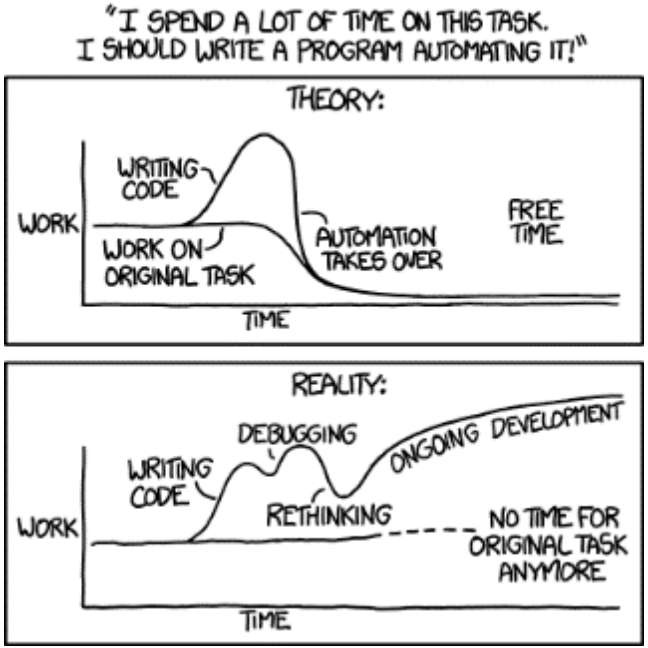


https://xkcd.com/927/

michael.jon.odonnell@bristol.ac.uk

Advantages of Automated Conversion

- Allows those unable to use R to still produce models suitable for use with modern techniques
- Allows for old models to be converted easily with minimal loss of productivity
- Prevents the lose of reproducibility if everyone moves away from Excel



https://xkcd.com/1319/

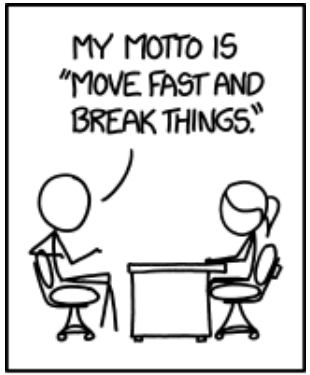
michael.jon.odonnell@bristol.ac.uk

Advantages of Automated Conversion

- Faster PSA calculations – Allows more complex models, or larger PSA sample sizes.
- Retains usability/familiarity of Excel
- Allows those with less time to learn the language to benefit from R (e.g. clinicians, HTA bodies, etc.)

michael.jon.odonnell@bristol.ac.uk

The Conversion Process



JOBS I'VE BEEN FIRED FROM

FEDEX DRIVER
CRANE OPERATOR
SURGEON
AIR TRAFFIC CONTROLLER
PHARMACIST
MUSEUM CURATOR
WAITER
DOG WALKER
OIL TANKER CAPTAIN
VIOLINIST
MARS ROVER DRIVER
MASSAGE THERAPIST

https://xkcd.com/1428/

michael.jon.odonnell@bristol.ac.uk

AI, Chat-GPT and machine learning

https://xkcd.com/1838/

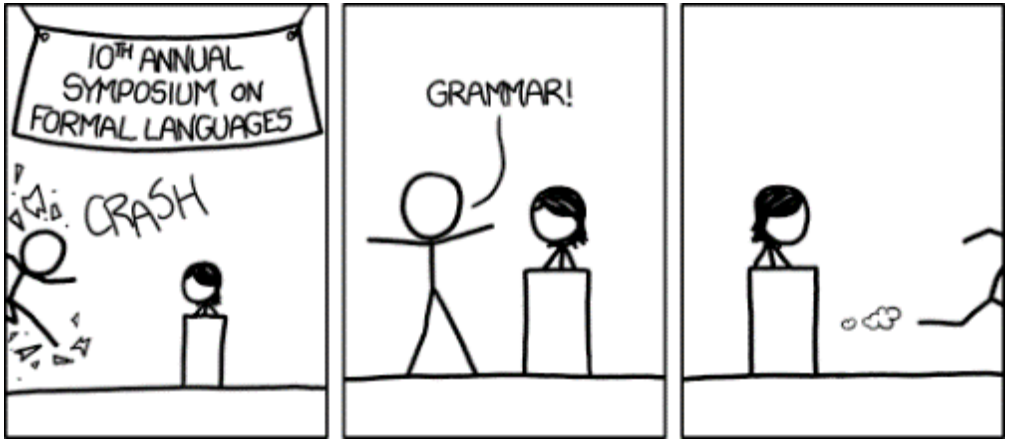
- Chat-GPT is a large language model, not suitable for complex programming
- Chat-GPT 4 will act confident while giving the wrong answer, so all inputs need checking via an expert
- Other AI style assistants more useful as trained specifically on code, however they face legal challenges: GitHub Copilot
- Machine learning excellent for result analysis and interpretation, but poor at code conversion

michael.jon.odonnell@bristol.ac.uk

How to convert Excel to R

- Read Excel with openpyxl
- Tokenize and create an Abstract Syntax Tree (AST)
- Traverse the AST and apply a grammar
- Order/sort the code
- Cull dead code
- Add peripherals

https://xkcd.com/1090/



michael.jon.odonnell@bristol.ac.uk

Excel Grammar

2.2.2 Formulas

The following ABNF grammar is used by formulas in other parts of this document.

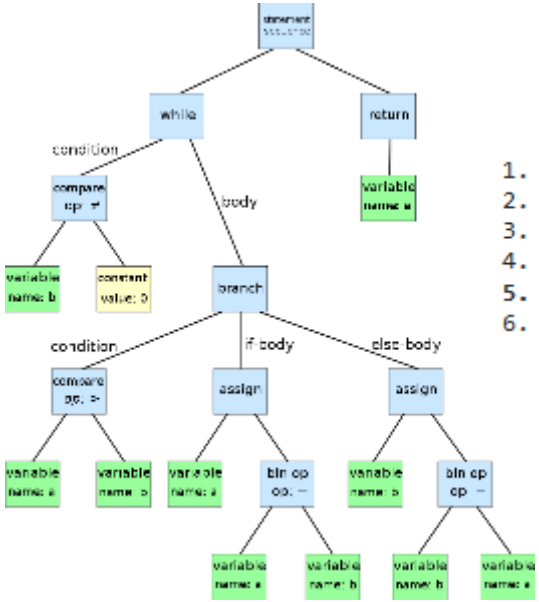
```
formula = expression

expression= ref-expression / *whitespace nospace-expression *whitespace
ref-expression= *whitespace ref-nospace-expression *whitespace
nospace-expression = "("  expression  ")" / constant / prefix-operator expression /
expression infix-operator expression / expression postfix-operator / function-call
ref-nospace-expression = "("  ref-expression  ")" / ref-constant / ref-expression ref-infix-
operator ref-expression / cell-reference / ref-function-call / name-reference / structure-
reference
constant = error-constant / logical-constant / numerical-constant / string-constant / array-
constant
ref-constant = "#REF!"
error-constant = ref-constant / "#DIV/0!"  /  "#N/A"  /  "#NAME?"  /  "#NULL!" / "#NUM!"  /
"#VALUE!" / "#GETTING_DATA"
logical-constant = "FALSE"  /  "TRUE"
numerical-constant = [neg-sign] significand-part [exponent-part]
significand-part = whole-number-part [fractional-part] / fractional-part
whole-number-part = digit-sequence
fractional-part = full-stop digit-sequence
exponent-part = exponent-character [ sign ] digit-sequence
full-stop = "."
sign = "+"  / neg-sign
neg-sign = "-"
exponent-character = "E"
digit-sequence = 1*decimal-digit
decimal-digit=  %x30-39
nonzero-decimal-digit = %x31-39
string-constant = double-quote [string-chars] double-quote
string-chars = string-char *string-char
string-char = escaped-double-quote / character ; MUST NOT be a double-quote
escaped-double-quote = 2double-quote
double-quote = %x22
;character = as defined by the production Char in the [W3C-XML] section 2.2
array-constant = "{" constant-list-rows "}"
constant-list-rows =  constant-list-row *(semicolon  constant-list-row)
semicolon =  ";"
constant-list-row = constant *(comma constant)
;An array-constant MUST NOT contain an array-constant or columns of unequal length or rows of
unequal length.
operator = ":" / comma / space / "^" / "*" / "/" / "+" / "-" / "&" / "=" / "<>" / "<" / "<="
/ ">" / ">=" / "%"
infix-operator = ref-infix-operator / value-infix-operator
value-infix-operator = "^" / "*" / "/" / "+" / "-" / "&"/ "=" / "<>" / "<" / "<=" / ">" /
">="
ref-infix-operator = range-operator / union-operator / intersection-operator
union-operator = comma
intersection-operator = space
range-operator = ":"
postfix-operator =  "%"
prefix-operator =  "+"  /  "-"
cell-reference = external-cell-reference / local-cell-reference
local-cell-reference = A1-reference
external-cell-reference = bang-reference / sheet-range-reference / single-sheet-reference
book-prefix = workbook-index "!"
```

- 25 Pages long
- Loosely follows the ABNF grammar system
- Allows anyone to write a “language” that can interpret the grammar
- Comp-sci degree advised

https://learn.microsoft.com/en-us/openspecs/office_file_formats/ms-xls/cd03cb5f-ca02-4934-a391-bb674cb8aa06

michael.jon.odonnell@bristol.ac.uk

Abstract Syntax Tree - General




```
1. while b ≠ 0:
2.     if a > b:
3.         a := a - b
4.     else:
5.         b := b - a
6. return a
```

- Walks the program
- Breaks code into constituent parts
- Builds a tree where each leaf is an elementary component of the program
- The tree is language agnostic

michael.jon.odonnell@bristol.ac.uk

Abstract Syntax Tree – Excel Tokenizer

= SUM(A1:A10)/SQRT(10)


```
{
    "py/object": "__main__.ProgramNode",
    "type": "Program",
    "body": [
        {
            "py/object": "__main__.ExpressionNode",
            "type": "FunctionCall",
            "name": "SUM(",
            "params": [
                {
                    "py/object": "__main__.OperandNode",
                    "type": "Operand",
                    "subtype": "RANGE",
                    "value": "A1:A10"
                },
                {
                    "py/object": "__main__.ParseNode",
                    "type": "FunctionClose",
                    "value": ")"
                }
            ]
        },
        {
            "py/object": "__main__.ParseNode",
            "type": "InfixOperator",
            "value": "/"
        },
        {
            "py/object": "__main__.ExpressionNode",
            "type": "FunctionCall",
            "name": "sqrt(",
            "params": [
                {
                    "py/object": "__main__.OperandNode",
                    "type": "Operand",
                    "subtype": "NUMBER",
                    "value": "10"
                },
                {
                    "py/object": "__main__.ParseNode",
                    "type": "FunctionClose",
                    "value": ")"
                }
            ]
        }
    ]
}
```


- Creates a JSON tree
- Each node contains
  - Type
  - Name
  - Then a combination of
    - Subtype
    - Value
    - Params

michael.jon.odonnell@bristol.ac.uk

Traverse and Translate

The broad strokes method

- Pick a cell
- Create AST
- Walk the AST
- Pick R equivalents
- Rebuild codeline

Caveats

- Cells picked from top left to bottom right.
- Each R equivalent needs to be written
- Need to store important syntactic data (variables, libraries, func type)

michael.jon.odonnell@bristol.ac.uk

Examples

```
('Inputparameters_F4', ['Inputparameters_F4 = Inputparameters_H4', ['Inputparameters_H4'], 'f'])
('Inputparameters_F5', ['Inputparameters_F5 = Inputparameters_H5', ['Inputparameters_H5'], 'f'])
('Inputparameters_F6', ['Inputparameters_F6 = Inputparameters_H6', ['Inputparameters_H6'], 'f'])
('Inputparameters_F7', ['Inputparameters_F7 = Inputparameters_H7', ['Inputparameters_H7'], 'f'])
('Inputparameters_F8', ['Inputparameters_F8 = Inputparameters_H8', ['Inputparameters_H8'], 'f'])
('Inputparameters_G8', ['Inputparameters_G8 = Inputparameters_F8', ['Inputparameters_F8'], 'f'])
('Inputparameters_F9', ['Inputparameters_F9 = Inputparameters_H9', ['Inputparameters_H9'], 'f'])
('Inputparameters_G9', ['Inputparameters_G9 = Inputparameters_F9', ['Inputparameters_F9'], 'f'])
```

```
('Inputparameters_G16', ['Inputparameters_G16 = qnorm(runif(numberOfRuns)%sep%Inputparameters_H16%sep%Inputparameters_I16)', ['Inputparameters_H16', 'Inputparameters_I16'], 'f'])
('Inputparameters_F17', ['Inputparameters_F17 = Inputparameters_H17', ['Inputparameters_H17'], 'f'])
('Inputparameters_G17', ['Inputparameters_G17 = qnorm(runif(numberOfRuns)%sep%Inputparameters_H17%sep%Inputparameters_I17)', ['Inputparameters_H17', 'Inputparameters_I17'], 'f'])
('Inputparameters_F18', ['Inputparameters_F18 = Inputparameters_H18', ['Inputparameters_H18'], 'f'])
('Inputparameters_G18', ['Inputparameters_G18 = qnorm(runif(numberOfRuns)%sep%Inputparameters_H18%sep%Inputparameters_I18)', ['Inputparameters_H18', 'Inputparameters_I18'], 'f'])
```


michael.jon.odonnell@bristol.ac.uk

Code Generation and Code Cull

- Need to reorder code
- Remove code that should not exist
- Explicitly write code in the correct format
- Handle slippages (None vs NA)

```python
def generate_code(self):
    """
    {'variable' : ["codeified string", ['list','of','contained','vars'],cell.data_type]}
    """
    with open(self.codefile,"w") as f:
        f.write(f"{self.mandatoryCode}")

        for item in self.culledcode.items():

            if item[1][2] != "f":
                f.write(f"{item[0]} = {item[1][0]}\n")
            else:
                item[1][0] = item[1][0].replace("%sep%", ",")
                f.write(f"{item[1][0]}\n")

        f.write(f"{self.outputs.add_output_code()}")
```


michael.jon.odonnell@bristol.ac.uk

REEEVR - R and External Integration

- Creation of a REEEVR library
  - Contains converted functions to mimic Excel behaviour
  - Contains functions to allow validation
  - Contains functions to allow integration with other packages
- External integrations
  - Currently hooks into BCEA to generate CE/CEACs
  - Potential to hook into any R package if desired
  - Qian Xin currently investigating MLMC integration

michael.jon.odonnell@bristol.ac.uk

Decision Tree - Excel

Inputs		Notes
Willingness-to-pay	20000	
Analysis type	Probabilistic	
Include MI?	Yes	*Base case includes MI but a sensitivity analysis is to set MI probabilities to zero*
Outputs		
ICER Warfarin vs aspirin	-1414.250708	
INB Warfarin vs aspirin	-31115.37592	

Run PSA

- Minimum viable model
- Contains only PSA
- Doesn’t generate graphs
- Contains VBA

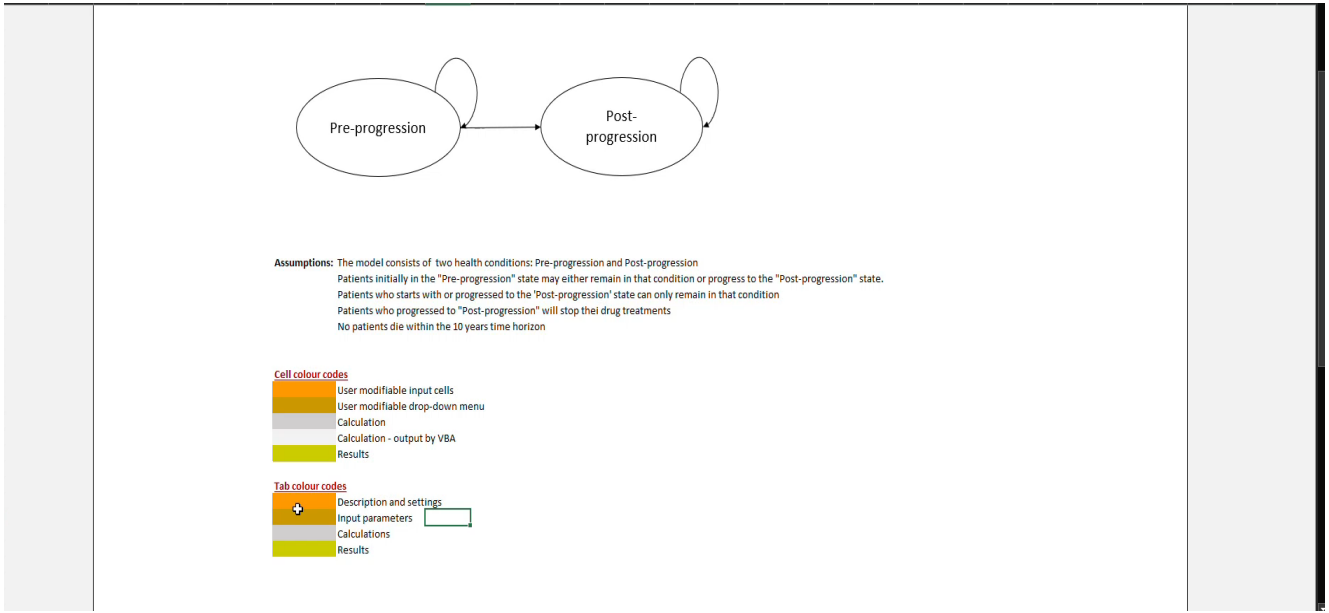
michael.jon.odonnell@bristol.ac.uk

Decision Tree – R (250 lines)

```r
library(reeevr)
library(BCEA)
numberofRuns = 1000
willingness_to_pay = 20000
analysis_type = 'Probabilistic'
Inputparameters_J4 = 10
Inputparameters_K4 = 90
Inputparameters_J5 = 2
Inputparameters_K5 = 78
Inputparameters_J6 = 4
Inputparameters_K6 = 156
Inputparameters_J7 = 30
Inputparameters_K7 = 70
Inputparameters_J8 = 4
Inputparameters_K8 = 76
Inputparameters_J9 = 1
Inputparameters_K9 = 99
Inputparameters_H10 = 10
Inputparameters_H11 = 500
Inputparameters_H13 = 400
Inputparameters_H14 = 2000
Inputparameters_H16 = 12
Inputparameters_I16 = 1
Inputparameters_H17 = 20
Inputparameters_I17 = 2
Inputparameters_H18 = 12
Inputparameters_I18 = 1
Costing_E4 = 5000
Costing_E5 = 400
Costing_E6 = 100
Costing_E7 = 150
Costing_E8 = 100
Costing_E9 = 100
QALYs_D4 = 14
QALYs_D5 = 15
QALYs_D6 = 12
QALYs_D7 = 16
QALYs_D8 = 14
QALYs_D9 = 14
QALYs_D10 = 15
QALYs_D11 = 13
QALYs_D12 = 12
QALYs_D13 = 16
QALYs_D14 = 16
QALYs_D15 = 10
QALYs_D16 = 14
QALYs_D17 = 15
QALYs_D18 = 14
QALYs_D19 = 13
QALYs_D20 = 17
QALYs_D21 = 12
QALYs_D22 = 18
QALYs_D23 = 14
average_costs_aspirin = 712.5050882589587
average_costs_warfarin = 2767.442383559008
average_incremental_costs = average_costs_warfarin - average_costs_aspirin
```


michael.jon.odonnell@bristol.ac.uk

2S Markov Model - Excel



- 2 – State Markov Model
- PSA and DSA
- Excel sheet creates graphs
- Contains VBA

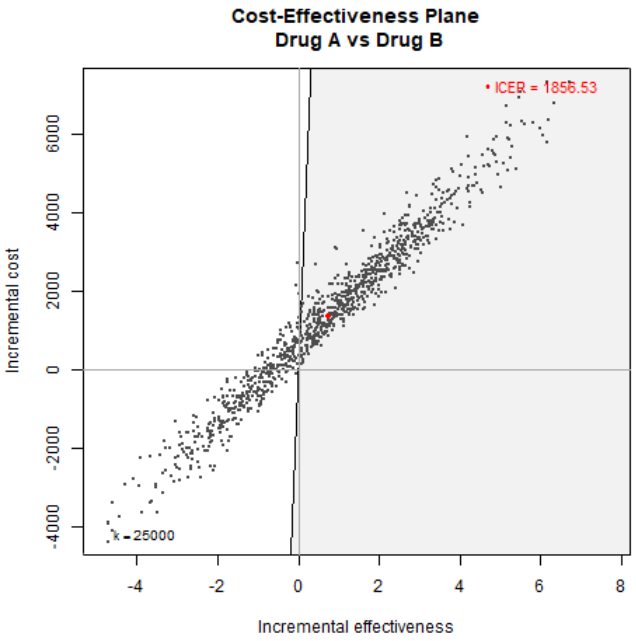
michael.jon.odonnell@bristol.ac.uk

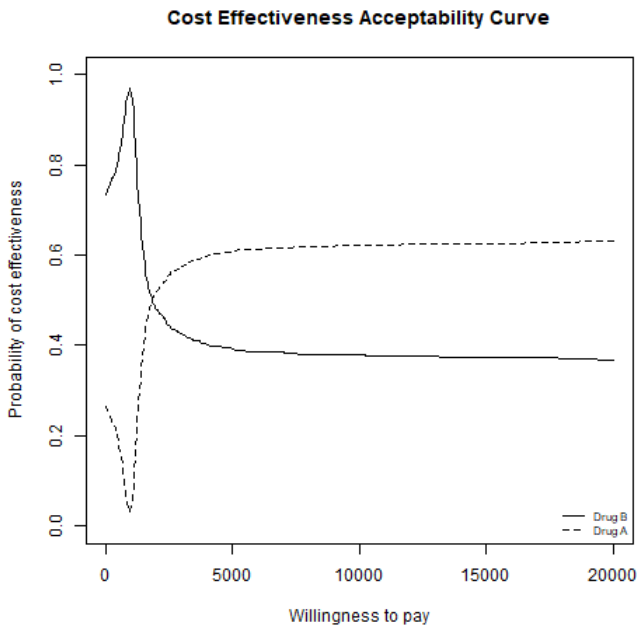
2S Markov Model – R – (400 lines)

```r
library(reeevr)
library(BCEA)
numberofRuns = 1000
willingness_to_pay = 20000
discount_cost = 0.03
discount_healthOutcome = 0.03
p_se = 0.1
p_lower = 0.025
p_upper = 0.975
Modelparameters_I14 = NA
Include_DSA = NA
Modelparameters_G16 = 0.8
Modelparameters_H16 = Modelparameters_G16 * p_se
Modelparameters_I16 = 'Yes'
Modelparameters_L16 = excel_iferror((Modelparameters_G16*(Modelparameters_G16*(1-Modelparameters_G16)/(Modelparameters_H16^2)-1)),"")
Modelparameters_M16 = excel_iferror(Modelparameters_L16*(1-Modelparameters_G16)/Modelparameters_G16,"")
Modelparameters_N16 = qbeta(p_lower,Modelparameters_L16,Modelparameters_M16)
Modelparameters_O16 = qbeta(p_upper,Modelparameters_L16,Modelparameters_M16)
Modelparameters_P16 = excel_iferror(qbeta(runif(numberofRuns),Modelparameters_L16,Modelparameters_M16),Modelparameters_G16)
Modelparameters_Q16 = 4
p_hh_A = excel_choose(Modelparameters_Q16,Modelparameters_G16,Modelparameters_N16,Modelparameters_O16,Modelparameters_P16)
Modelparameters_G17 = 0.2
Modelparameters_H17 = Modelparameters_G17 * p_se
Modelparameters_I17 = 'No'
Modelparameters_L17 = excel_iferror((Modelparameters_G17*(Modelparameters_G17*(1-Modelparameters_G17)/(Modelparameters_H17^2)-1)),"")
Modelparameters_M17 = excel_iferror(Modelparameters_L17*(1-Modelparameters_G17)/Modelparameters_G17,"")
Modelparameters_N17 = qbeta(p_lower,Modelparameters_L17,Modelparameters_M17)
Modelparameters_O17 = qbeta(p_upper,Modelparameters_L17,Modelparameters_M17)
Modelparameters_P17 = excel_iferror(qbeta(runif(numberofRuns),Modelparameters_L17,Modelparameters_M17),Modelparameters_G17)
Modelparameters_Q17 = 1
p_hs_A = excel_choose(Modelparameters_Q17,Modelparameters_G17,Modelparameters_N17,Modelparameters_O17,Modelparameters_P17)
Modelparameters_G18 = 1
Modelparameters_I18 = 'No'
Modelparameters_N18 = NA
Modelparameters_O18 = NA
Modelparameters_P18 = NA
Modelparameters_Q18 = 1
p_ss_A = excel_choose(Modelparameters_Q18,Modelparameters_G18,Modelparameters_N18,Modelparameters_O18,Modelparameters_P18)
Modelparameters_G19 = 0.7
Modelparameters_H19 = Modelparameters_G19 * p_se
Modelparameters_I19 = 'Yes'
Modelparameters_L19 = excel_iferror((Modelparameters_G19*(Modelparameters_G19*(1-Modelparameters_G19)/(Modelparameters_H19^2)-1)),"")
Modelparameters_M19 = excel_iferror(Modelparameters_L19*(1-Modelparameters_G19)/Modelparameters_G19,"")
Modelparameters_N19 = qbeta(p_lower,Modelparameters_L19,Modelparameters_M19)
Modelparameters_O19 = qbeta(p_upper,Modelparameters_L19,Modelparameters_M19)
Modelparameters_P19 = excel_iferror(qbeta(runif(numberofRuns),Modelparameters_L19,Modelparameters_M19),Modelparameters_G19)
Modelparameters_Q19 = 4
p_hh_B = excel_choose(Modelparameters_Q19,Modelparameters_G19,Modelparameters_N19,Modelparameters_O19,Modelparameters_P19)
Modelparameters_G20 = 0.3
Modelparameters_H20 = Modelparameters_G20 * p_se
Modelparameters_I20 = 'No'
Modelparameters_L20 = excel_iferror((Modelparameters_G20*(Modelparameters_G20*(1-Modelparameters_G20)/(Modelparameters_H20^2)-1)),"")
Modelparameters_M20 = excel_iferror(Modelparameters_L20*(1-Modelparameters_G20)/Modelparameters_G20,"")
Modelparameters_N20 = qbeta(p_lower,Modelparameters_L20,Modelparameters_M20)
Modelparameters_O20 = qbeta(p_upper,Modelparameters_L20,Modelparameters_M20)
Modelparameters_P20 = excel_iferror(qbeta(runif(numberofRuns),Modelparameters_L20,Modelparameters_M20),Modelparameters_G20)
```


michael.jon.odonnell@bristol.ac.uk

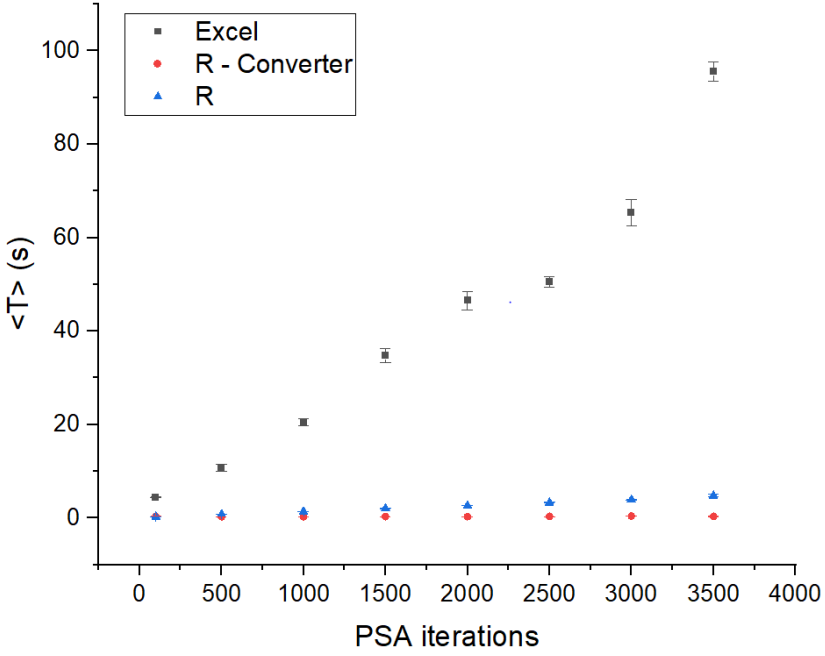
2S Markov Model - Automated BCEA output

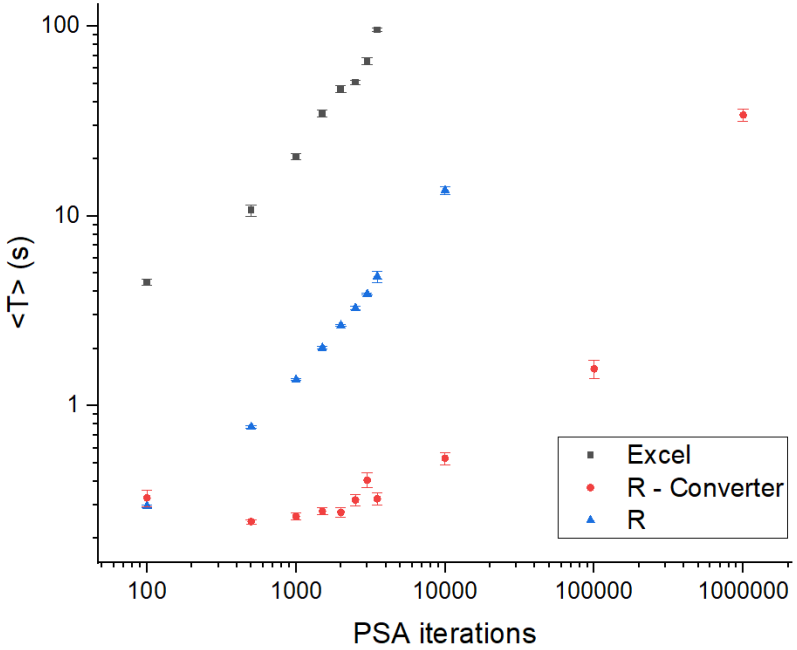




michael.jon.odonnell@bristol.ac.uk

Speed comparison for HIPS model





michael.jon.odonnell@bristol.ac.uk

Demonstration – HIPS model (2600 lines)

michael.jon.odonnell@bristol.ac.uk

Validation – show hips output

- Current state of Validation
  - Validation code automatically added to R output
  - Works by propagating known values through R to give known costs/utilities
  - Requires user to add equivalent validation to Excel model (change inputs to known default values)
- To be implemented
  - Validation via mean/median values to simplify Excel model changes
  - Suggestions welcome!

michael.jon.odonnell@bristol.ac.uk

Limitations and Caveats

- DSA may not be implemented
- If DSA implemented, likely to be median DSA, not mean DSA
- The full suite of Excel functions is not currently implemented
- The GUI requires additional usability work
- The R output, while “readable” is difficult to follow

michael.jon.odonnell@bristol.ac.uk

Future Work

- Unit and Integration testing
  - Currently utilise basic black box testing
  - Unit testing needed for Excel conversions
- Openpyxl updates to tokenizer (3.2?)
- Addition of missing Excel functions
- Testing on wider range of Excel models
  - Prior work guides development
- Integration of MLMC and BCEA more completely via optional commands

https://xkcd.com/1421/

michael.jon.odonnell@bristol.ac.uk