Going Beyond Excel Vs. R

An Introduction To Visual Programming For Health Economics Modelling

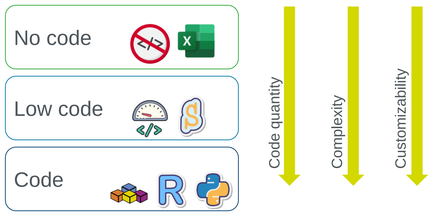
JE. Poirrier [1], J. Vanderpuye-Orgle [2]

MSR44

View all Parexel's posters at ISPOR 2025

Background

- Decision-makers within the healthcare sector are faced with the choice between numerous competing healthcare interventions and programs, of which only a proportion can be provided with the available resources [1]. Data input, calculation, and result presentation for these choices are modelled in the Cost-Effectiveness Model or Budget Impact Model, for instance.
- Nowadays, this modelling is performed with general-purpose of specialised software.
  - MS Excel, a general-purpose spreadsheet software, is often the default choice for most health economics models (HEM) because of its wide availability, simplicity to approach and universal acceptance by Health Technology Assessment (HTA) agencies worldwide. However, because of its general-purpose nature, representing a model in Excel can quickly become tedious, and adapting a model to a different structure is often more complex than re-coding it from scratch. Finally, Excel performs poorly in compute-intensive tasks (e.g. sensitivity analysis, some model structure, …).
  - R has grown popular for over a decade [2]. This statistical programming language provides more transparency [3], automation and reproducibility [4], flexibility (based on packages dedicated to HEMs) and user-friendliness (based on the use of Shiny [5]). However, because it is based on code, its approach is tedious for non-programmers, model structures are still complex to adapt (unless the modeler did a proper analysis including potential adaptations), and most HTA agencies have not entirely accepted models written in R yet.
- Between Code and No-Code, Low-Code is a recent way of programming that presents low code requirements but allows for complexity and customizability (Figure 1):
  - A low-code development platform provides a development environment for creating application software, generally through a graphical user interface. A low-code platform may produce entirely operational applications or require additional coding for specific situations.



- Can low-code programming facilitate quick prototyping and the exploration of alternate model structures?

Methods

- This research presents the concept of visual programming, a programming method that lets health economists create models by manipulating program elements graphically rather than specifying them textually (Figure 2).
  - In visual programming, elements are linked (and unlinked) by dragging connectors (with a mouse in an editor). These connectors ensure information is transferred between graphical elements and determine the execution order (Figure 2).
  - Specifically, in this research, we are using Flyde [6] and its Visual Studio Code (v. 1.94.0) extension (v. 0.105.2) on an MS Windows 11 laptop (Figure 3).
  - Flyde developers created a web version, further simplifying the use of visual programming in a potentially restricted IT environment. Although this is not investigated here, concepts can be applied similarly.
- Each graphical element can represent model concepts at different levels:
  - simple programming elements ("if", "switch", …),
  - simple modelling concepts (unit costs, market share projections, adverse events, e.g.),
  - complex modelling concepts (like tree node, health state, sensitivity analyses),
  - or even encapsulate a whole Markov trace or model.
  - Some graphical elements can also represent functions modulating the expected behaviour of other elements (like discounting, allowing vial sharing or not, …) or manage data input and output (including charting and report generation).
  - These levels and the abstraction they contain allow for a quick learning curve of the different visual elements within a familiar environment (Figure 4).
- Due to its visual nature, this versatility allows for faster prototyping, allowing modellers to build and test ideas quickly, for instance, while building an early model.
  - Adding a fourth health state ("Recover" in Figure 5) is only a matter of cloning an existing "health state" element, changing its behaviour (code) and reconnecting the model appropriately.
- Elements and connectors can be automatically translated into classical modelling languages (Excel or R, for instance), and these implementations could be of higher quality (with several rounds of specialised review) and well documented.
- Table 1 summarises the benefits and challenges of Visual Programming.

Results

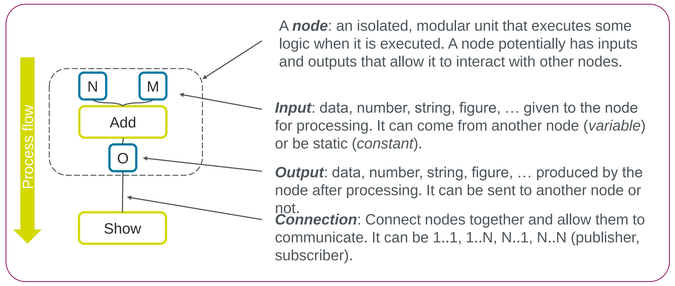


Figure 2: Elements of Visual Programming

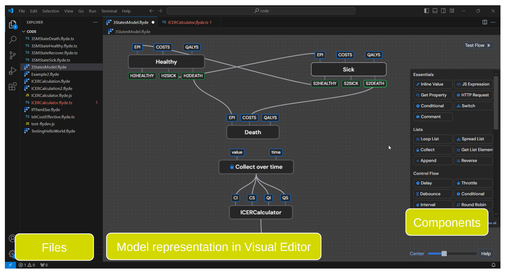
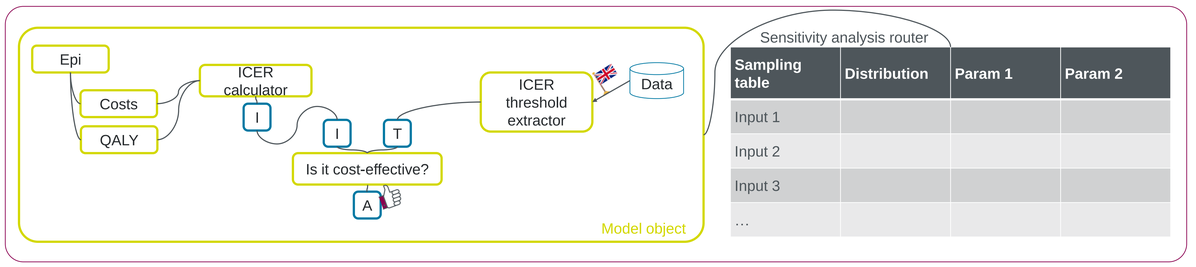


Figure 3: Visual Programming with Flyde in VS Code



Sampling table	Distribution	Param 1	Param 2
Input 1			
Input 2			
Input 3			
…			

Figure 4: Example of two abstraction levels used to represent a cost-effectiveness model: model details and model object (used for sensitivity analysis)

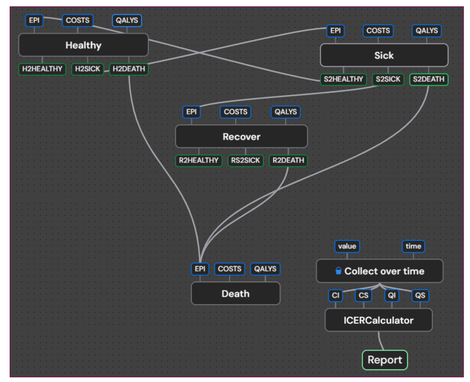


Figure 5: Example of quick model prototyping by the addition of a fourth health state (compared to Figure 3)

Benefits	Challenges
Faster prototyping › Build and test ideas faster, quick prototyping › Fast exploration of alternate model structure › Also: educational use, learning curve Expandability (reuse), versatility, abstraction Higher quality › Visual debugging › Provided each element is QC'ed and adequately documented Automated translation to other "classical" programming languages Open Source (Flyde)	Programming › Yet another language to program nodes (Typescript for Flyde) Readability › Use of space, multiple connections (see Figure 5) Reproducibility › Early model vs. HTA model Performances? Open Source › Flyde is under development

Table 1: Benefits and challenges of Visual Programming for Health Economics modelling

Conclusions

- This research presents the concept of visual programming of health economics models.
  - We show the quick learning curve to use and connect the visual elements into a meaningful model
- Visual Programming can be used in health economics modelling programming
- Visual Programming allows for faster model prototyping & easy exploration of alternate model structures
  - It still allows for an in-depth understanding of the code underneath
  - It also preserves the possibility of transforming them into more classical languages (like R or Python)
- Highly reviewed nodes can be building blocks of high-quality Visual Programming models
- Further practical implementations and visualisation aid will follow this research

REFERENCES

[1] Griffin, S. and K. Claxton (2011). Analyzing uncertainty in cost-effectiveness for decision-making. The Oxford Handbook of Health Economics. S. Glied and P. C. Smith. Oxford, Oxford University Press.
[2] Jalal, H., et al. (2017). "An Overview of R in Health Decision Sciences." Med Decis Making 37(7): 735-746.
[3] Alarid-Escudero, F., et al. (2019). "A Need for Change! A Coding Framework for Improving Transparency in Decision Modeling." Pharmacoeconomics 37(11): 1329-1339.
[4] Smith, R. A., et al. (2022). "Living HTA: Automating Health Economic Evaluation with R." Wellcome Open Res 7: 194.
[5] Smith, R. and P. Schneider (2020). "Making health economic models Shiny: A tutorial." Wellcome Open Res 5: 69.
[6] Flyde: https://www.flyde.dev/ (last visited: 10 April 2025)

AFFILIATIONS
[1] HEOR Modelling, Parexel International, Wavre, Belgium
[2] Quantitative Insights, Parexel International, Los Angeles, USA

© 2025 Parexel International (MA) Corporation www.parexel.com