Identical Outputs, Divergent Defensibility: LLM Interaction Style in Health-Economic Modeling

EE169

Benjamin White, MSc[1], Anshul Shah, BPharm, MSc[2], Sandra Milev*, MSc[1]
*Presenting author
[1]Red Nucleus, Yardley, PA, USA; [2]Red Nucleus, London, UK
Supplementary handout · ISPOR Europe 2026, Vienna · Poster Session 2, Monday 9 November

At a glance

We rebuilt one validated Excel cost-effectiveness model twice as an R package, from identical specifications and inputs, varying only how the analyst worked with the LLM. Both builds produced near-identical results (ICERs 0.4% apart) and tied on numerical correctness. They diverged on defensibility: code quality 86.9% vs 99.5% and analyst understanding 66.2% vs 82.5% (SIMPLE vs MODULAR), with the gap concentrated in the implementation decisions the specification left open. This handout gives the detail behind the poster.

1 Study design

	SIMPLE	MODULAR
How work was requested	One broad, end-to-end request for the whole model engine, written as a competent, good-faith prompt rather than a strawman	Component by component, in small verifiable slices
Who designed the structure	The LLM owned all decisions not in the specifications or prompt, choosing the modules, data objects and internal mechanisms; no interfaces were defined in advance	Before any LLM implementation, the analyst wrote a contract for each component: purpose, inputs, outputs and units, invariants, source requirement, known ambiguities, isolated check and integration point
Verification	Comparison against Excel deferred until a full or near-full model run existed	Each component checked in isolation (known values, hand calculation, specification or workbook) before integration
Implementation order	Built first, from a single prompt written before any MODULAR work; the analyst did not inspect the result, and it was not repaired before assessment	Built second, component by component
Assessment (quiz) order	Quizzed second	Quizzed first

Why this order: building SIMPLE first kept its single prompt a good-faith request, uninformed by anything learned while building MODULAR. Quizzing MODULAR first means anything the analyst picked up from the first quiz could only help the SIMPLE answers; the reverse order would have biased the comparison toward MODULAR.

Held constant: the same analyst, byte-identical build specification and parameter inputs, and the same R package tooling and scaffolding. The full model specification was held out of both builds and used only as the grading answer key. The reference is a validated and anonymized Markov cost-effectiveness model in a metabolic disease (10 alive health states, intervention vs comparator); each model was built as an R package varying only interaction style.

2 How the models were graded

Quiz scoring scale

Score	Meaning
0	No meaningful answer, or materially wrong
1	Recognises the topic but misses the central mechanism
2	Partly correct, with important gaps or unsupported claims
3	Main mechanism correct, minor omissions
4	Complete, specific, defensible, tied to the implementation

Nine closed-book questions per model, answered from memory (build notes allowed, no source code). An answer that only restates the specification on a question about the implementation is capped at 2.

Grading panels

Each evaluation was scored by an independent panel of 10 LLM graders (Claude Opus) per model, using the same prompt for every item; graders were not told which approach was expected to do better. Scores were averaged by script and grader spread recorded. Code-quality results had one reconciliation pass per model; understanding items were reconciled where grader scores spread by 2 or more points (SIMPLE only). Reconciliation adjudicated contradictions without re-grading. Graders also flagged answers (definitions on page 2).

3 What each measure means

Code quality evaluation

Category / dimension	What it measures	Weight
C1 Correctness and evidence	**Is the model right, and can that be shown?**	**58%**
Traceability to specification	Each requirement can be followed from the specification to the code that implements it and the test that checks it	14%
Test design and adequacy	Tests check meaningful results at the right points, not just that the code runs	16%
Numerical correctness (Markov, HEOR)	The Markov mechanics (cycles, transitions, discounting, half-cycle correction) are implemented correctly, whether or not they match Excel	16%
Input validation; invariants enforced at run time	Inputs are validated, and rules such as "probabilities sum to one" are checked automatically on every model run	12%
C2 Structure and maintainability	**Is the code organised so it can be changed safely?**	**16%**
Separation of concerns	Each piece of code owns one coherent task and is not tangled up with others	10%
Maintainability and extensibility	A new analyst could add a treatment arm, swap a component or add a scenario without rediscovering hidden structure	6%
C3 Transparency and honesty	**Does the package tell the truth about itself?**	**22%**
Transparency of assumptions and divergences	Ambiguous choices, and any departures from the specification or workbook, are recorded next to the code that depends on them	8%
Benchmark honesty	The code and documentation describe agreement with Excel truthfully, and any agreement rests on sound structure rather than coincidence	6%
Documentation accuracy	Comments and design documents are accurate and current, so a reader can follow a non-obvious decision to the code	8%
C4 Package craft	**Is it built to R package standards?**	**4%**
R-package craft	A real R package with a clear interface, help files and passing package checks, not a script in package form	4%

Each dimension is scored 0–4 against written anchors, with cited code evidence. The composite is the weighted mean, shown as % of the maximum.

Analyst understanding evaluation

Category	Questions (S/M)	What it measures
Model flow	2 / 2	Explaining from memory the path from inputs to ICER, the order of events within a cycle, and how the submodels combine
Sensitivity reasoning	4 / 5	Explaining why results move when settings change: drug cost, the disease-progression treatment effect, the lipid pathway, discontinuation (MODULAR only) and structural settings
Benchmark reconciliation	1 / 1	Judging whether a divergence from Excel is a coding error, a configuration difference or an open ambiguity
Self-assessment	1 / 1	The analyst's own assessment of what they can explain versus what they would need to rediscover, graded on how honest and well calibrated it is
Design ownership	1 / 0	Which design boundaries the LLM chose (SIMPLE only; in MODULAR the analyst set them)

S/M = SIMPLE / MODULAR. Category scores on page 3 are weighted means of question scores; categories with one question are descriptive only.

Grader flags (quiz answers)

Flag	What it means
Restated the specification	The answer restated the written specification without explaining how the model implements it
Overclaimed	The answer asserted implementation behaviour more confidently than the evidence supported
Code search needed	The answer needed source code despite a closed-book question
Implementation insight	The answer explained an implementation choice that is not obvious from the specification

Counts per model are on page 3.

4 Results in full

Code quality (% of maximum)

Dimension	Weight	SIMPLE	MODULAR	Gap
Composite		**86.9**	**99.5**	**+12.6**
C1 Correctness, evidence	**58%**	**81.2**	**100**	**+18.8**
Run-time invariant checks	12%	60.0	100	+40.0
Test design and adequacy	16%	75.0	100	+25.0
Spec traceability	14%	85.0	100	+15.0
Numerical correctness	16%	100	100	0
C2 Structure	**16%**	**98.1**	**99.1**	**+1.0**
Separation of concerns	10%	100	100	0
Maintainability	6%	95.0	97.5	+2.5
C3 Transparency, honesty	**22%**	**91.8**	**98.2**	**+6.4**
Documentation accuracy	8%	80.0	95.0	+15.0
Assumption transparency	8%	97.5	100	+2.5
Benchmark honesty	6%	100	100	0
C4 Package craft	**4%**	**97.5**	**100**	**+2.5**
R-package craft	4%	97.5	100	+2.5

Gap = MODULAR − SIMPLE, in percentage points. **Behind the largest gaps:** SIMPLE contains input and invariant checks, but they are never called when the model runs; and its design record states zero adverse-event burden for the intervention arm while its code computes, and a test asserts, a non-zero value.

Package size: context, not quality

Metric	SIMPLE	MODULAR
R source files	11	19
R code lines	1,022	3,106
Comment-to-code ratio	0.49	1.29
Test files	10	20
Test blocks	45	525
Test assertions	141	1,476
Line coverage	99.4%	100%
R CMD check errors / warnings / notes	0 / 0 / 0	0 / 0 / 0
Recorded design decisions	16	25

Both builds pass every automated check with near-complete coverage: the difference is how deeply behaviour is tested, not whether lines run. Larger is not automatically better.

Analyst understanding (% of maximum)

Category	N (S/M)	SIMPLE	MODULAR	Gap
Composite	**9 / 9**	**66.2**	**82.5**	**+16.3**
Model flow	2 / 2	56.3	76.3	+20.0
Sensitivity reasoning	4 / 5	65.3	86.5	+21.2
Benchmark reconciliation	1 / 1	67.5	70.0	+2.5
Self-assessment	1 / 1	80.0	82.5	+2.5
Design ownership	1 / 0	75.0	–	–

Category scores are weighted means of question scores. Categories with one question are descriptive only.

Same topic, both quizzes

Question topic	SIMPLE	MODULAR	Gap
Lipid-pathway sensitivity	65.0	100	+35.0
Drug-cost sensitivity	50.0	75.0	+25.0
How submodels combine each cycle	50.0	75.0	+25.0
Disease-progression treatment effect	80.0	95.0	+15.0
Pipeline and cycle ordering	62.5	77.5	+15.0
Structural scenarios	66.0	70.8	+4.9
Benchmark reconciliation	67.5	70.0	+2.5
Self-assessment	80.0	82.5	+2.5

MODULAR scored at least as well on all 8 shared topics. Unpaired questions: SIMPLE design ownership 75.0%; MODULAR discontinuation sensitivity 87.5%. Structural scenarios cover time horizon, discounting and treatment continuation.

Grader flags (count)

Flag	SIMPLE	MODULAR
Restated the specification	7	0
Overclaimed	14	1
Code search needed	2	0
Implementation insight	9	21

Counts are summed over 10 graders × 9 questions per model. Flag definitions are on page 2.

5 Model outputs vs the Excel reference

Incremental outcomes (discounted, per patient)	SIMPLE	MODULAR	Excel	SIMPLE − Excel	MODULAR − Excel
ICER (£/QALY)	49,845*	50,048	52,828	−2,983 (−5.6%)	−2,780 (−5.3%)
Incremental costs (£)	53,975	53,861	48,088	+5,887 (+12.2%)	+5,773 (+12.0%)
Incremental QALYs	1.083	1.076	0.910	+0.173 (+19.0%)	+0.166 (+18.2%)
Incremental life-years	0.977	0.968	0.615	+0.362 (+58.8%)†	+0.353 (+57.4%)†

Per-arm totals (discounted, per patient)	SIMPLE	MODULAR	Excel	SIMPLE vs Excel	MODULAR vs Excel
Intervention arm					
Total costs (£)	95,221	96,502	90,675	+5.0%	+6.4%
Total QALYs	10.833	10.854	10.031	+8.0%	+8.2%
Total life-years	14.840	14.865	13.806	+7.5%	+7.7%
Comparator arm					
Total costs (£)	41,245	42,641	42,587	−3.2%	+0.1%
Total QALYs	9.750	9.778	9.121	+6.9%	+7.2%
Total life-years	13.863	13.898	13.191	+5.1%	+5.4%

The two builds agree with each other to within 1% on every incremental outcome (ICER 0.4%, costs 0.2%, QALYs 0.6%, life-years 0.9%). *SIMPLE values are from the post-repair build; the graded version's ICER was 49,464 (1.2% from MODULAR).

† Reading the 58%

Incremental life-years are a small difference between two large totals (about 14.8 and 13.9 life-years per patient). Both builds estimate slightly longer survival than Excel in both arms (5–8% on the totals). The excess is a little larger in the intervention arm, so the difference between arms grows by about 0.36 life-years, roughly 4 months per patient. Against an Excel increment of only 0.61 life-years, that is 58%: a large percentage of an already small number. Percentage differences in incremental outcomes should be compared with caution; the absolute differences and per-arm totals are the better guide. The ICER, which divides two inflated quantities, moves least (about −5%).

What the gap is and is not: inputs were checked and match Excel. The gap traces to the survival calculations and likely reflects implementation decisions unrelated to interaction style (e.g., decisions about tunnel states). It is shared by both builds in size and direction, and was deliberately not diagnosed further because it does not bear on the comparison between them.

6 Interpreting the results

Verify. A quick single-pass build is a legitimate independent check that a result reproduces. Here it matched the careful build to within 1%.
Build. Single-pass delegation should not build the model a team defends: it carried unenforced invariants, thin tests, documentation contradicted by code, and an analyst who could not fully explain it.
Validate. Agreement between builds is not proof of correctness. Both builds agreed, and both missed Excel by the same margin: replication catches build-specific errors, not errors shared through common inputs or specifications.

Questions to ask of any LLM-assisted model

- ☐ Are invariants (e.g., probabilities summing to one) enforced every time it runs?
- ☐ Do the tests check behaviour, not just line coverage?
- ☐ Does each input and equation trace to a source?
- ☐ Does the documentation match the code?
- ☐ Has it been reconciled against an independent benchmark?

7 Limitations and future work

- Single analyst and one model: an illustrative case study, not a powered experiment.
- Fixed order: SIMPLE was built first, which may favour MODULAR (built with more experience); MODULAR was assessed first, which may favour SIMPLE (assessed later). The biases act on different things and their sizes are unknown.
- The analyst was not blinded to the arm. Closed-book conditions relied on the analyst's honesty.
- LLM graders measured LLM-induced debt, and package folder names revealed the arm to graders.
- Scope is implementation-level debt under a fixed specification; design-level debt was not tested. Specification-answerable questions score near ceiling in both arms, compressing differences.
- The code-open quiz (a further nine questions per model) was prepared but not taken; only the closed-book quiz was answered and graded.

Future work: replicate with multiple analysts and models; re-grade under arm-neutral names; add a reduced-specification case to probe design-level debt; take and grade the code-open quiz.

Reference. Kosmyna N, Hauptmann E, Yuan YT, et al. Your brain on ChatGPT: accumulation of cognitive debt when using an AI assistant for essay writing task. arXiv:2506.08872; 2025.
Disclosures. All authors are employees of Red Nucleus. LLMs were used to build both models and as graders.
Abbreviations. HEOR, health economics and outcomes research; ICER, incremental cost-effectiveness ratio; LLM, large language model; QALY, quality-adjusted life-year.

Contact
Benjamin White
bwhite@rednucleus.com