R FOR HTA 2026

Turning a model “description” into R code that you can actually trust

A Decomposition Framework for Translating Published Health Economic Model Descriptions into Executable R Code Using Agentic AI

BARIS DENIZ · AIDE SOLUTIONS · R FOR HTA 2026 · 24 JUNE 2026

THE PREMISE

Modeling has always been a translation activity

- **A model is a translation.** Our understanding of a disease and its treatments, rendered as states and transitions.
- **It runs on tacit knowledge.** Modeling intuition and good practice shape the structure. And it works.
- **So write-ups leave the obvious unstated.** The reader fills the gaps, because the reader is a modeler too.

Running example throughout

Heinz et al. 2022 (J Med Econ)
JAK inhibitor vs dupilumab
Atopic dermatitis · UK NHS · Markov cohort
16-week cycle · Lifetime horizon

The discontinuation question

When the paper says "discontinuation," does it mean:
a state transition?   an arm reassignment?   a model exit?

A trained modeler knows immediately. The paper doesn't say.

THE FAILURE MODE

Machines don't share our tacit knowledge

- **LLMs fill gaps silently.** Trained to produce an answer, they invent a resolution to reach one.
- **The code runs either way.** Faithful extraction and silent guess look identical.
- **The rule isn't in the trials.** Heinz's stopping rule comes from NICE context a modeler imports. A machine wouldn't.

The skill is knowing where not to let AI improvise.

Single-step LLM approach

Article text → *one prompt* → R code

Silent assumptions buried in the output. No review surface. No auditability. Code runs; correctness unknowable.

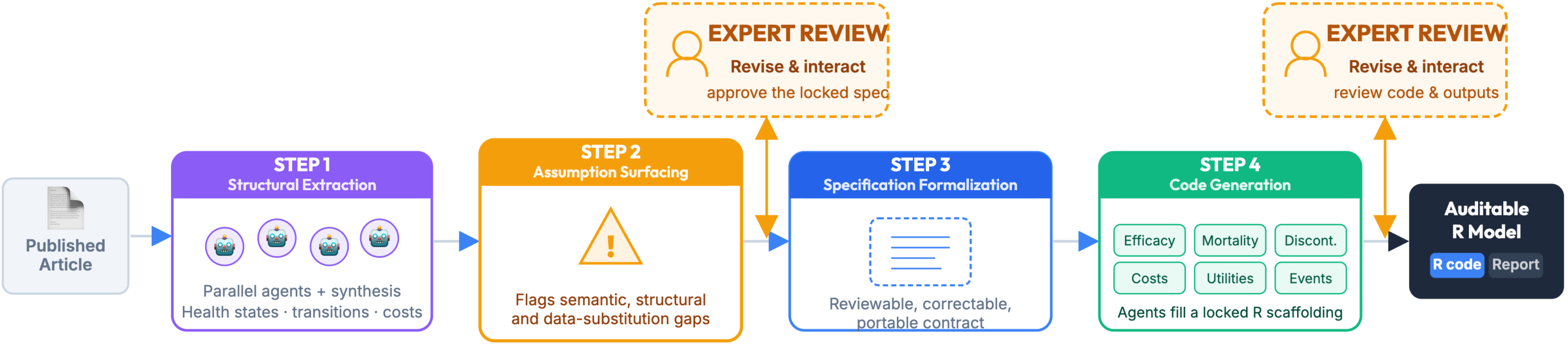
Decomposition approach

Article text → *four verifiable steps* → R code

Every assumption surfaced and reviewed before any code is written.

THE FRAMEWORK

Four discrete, independently verifiable steps



Each step yields a reviewable artifact, with the expert at the controls. Nothing is built until the spec is approved.

1 STRUCTURAL EXTRACTION

Identify, cite, verify

Five parallel agents, one per domain, pull the model's structure from the text, every element tagged to its source.

- **Model structure & health states**
- **Settings:** horizon, cycle, discount, perspective
- **Population & treatment strategies**
- **Parameters:** efficacy, transitions, costs, utilities
- **Analysis plan:** outputs, scenarios

Every element carries a citation. Clear where the paper is explicit; Step 2 flags where it is silent.

STRUCTURAL EXTRACTION · HEINZ 2022

```
HEALTH STATES (normalized)
Induction · EASI-50 · EASI-75 · EASI-90 · No Response · Death (absorbing)
⚠ "Induction BSC" + "BSC EASI-50" consolidated into base states; BSC limits
enforced via zero transition probabilities

SETTINGS
cycle: 16 weeks horizon: lifetime discount: 3.5% (NICE reference case)
perspective: UK NHS/PSS currency: 2020 GBP

POPULATION
age 38 · 60% male · 29-yr mean disease duration ← NICE TA534

SOURCED PARAMETERS
transitions ← Table 1 utilities ← Table 2 costs ← Table 3
BSC efficacy ← LIBERTY AD CHRONOS (placebo arm)

Snapshot · the full extraction is shown in the demo
```

Implicit and explicit assumptions, surfaced for review

A careful pass over the proposed structure collects the model's assumptions and puts each one back to the researcher.

Explicit assumptions

Stated in the paper. Straightforward to carry into the model.

Implicit assumptions

Modelers apply these from accumulated knowledge, usually without a second thought. **This is where AI fails silently**, filling the gap with whatever it judges appropriate.

Example: “discontinuation”

A modeler knows it means the patient **moves to best supportive care.** An LLM does not, and could as readily model it as a **switch to another therapy**, an **exit from the model**, or **death**. Each is a different structure, and the code runs either way.

Collect every assumption, implicit and explicit, and put it to the researcher:

here is the proposed approach. Is it right for your context?

The contract between extraction and implementation

The reviewed assumptions, formalized into a machine-readable contract.

- **The review gate lives here:** locked before any code agent starts.
- **Revise, annotate, or override,** upstream of code generation.
- **Tooling-independent:** works with any coding assistant.

Documented, versioned, shareable. Not buried in a prompt.

MODEL SPECIFICATION · HEINZ 2022 · V1.0

```
FIVE SECTIONS
1  Model description, settings & population
2  Natural disease progression
3  Treatment efficacy & safety
4  Costs & utilities
5  PSA, outcomes & assumptions

RESOLVED ASSUMPTIONS (from review)
discontinuation → Induction, BSC line (16-wk stopping rule) ⚠ NICE TA534; not in trials
EASI-90 loss → drop below EASI-50 (no inter-response transitions) ⚠ confirmed per structure
upa 30 mg price = 2 × BNF upa 15 mg ⚠ unlicensed; proxy
BSC efficacy = LIBERTY AD CHRONOS placebo arm

✓ Reviewed & locked · code generation authorised
```

4 CODE GENERATION

R generated from the spec, not from prose

- **Deterministic scaffolding.** Markov trace, discounting, ICER, half-cycle: fixed code, not AI-generated.
- **AI fills disease-specific content** inside a locked architecture.
- **Transparent, standard R** any reviewer can read.

Agents see only the approved spec, never the article. **Hallucination has no surface to land on.**

GENERATED R · MARKOV ENGINE (EXCERPT)

```r
# Deterministic scaffolding (identical across models, not AI-generated)

# Discounting (cycle 0 weight = 1)
func_discount_vector <- function(n_cycles, r, cycle_years) {
  1 / (1 + r)^((0:(n_cycles - 1)) * cycle_years)
}

# Markov trace: time-dependent transition array a_P[,,t]
m_M <- matrix(0, n_cycles + 1, n_states,
        dimnames = list(0:n_cycles, v_states))
m_M[1, ] <- v_start_dist # cycle 0
for (t in 1:n_cycles)
  m_M[t + 1, ] <- m_M[t, ] %*% a_P[, , t] # age-dependent

# Half-cycle correction (trapezoidal), before discounting
v_hcc <- rep(1, n_cycles); v_hcc[1] <- 0.5; v_hcc[n_cycles] <- 0.5
v_disc <- func_discount_vector(n_cycles, r = 0.035, cycle_years = 16/52)
total <- function(v) sum(v * v_hcc * v_disc)

# ICER (spec-derived outcomes)
ICER <- (total(v_cost_upa) - total(v_cost_dup)) /
      (total(v_qaly_upa) - total(v_qaly_dup))
# Target: £219,733.88 / QALY (Heinz 2022, Table 4)
```

DEMONSTRATION

From specification to R code: Heinz 2022

1 · Extract 2 · Surface assumptions 3 · Review spec 4 · Build R

What it shows

Extract → surface assumptions → review spec → generate R, end to end.

Watch for

The discontinuation assumption surfaced in Step 2, confirmed before any code is written.

AUDITABILITY

Every element traces back to where it came from

A full provenance chain from source to output. The judgment that used to live in the modeler's head is now documented.

PUBLISHED ARTICLE
Every extracted element cited to source location
(Fig, Table, paragraph)

→

ASSUMPTION REVIEW
Every assumption surfaced
Approve · question · override
(who decided; on what basis)

→

APPROVED SPEC
All assumptions explicit
Locked before coding
(version-controlled)

→

R CODE
Generated from spec
Standard, readable R
(any reviewer can audit)

→

TECHNICAL REPORT
Full provenance
From article to output
(reproducibility package)

Trace any number in the R output back to the source and the decision that shaped it.

TAKEAWAYS

What the framework gives you

1. **"Code that runs" becomes a model you can trust.** A reviewed, documented spec and plain, auditable R take the black box out of AI-built models.
2. **Tooling-independent.** The spec step is adoptable on its own, with any coding assistant.
3. **Surfacing assumptions and data gaps is the differentiator.** Making the implicit explicit, before any code, separates responsible AI from silent hallucination.

Transparency, traceability, and control.

Every decision surfaced, every number traceable to its source, the expert deciding at each gate.

Questions welcome. Baris Deniz · barisdeniz@aidesolutions.net