PolicyBench

Benchmarking no-tool tax-and-benefit estimation in frontier language models

Max Ghenis*

Table of contents

Abstract 1

1 Introduction 2

2 Related work 2

3 Benchmark design 4
- 3.1 Headline metric: exact match . . . 4
- 3.2 Near-miss companion: within 1% . . . 5
- 3.3 Secondary metric: bounded continuous score . . . 5
- 3.4 Frozen snapshot and open-set status . . . 7

4 Data and scenario construction 16
- 4.1 United States . . . 16
- 4.2 Disability in the household facts . . . 17
- 4.3 Reference-output credibility . . . 19

5 Results 37
- 5.1 United States leaderboard . . . 37
- 5.2 Uncertainty . . . 37
- 5.3 Cost and latency . . . 38
- 5.4 Simple baselines . . . 40
- 5.5 Sensitivity to benchmark view . . . 41
- 5.6 Hardest benchmark targets . . . 42
- 5.7 Zero and positive cases . . . 43
- 5.8 Open-weight time lag . . . 43

*PolicyEngine and Axiom Foundation.

6 Failure modes 45

7 Limitations 47

8 Conclusion 48

Appendix A: Structured-output audit 49

Competing interests 50

Source: Article Notebook

Abstract

PolicyBench evaluates whether frontier language models can estimate household tax and benefit outputs from household facts without tools. This release covers the United States, drawing sampled households from the certified PolicyEngine populace microdataset and scoring answers against reference outputs from policyengine-us 2.60.0. It evaluates 51 frontier models on 100 households across 18 output groups, and reports three complementary scores: an exact-match rate as the headline deployability bar (a prediction counts only if it matches the PolicyEngine reference to the dollar for amounts or to the eligibility flag for booleans), a within-1% hit rate as a near-miss-tolerant companion, and a continuous bounded score that awards partial credit for close answers. The headline is exact match on a household-impact-weighted leaderboard: 85% of US reference outputs are exact zeros, and the weighting concentrates scores in the high-dollar output groups a hedge-to-zero strategy cannot answer, so the top model leads an always-zero baseline of 69.6% weighted exact by 25.5 points (Section 3.1 details the construction). The manuscript snapshot is a 100-household public preview with no protected held-out set: the current public scenario explorer exposes prompts and reference outputs, so open-set leakage is a central limitation of any public ranking. In the frozen snapshot used for this manuscript (2026-10-27), the top-scoring US model on exact match is GPT-6 Sol at 95.2%, while GPT-5.4 nano is lowest at 65.3%. Within model families, newer is not always stronger: Claude Opus 4.8 scores 75.8% exact, below the 80.5% of the earlier Claude Opus 4.7. Multi-step tax quantities and positive-dollar benefit cases are materially harder than zero cases. The frozen audit annotations cover 9,050 rows whose legacy threshold score is below 1. That universe contains 9,046 of the snapshot's 9,046 exact-match misses and 4 exact hits; 2,407 additional rows with bounded score below 100 were not selected or annotated. Every reference-suspect flag was adjudicated against primary sources and engine traces. PolicyBench rebuilt the references on 2026-10-23 with policyengine-us 2.60.0, the newest release when it began sweeping them that day. Each scored reference follows law published before PolicyBench froze the references on 2026-07-03. Where the engine projects a 2026 amount or takes one from a later publication, the reference holds the amount published before the freeze or, where none was published, the last amount published. PolicyBench excludes from scoring, for every model, the 47 outputs

whose references depend on an input the prompt does not state and the two whose references rest on an amount a state published after the freeze. No excluded output rests on an engine defect the audits found that policyengine-us 2.60.0 still has. The 0.7% of cells with a missing or unparsable answer stay in the denominator as misses. The live benchmark is available at https://policybench.org.

1 Introduction

Household tax-and-benefit estimation sits between arithmetic and policy discussion. Each case has numeric labels, but generating those labels requires filing status, household composition, income concepts, program thresholds, and jurisdiction-specific rules. PolicyBench measures whether models can map household records to those outputs.

Most current language-model evaluations do not test this mapping directly. General math benchmarks emphasize symbolic manipulation and exact final answers (Cobbe et al. 2021; Hendrycks et al. 2021), while generic question-answering benchmarks emphasize recall or instruction following. PolicyBench instead asks whether models can transform household facts into policy outputs without access to tools or simulators.

This matters for public-facing calculators, analyst workflows, and tax-preparation or screening systems. The benchmark is intended to separate verbal policy fluency from household-level quantitative prediction.

2 Related work

The most relevant prior work is tax and statutory reasoning. SARA frames statutory interpretation, including tax-relevant rule application, as a language-understanding problem (Holzenberger et al. 2021). LegalBench broadens this view and includes tax-oriented numeric tasks such as sara_numeric (Guha et al. 2023). RuleArena evaluates rule-guided reasoning in regulation-style settings (Zhou et al. 2025), and TaxCalcBench evaluates frontier models on tax calculation from structured return-like inputs (Bock et al. 2025). Shanahan et al. (2025) report a similar split on the Volunteer Income Tax Assistance (VITA) test: models do better on tax knowledge questions than on open-ended calculation tasks.

PolicyBench also sits within a broader literature on quantitative reasoning benchmarks. Canonical math evaluations such as GSM8K and MATH normalized exact final-answer scoring for numeric tasks (Cobbe et al. 2021; Hendrycks et al. 2021). Recent work has questioned pure exact-match scoring in numeric QA and temporal reasoning, arguing for error-aware metrics instead (Abbood et al. 2025). PolicyBench treats this as a measurement design choice: it leads with a household-impact-weighted exact-match rate and reports near-miss and bounded

companions; the metric definitions, and the weighting that keeps the zero-inflated panel from rewarding hedge-to-zero answers, are specified in the benchmark design section.

Finance-domain benchmarks provide another relevant precedent. FinQA and TAT-QA both show that realistic quantitative reasoning often requires combining structured numbers with domain knowledge, which stylized school-math problems do not test (Chen et al. 2021; Zhu et al. 2021). PolicyBench differs in generating its reference outputs with executable tax-benefit microsimulation, where those benchmarks annotate questions about documents, but the underlying motivation is similar: domain-specific quantitative reasoning deserves dedicated evaluation.

The nearest public-benefits benchmark is the concurrent Public Benefits Bench, built by Vals AI with the Center for Civic Futures and Code for America (Vals AI 2026). It evaluates models on 459 expert-validated SNAP guidance scenarios (230 in the published test set) spanning all 50 US states plus Guam and the Virgin Islands, and grades free-text answers against expert-written rubric criteria using a judge model that agreed with SNAP policy experts on 80.6% of grading decisions. Where PolicyBench isolates parametric knowledge by withholding tools, Public Benefits Bench varies tool access directly: across models, multi-turn conversation raises pass rates by roughly 7.6 percentage points and web search by about 6.9. The two benchmarks probe complementary failure surfaces — conversational guidance quality for a single program with tools available, versus no-tool numeric estimation across tax and transfer outputs scored deterministically against microsimulation references — and reach consistent conclusions. The best model passes 71.7% of rubric criteria under Public Benefits Bench's strongest condition, and its authors conclude that "no general-purpose AI model performs well enough to be trusted with SNAP benefits guidance," matching the deployability gaps PolicyBench reports for unaided calculation.

Finally, PolicyBench depends on two infrastructure literatures that are not themselves large language model (LLM) benchmarks. One is structured-output reliability, since the benchmark relies on multi-output JSON responses and parse coverage (Shorten et al. 2024). The other is tax-benefit microsimulation, where systems such as EUROMOD provide the methodological precedent for evaluating policy rules over household microdata (Sutherland and Figari 2013). There is also operational work on applying and evaluating AI systems in public-benefits settings, including caseworker-assist and Supplemental Nutrition Assistance Program (SNAP)-focused evaluations (Nava Labs 2026, 2025; ZenML LLMOps Database 2025). PolicyBench combines these strands into a public cross-model benchmark over household-level tax and benefit outputs.

3 Benchmark design

The canonical PolicyBench request asks each model to predict every benchmark output for a household in a whole-scenario response under its structured answer contract. Of the 51 models, 10 received the documented chunked accommodation and instead answered the same prompt

template over one- or three-output subsets, reducing each response's completion demand while preserving the household-level task.

3.1 Headline metric: exact match

The headline metric is the exact-match rate: the share of requested outputs the model gets right to the dollar (or, for eligibility flags, to the boolean). This is the deployability bar. A tax filer, a benefit estimator, or a caseworker cannot ship "close" — a prediction off by $50 is no more usable than one off by $500, because neither matches the bottom line a downstream system or a household needs. For currency amounts, "exact" means within 1 currency unit of the reference value after numeric parsing. For binary outputs, "exact" means the parsed value is exactly the same 0/1 flag as the reference.

Exact match can be a deceptive headline in a zero-inflated panel — but only when the rate is computed unweighted. In the frozen manuscript snapshot, 85% of reference outputs are exact zeros: most sampled households are not eligible for any given program, owe no capital gains tax, do not phase into a refundable credit, and so on. A model that hedges to zero on every requested output earns full exact credit on every zero case, so a naive unweighted exact rate is dominated by those zeros — a hedge-to-zero model scores roughly 85% unweighted, leaving little room above the floor to separate models that actually understand the rules.

The public leaderboard corrects for this by household-impact weighting. Each output group's weight is its population-average share of the household's dollar stake, computed as |ref| / max(|household_net_income|, Σ |ref|) per source household and averaged with calibrated household weights over the full populace population (Section 5.5 gives the full construction). Zero-reference rows are not removed — every row carries its output group's weight — but the weighting concentrates mass in high-dollar output groups, which are rarely zero, so the share of weight on zero-reference rows falls from their 85% row share to 69.6%. An always-zero baseline therefore scores 69.6% weighted exact, against the 85% it earns unweighted, and the top model clears that baseline by 25.5 points — a margin comparable to within-1%. Across the 51-model roster, the weighted exact rate spreads about as widely as within-1% and ranks models in nearly the same order, so exact match discriminates between rule comprehension and zero-hedging while preserving the to-the-dollar deployability meaning.

The exact rate is aggregated to a household score and then to a country score, with the same population-derived output weights that the within-1% companion and the bounded score use. Equal household weight preserves comparability across scenarios and slices. The leaderboard reports US results on this fixed test set.

3.2 Near-miss companion: within 1%

The within-1% hit rate is the near-miss-tolerant companion to the headline: the share of requested outputs the model gets within one percent of the PolicyEngine reference value. For

currency amounts, "within 1%" means |pred − ref| 0.01 × |ref| when the reference is nonzero and |pred| 1 (the same one-currency-unit absolute tolerance used by exact match) when the reference is zero. Binary outputs are requested as integer 0/1 eligibility flags and scored identically to the exact-match metric.

Within-1% relaxes the exact bar by tolerating rounding and small rate or parameter drift on positive-reference cases, so it credits a model that is consistently close but rarely to-the-dollar. It uses the same per-output household-impact weighting and the same household-equal aggregation as the headline exact rate, and it is reported alongside exact in every leaderboard table. Because both metrics share the weighting, the two rank models almost identically; within-1% sits a few points above exact and is the right view when the question is near-miss accuracy; exact match is the view for strict deployability.

3.3 Secondary metric: bounded continuous score

PolicyBench also reports a bounded 0-100 score that awards smooth partial credit for close-but-not-exact amount answers. This is informative for tracking conceptual progress year over year and captures the qualitative gap between a model that is consistently close and one that is consistently far from the reference value. For amount variables, the row-level bounded score is max(0, 1 - |pred - ref| / |ref|) when the reference value is nonzero and 1{pred = 0} when the reference value is zero. For binary outputs, it uses exact 0/1 matching. Note that the zero-reference branch is stricter than the headline's one-currency-unit tolerance — a $0.50 prediction on a zero-reference amount counts as exact but earns no bounded credit — so the bounded score is not a uniform relaxation of exact match. The exact, within-1%, and within-10% hit rates are reported separately as threshold diagnostics; they are not averaged to create the bounded score.

PolicyBench also tracks mean absolute error and related error metrics. These remain secondary diagnostics under the headline exact-match rate, the within-1% near-miss companion, and the bounded score. The three-metric design follows recent numeric-evaluation literature that questions pure exact-match scoring (Abbood et al. 2025) while keeping exactness as the production bar.

All requested US outputs are annual amounts or annual eligibility indicators for tax year 2026, scored under the tolerances defined above. Missing, unparseable, or non-0/1 answers receive zero score for that requested output.

Aggregation proceeds in three steps. First, each household-output prediction receives a 0-100 score. Second, each household receives one model score: requested output rows are weighted by output-group weights constructed from the full weighting population, then renormalized within the household over outputs that are actually requested for that household. Person-level coverage flags are scored at the person row; their output-group weight is split across the relevant people in the household and is based on PolicyEngine value proxies where available. Third, the country score averages those household scores with equal household weight. The leaderboard

reports US results on this fixed test set. Equal-output-group scores and other alternative views are reported as sensitivity checks.

The benchmark requires each model response to include numeric answers and one explanation per requested output. Both the exact-match rate and the bounded score use only the numeric answers. Explanations are retained for scenario exploration and qualitative error analysis; they should not be interpreted as faithful traces of model reasoning.

Because explanations are required, the canonical task measures policy estimation under a public-facing structured-response contract, of which isolated arithmetic accuracy is one part. Prompt fairness is part of the benchmark contract. The current release uses one prompt template per country, with no model-specific tuning. Models receive the same household facts and requested outputs. They receive no web or external tool access. The prompt sets unlisted numeric inputs to 0, unlisted boolean or status facts to false, and household characteristics as constant over the tax-benefit year. Provider-specific differences are limited to structured-output transport, request shape, and the reasoning setup in Table 2. The harness sends no reasoning-control or sampling parameters. It leaves reasoning effort, temperature, and related decoding controls at provider defaults. The frozen GPT-5.5 wave uses provider-default reasoning effort.

The forced tool_choice suppresses Claude's extended thinking in this snapshot (sensitivity/claude-thinking-2026-08.md). A labeled sensitivity with tool_choice: "auto" is published beside the board. Claude Fable 5.1, released September 1, 2026, closes the interaction from the API side: it rejects forced tool use with an error, so its row answers through the JSON transport and reasons at the provider default; a sensitivity run with the tool declared under tool_choice: "auto" scores 0.8 points above that row. Claude Opus 5.5, released September 21, 2026, and Claude Sonnet 5.5, released September 28, 2026, reject forced tool use the same way, so their rows also answer through the JSON transport and reason at the provider default; neither has a sensitivity run. The next board version moves every model to tool_choice: "auto" so each provider's default reasoning posture can engage.

Two dimensions on which requests were not identical across models are request shape and answer transport (Table 2). Of the 51 models, 10 could not reliably complete the whole-scenario request because their serving stacks rejected the structured-output call, exhausted completion budgets mid-response, or timed out. They instead answered the same prompt template over subsets of the requested outputs, one to three outputs per request. Subsetting gives each response fewer requested outputs and a fresh completion budget over the same household facts, so the accommodation, if it moves scores at all, should favor the chunked models; their scores are not strictly comparable to whole-scenario scores. This accommodation is closed going forward: a model added after this snapshot either answers the canonical whole-scenario request or is listed as not scorable, without accommodation.

One row, GLM-5.3-Flash (preview), answered as a cloaked OpenRouter listing called "Ox Alpha" and is labeled a preview. It was publicly callable under the canonical whole-scenario request and forced-tool transport, and its date is its OpenRouter listing date. After the row's run, Z.ai identified the listing as GLM-5.3-Flash (OpenRouter 2026); the row carries the

disclosed name and keeps the preview qualifier, because whether the checkpoint that answered is the one later released under that name has not been stated, and it is distinct from the GLM-5.3 row, which is Z.ai's larger model called under its own id. Its artifact key stays ox-alpha, the name it was queried under.

3.4 Frozen snapshot and open-set status

Leaderboard positions are version-sensitive, so manuscript claims refer to the frozen source-run exports in Table 1; the live site can change after them. The committed source-run exports are the manuscript artifacts. The public site exposes the current prompts, predictions, explanations, and reference outputs for transparency. This makes the public leaderboard open-set: models, model providers, or benchmark users could learn from the released cases before later runs. Protected leaderboard claims would require a separate held-out or rotating set.

Source: Article Notebook

Table 1: Frozen manuscript snapshot.

Item	Value
Snapshot date	2026-10-27
Model response date	2026-06-12 to 2026-10-26
Policy period	US tax year 2026
Frozen export	paper/snapshot/20260501/runs/us_full_run_20260612_policyengine_4_16_1_populace/data.json.gz
Frozen export SHA-256 prefix	US b0db08bc6ec5
Snapshot manifest	paper/snapshot/20260501/manifest.json
Audit annotations	annotations/us_full_run_20260612_policyengine_4_16_1_populace
US scenarios SHA-256 prefix	71b16212f0c0
US reference outputs SHA-256 prefix	eab09a0c184c
Benchmark spec SHA-256 prefix	89067ba731e8
Frozen prompt payload SHA-256 prefix	US 1c3f94362d0b
Scoring code SHA-256 prefix	0fdef4505de8
US run label	us_full_run_20260612_policyengine_4_16_1_populace
Household sample seed	42
Reference engine	policyengine-us 2.60.0
PolicyEngine.py (recorded for provenance)	6.1.2
US dataset	populace_us_2024 (populace-us-2024-5da5a95-20260611)
Households	100 US

Table 1: Frozen manuscript snapshot.

Item	Value
Models	51
Output groups	18 US
Condition	No tools, no web access; structured responses covering every requested output
Response contract	Numeric answer and non-empty explanation for every requested output

Source: Article Notebook

Table 2: Model run configuration in the frozen manuscript snapshot. Every model answers through its row's transport, with no external tools, provider-default temperature and sampling, no reasoning-control parameters, and the same household facts and requested outputs. Transport is the answer contract: a forced answer-schema tool call (tool_choice set to the answer tool) or a JSON object, the latter where the model card or its family default selects JSON, for most such rows because the provider rejects a forced tool call. Supervised-run fingerprints pin answer contract, request shape, and completion ceiling for 22 rows; tool choice for 21 rows; reasoning setup and timeouts for 12 rows. Reasoning setup and timeouts for the other ten fingerprinted rows, and all fields for the other 29 rows, are the harness registry as frozen in the snapshot's serving-configuration file. Rows with an output count answered the template over subsets of the requested outputs, an accommodation that predates the canonical whole-scenario rule described in the text. Released is the first public availability (paid tiers count; trusted-tester previews do not), compiled from vendor announcements and contemporaneous press; grok-build-0.1's date rests on secondary trackers. Leaderboard artifact keys match the final segment of the provider id (qwen-3.7-max adds a hyphen). A dagger marks open-weight models; Kimi K3's weights, announced at its API launch (Willison 2026), shipped on Hugging Face on July 27, 2026 under a custom license (Tech Times 2026). GLM-5.3-Flash (preview) was queried as the OpenRouter stealth listing Ox Alpha (artifact key ox-alpha); Z.ai identified the listing as GLM-5.3-Flash after its run (OpenRouter 2026), and its Released date is its August 20, 2026 OpenRouter listing date.

Model	Provider ID	Released	Parsed	Transport	Request shape	Reasoning setup
Claude Fable 5	claude-fable-5	2026-06-09	1,935/ 1,935	forced tool	1 output/ request	thinking does not engage under forced tool call
Claude Fable 5.1	claude-fable-5-1	2026-09-01	1,935/ 1,935	JSON	whole scenario	provider default; 16,384-token shared budget
Claude Haiku 4.5	claude-haiku-4-5-20251001	2025-10-15	1,935/ 1,935	forced tool	1 output/ request	provider default

Model	Provider ID	Released	Parsed	Transport	Request shape	Reasoning setup
Claude Haiku 5.5	claude-haiku-5-5	2026-10-07	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
Claude Opus 4.7	claude-opus-4-7	2026-04-16	1,935/ 1,935	forced tool	1 output/ request	provider default
Claude Opus 4.8	claude-opus-4-8	2026-05-28	1,935/ 1,935	forced tool	1 output/ request	provider default
Claude Opus 5	claude-opus-5	2026-07-24	1,935/ 1,935	forced tool	whole scenario	thinking does not engage under forced tool call
Claude Opus 5.5	claude-opus-5-5	2026-09-21	1,935/ 1,935	JSON	whole scenario	provider default; 16,384-token shared budget
Claude Sonnet 4.6	claude-sonnet-4-6	2026-02-17	1,935/ 1,935	forced tool	1 output/ request	provider default
Claude Sonnet 5	claude-sonnet-5	2026-06-30	1,935/ 1,935	forced tool	1 output/ request	thinking does not engage under forced tool call
Claude Sonnet 5.5	claude-sonnet-5-5	2026-09-28	1,935/ 1,935	JSON	whole scenario	provider default; 16,384-token shared budget

Model	Provider ID	Released	Parsed	Transport	Request shape	Reasoning setup
DeepSeek V4 Flash 0731	openrouter/deepseek/deepseek-v4-flash-0731	2026-07-31†	1,935/1,935	forced tool	whole scenario	provider default; 98,304-token shared budget
DeepSeek V4 Pro	deepseek/deepseek-v4-pro	2026-04-24†	1,935/1,935	JSON	whole scenario	provider default; 16,384-token shared budget
DeepSeek V4 Pro 0813	openrouter/deepseek/deepseek-v4-pro-0813	2026-08-13†	1,935/1,935	forced tool	whole scenario	provider default; 98,304-token shared budget
DeepSeek V4.1 Flash	deepseek/deepseek-flash	2026-09-10†	1,935/1,935	JSON	whole scenario	provider default; 16,384-token shared budget
Gemini 3 Flash Preview	gemini/gemini-3-flash-preview	2025-12-17	1,935/1,935	JSON	whole scenario	provider default
Gemini 3.1 Flash Lite Preview	gemini/gemini-3.1-flash-lite-preview	2026-03-03	1,935/1,935	JSON	whole scenario	provider default
Gemini 3.1 Pro Preview	gemini/gemini-3.1-pro-preview	2026-02-19	1,935/1,935	JSON	whole scenario	provider default
Gemini 3.5 Flash	gemini/gemini-3.5-flash	2026-05-19	1,935/1,935	JSON	whole scenario	provider default

Model	Provider ID	Released	Parsed	Transport	Request shape	Reasoning setup
Gemini 3.5 Flash-Lite	gemini/ gemini-3.5-flash-lite	2026-07-21	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
Gemini 3.6 Flash	gemini/ gemini-3.6-flash	2026-07-21	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
Gemini 3.7 Flash	gemini/ gemini-3.7-flash	2026-08-13	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
Gemini 3.8 Flash	gemini/ gemini-3.8-flash	2026-09-02	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
Gemini 4 Argon	gemini/ gemini-4-argon	2026-10-13	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
GLM-5.2	open-router/ z-ai/ glm-5.2	2026-06-13†	1,799/ 1,935	JSON	3 outputs/ request	provider default; 16,384-token shared budget

Model	Provider ID	Released	Parsed	Transport	Request shape	Reasoning setup
GLM-5.3	open-router/ z-ai/ glm-5.3	2026-08-14†	1,860/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
GPT-5.4 mini	gpt-5.4-mini	2026-03-17	1,935/ 1,935	forced tool	whole scenario	provider default
GPT-5.4 nano	gpt-5.4-nano	2026-03-17	1,935/ 1,935	forced tool	whole scenario	provider default
GPT-5.5	gpt-5.5	2026-04-23	1,935/ 1,935	forced tool	3 outputs/ request	provider default; 16,384-token shared budget
GPT-5.6 Luna	gpt-5.6-luna	2026-07-09	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
GPT-5.6 Sol	gpt-5.6-sol	2026-07-09	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
GPT-5.6 Terra	gpt-5.6-terra	2026-07-09	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
GPT-6 Astra	gpt-6-astra	2026-09-04	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget

Model	Provider ID	Released	Parsed	Transport	Request shape	Reasoning setup
GPT-6 Luna	gpt-6-luna	2026-09-22	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
GPT-6 Sol	gpt-6-sol	2026-09-22	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
GPT-6.1 Sol	gpt-6.1-sol	2026-09-29	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
GPT-6.1 Terra	gpt-6.1-terra	2026-10-26	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
Grok 4.3	xai/ grok-4.3	2026-04-17	1,935/ 1,935	forced tool	whole scenario	provider default
Grok 4.5	xai/ grok-4.5	2026-07-08	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
Grok 4.6	xai/ grok-4.6	2026-08-12	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget

Model	Provider ID	Released	Parsed	Transport	Request shape	Reasoning setup
Grok 4.7	xai/ grok-4.7	2026-09-21	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
Grok Build 0.1	xai/grok-build-0.1	2026-05-29	1,935/ 1,935	forced tool	whole scenario	provider default
Inkling	open-router/ thinking-machines/ inkling	2026-07-15†	1,935/ 1,935	forced tool	whole scenario	provider default; 49,152-token shared budget
Kimi K2.6	open-router/ moon-shotai/ kimi-k2.6	2026-04-20†	1,538/ 1,935	JSON	3 outputs/ request	provider default; 16,384-token shared budget
Kimi K3	open-router/ moon-shotai/ kimi-k3	2026-07-16†	1,874/ 1,935	JSON	whole scenario	provider default; 49,152-token shared budget
MiniMax M3	open-router/ minimax/ minimax-m3	2026-06-01†	1,935/ 1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
Muse Spark 1.3	open-router/ meta/ muse-spark-1.3	2026-09-02	1,935/ 1,935	JSON	whole scenario	provider default; 16,384-token shared budget

Model	Provider ID	Released	Parsed	Transport	Request shape	Reasoning setup
Muse Wonder	openrouter/meta/muse-wonder	2026-10-12	1,935/1,935	JSON	whole scenario	provider default; 16,384-token shared budget
GLM-5.3-Flash (preview)	openrouter/stealth/ox-alpha	2026-08-20	1,935/1,935	forced tool	whole scenario	provider default; 16,384-token shared budget
Qwen3.7-max	openrouter/qwen/qwen3.7-max	2026-05-19	1,935/1,935	JSON	3 outputs/request	provider default; 16,384-token shared budget
Qwen3.8-Max	openrouter/qwen/qwen3.8-max	2026-08-03	1,935/1,935	JSON	whole scenario	provider default; 98,304-token shared budget

Source: Article Notebook

4 Data and scenario construction

4.1 United States

The US benchmark is built from the certified PolicyEngine populace dataset (populace_us_2024) using PolicyEngine US. The dataset contains 160,858 people in 75,112 households, with household, tax-unit, SPM-unit, family, marital-unit, and person records for PolicyEngine calculations. The sampled households are filtered to keep a single-tax-unit, single-family, single-Supplemental Poverty Measure (SPM)-unit structure with at least one adult and a supported filing status. Of the 75,112 households, 63,128 (84.0%) pass the filter and form the eligible draw (this filter-pass rate is coincidentally close to the 85% zero-output

share discussed under the benchmark design; the two figures are unrelated). The 16.0% excluded by the filter include multi-tax-unit households (e.g., adult roommates), multi-family households, multi-SPM-unit households, and households whose head reports a filing status outside the supported set. These excluded compositions are exactly the kind of cases where federal/state credit allocations and benefit-unit rules become hardest, so the eligible draw is a tractable subset of the full distribution of US households. Prompts carry the nonzero promptable raw inputs across relevant entities directly, with no hand-curated summary, so the models see many of the same facts the simulator receives. Filing status is not stated in the prompt; the reference computation infers it from tax-unit role flags. Models therefore see the same household facts that drive the reference filing-status assignment, but they do not receive that assignment as a label.

4.2 Disability in the household facts

The household facts carry one disability fact, while the programs apply several. The certified microdata marks a person as disabled when they report any of the six disability-difficulty items in the Current Population Survey, and the prompt passes that flag through as is disabled: 33 of the 177 people in the benchmark households carry it. The law defines disability program by program, and policyengine-us follows it with a separate input for each determination. SSI requires its own disability criterion and no substantial gainful activity (42 U.S.C. 1382c(a)(3)). SNAP counts a member as disabled through receipt-based routes, among them receiving SSI or Social Security disability benefits and a qualifying veterans' status (7 U.S.C. 2012(j)); policyengine-us 1.755.4, behind the September 22 references, approximated the SSI route with SSI disability status, which does not require receiving SSI; 2.60.0 tests receipt (SSI, Social Security disability benefits and the veterans' statuses). Medicare's disability route before age 65 requires 24 months of entitlement to Social Security disability benefits (42 U.S.C. 426(b)), the route policyengine-us models. The tax code uses permanent and total disability for dependents (26 U.S.C. 152), retirement on disability for the credit for the elderly and disabled (26 U.S.C. 22), and incapacity for self-care for the child and dependent care credit (26 U.S.C. 21). The general flag enters none of these determinations; in the reference engine it reaches mainly state provisions, and in SNAP only the work-requirement and student rules. One of those state provisions is California's Working Disabled Program, a Medi-Cal pathway for people with disabilities who work: the program uses SSI's definition of disability (42 CFR 435.540(a)), and policyengine-us tests the general flag. No benchmark person carries any of the program-specific inputs. The certified build never sets SSI's disability criterion, and the months of Social Security disability receipt are integer-typed, which the prompt builder does not expose. The references of a person the prompt calls disabled therefore take the non-disabled path through each of these determinations unless another listed fact establishes it: listed Social Security disability income, for example, makes a person disabled for SNAP. A careful reader of is disabled could take either path, so wherever the other reading moves a reference, the output is excluded from scoring for every model (Section 4.3): SSI and SNAP outputs under SSI's disability criterion, the SNAP output whose standard utility allowance turns on whether its head is

a disabled member and on its energy subsidy's source, and one California Medicaid output, which the other reading moves once the engine's Working Disabled Program test reads SSI's definition. Separately, the prompt lists Social Security disability income but not how long it has been received, so the Medicare eligibility of the five household heads with that income, which turns on 24 months of it, is excluded whether or not the prompt calls them disabled. Where the other reading moves nothing, the output stays scored. The next scenario refresh defines the prompt's disability fact and states the program-specific determinations as separate facts (PolicyEngine/policybench#165, PolicyEngine/microcosm#876).

The current US release requests 18 scored output groups spanning federal income tax, refundable credits, payroll and self-employment tax, state and local income tax, Supplemental Nutrition Assistance Program (SNAP), Supplemental Security Income (SSI), Temporary Assistance for Needy Families (TANF), school-meal eligibility, and person-level coverage eligibility for the Special Supplemental Nutrition Program for Women, Infants, and Children (WIC), Medicaid, the Children's Health Insurance Program (CHIP), Medicare, Head Start, and Early Head Start; several of these categories span two scored groups each (federal and state refundable credits, payroll and self-employment tax, state and local income tax, free and reduced-price school meals), which is how the named categories reconcile to the 18 output groups. The source run also requested the Affordable Care Act (ACA) Premium Tax Credit (PTC), but explanation audits showed that the prompt could be misleading when households lacked plan-specific Marketplace information. We therefore preserve those raw responses but exclude PTC from the canonical scored leaderboard until the prompt contract is revised.

The output scope is intentionally narrower than the full PolicyEngine model. Table 3 summarizes the inclusion rule. The benchmark asks only for WIC eligibility, and WIC dollar values serve as impact-weight proxies for coverage flags.

Source: Article Notebook

Table 3: Output-selection rationale.

Scope decision	Rationale
Included	Direct tax, credit, benefit, health-support, and coverage outputs that a household-facing model could plausibly be asked to estimate from household facts.
Excluded	Intermediate tax bases, payroll subcomponents, and outputs whose answer turns mainly on unavailable history, restricted local market data, restricted program-administration data, or take-up assignment.

Table 3: Output-selection rationale.

Scope decision	Rationale
Binary coverage outputs	Requested as 0/1 eligibility flags and scored as classification tasks; their dollar values serve only as impact-weight proxies, and no prompt asks for them.
WIC	The benchmark asks for person-level WIC eligibility. It does not ask models to estimate a WIC dollar amount.

Source: Article Notebook

4.3 Reference-output credibility

PolicyBench treats PolicyEngine outputs as benchmark reference outputs, and no administrative record stands behind them. The reference source is nevertheless stronger than an ad hoc answer key: PolicyEngine is open source, used for household calculators and reform analysis, and externally checked in specific domains. No. 10 Downing Street's data science team adapted PolicyEngine's open-source microsimulation model for experimental policy simulation, with validation against external projections before use (Woodruff 2026; Ghenis 2026). In the US, PolicyEngine (2024) reports matching the National Bureau of Economic Research (NBER) TAXSIM-35 model (Feenberg and Coutts 1993) to the cent on the vast majority of cases for the 2021 tax year across hundreds of thousands of tax units per state, with state-specific differences documented in the integration tests. We do not restate that comparison as a single percentage because the published source states it qualitatively and gives no headline accuracy number. PolicyEngine has also signed a memorandum of understanding (Ghenis and Makarchuk 2025) with the Federal Reserve Bank of Atlanta for future validation work against its Policy Rules Database (2026). The Atlanta Fed sources are a caveat for this benchmark: they document planned collaboration and the comparison source, and no finished check of the frozen PolicyBench outputs. Taken together, these sources support using PolicyEngine as a transparent reference implementation with partial external validation, but they do not validate every benchmark output.

This does not make PolicyEngine infallible, so we treated the reference outputs as claims to be attacked. Across development and each later audit, the model-assisted review covered 9,050 rows whose legacy threshold score is below 1. That selected universe contains 9,046 exact-match misses and 4 exact hits; 2,407 additional rows with bounded score below 100 were not annotated. Reviewers adjudicated each reference-suspect flag by hand against primary sources, including statutes, regulations, and published program parameters, and against the engine's computation traces. Before the September 22 wave, reviewers had raised 35 reference-suspect

flags. Thirty-one dissolved under adjudication, with the reviewer's hypothesis failing against a primary source (stale thresholds, missed program mergers, missed federal-conformity elections, misread eligibility-income definitions) or against the engine trace. Four were genuine: two Head Start eligibility flags and one New Jersey state income tax cell reflected policyengine-us defects (an age-window bound and a filing-threshold floor the benchmarked path bypassed), and one SNAP cell rested on a scenario whose encoded inputs did not match the disability status its prompt implied. The engine defects were fixed upstream (policyengine-us #8845 and #8846, plus #8844, found in the same review but affecting no benchmark cell), and this snapshot's scored reference outputs come from the fixed engine, policyengine-us 2.60.0. The mis-encoded scenario is disclosed under Limitations pending the next scenario refresh. Table 4 summarizes the reviewed discrepancy classes and outcomes.

The audit wave run with the September 22 additions used a stronger judge, Claude Opus 5.5, and it found what the earlier waves had not. Its flags and those of later waves are settled in the adjudication record. Of the 24 judge-flagged references whose decisions this snapshot keeps, adjudication against primary sources and sandbox runs of the reference engine affirmed 6, traced 9 to inputs the prompt never lists, replaced 5 with a regenerated reference, confirmed 3 as defects in policyengine-us 1.755.4, the version behind the references then, and traced 1 to an amount a state published after the freeze. A later engine upgrade that regenerates a reference drops its decision, so earlier releases kept more of them. The judge's stated hypothesis was not always the defect: a Wisconsin flag blamed a missing retirement exclusion that the engine does apply, on a path the benchmarked output skips, and a Montana flag cited a child tax credit that was never enacted, while the reference was wrong for another reason. The judge also sees only the cases a new model joins, so its flags sample each defect and cannot bound it. Each root cause's rule was therefore implemented as a sandbox fix on the same engine version and every reference recomputed under it. Of those root causes, 8 had been fixed in policyengine-us after the references were frozen (#8839, #9162, #9301, #9313, #9318, #9363 and #9586): capital gain distributions, New York's renter cap, the CalEITC's lookup at adjusted gross income, and the SNAP rounding and SNAP child support option described below. For the September 22 references, PolicyBench applied each fix on that engine version to the scored references it moves, 15 in all, and policyengine-us 2.60.0 contains all 8 fixes. An output one of them moves that an unfixed cause also moves stays excluded. The capital gain fix as first built also extended Wisconsin's 30 percent capital gain exclusion (Wis. Stat. 71.05(6)(b)9) to the distributions. The upstream fix did not, so that part was recorded as its own defect, measured on top of the upstream fix and Wisconsin's published amounts, and the two outputs it moved were excluded (one of them also for another Wisconsin defect) until policyengine-us fixed it too (#10026); both return to scoring on 2.60.0. Every output a defect not yet fixed upstream moves by more than a dollar is excluded from scoring for every model. The exclusion record holds 5 engine-defect exclusions in 5 households across 3 root causes, and 2 of them had never been flagged (an output such a defect moves that was already excluded for an unlisted input keeps that record). policyengine-us 2.60.0 no longer has the recorded defect behind any of them. An output returns to scoring when references built on an engine that fixes its defect replace it, unless it also moves under an input the prompt does not state or its reference rests on a

further defect that engine has. A second model, GPT-6 Astra, then re-derived the sandbox fixes the first pass built from primary sources and reran their sweeps. It narrowed several fixes to the benchmark year and filers they reach, reclassified two root causes as unlisted inputs (the type of a household's survivor benefits, and whether listed mortgage interest is on the home the household occupies), and found that a California credit move came from a separate defect: the engine never looks the CalEITC up a second time at adjusted gross income, as the credit worksheet requires.

The same wave tested a failure mode no judge flagged. policyengine-us fills a parameter's values past its last encoded year by projecting it with a price index, so a 2026 reference can rest on the engine's forecast of an amount a government has published. Every reference was recomputed with each projected 2026 value held at its last encoded value, and each projected amount behind a moved output was then sourced. Where the government published the 2026 amount before the freeze, the reference takes it: Minnesota's, Michigan's and Missouri's brackets, deductions and exemptions, and Maryland's withholding allowance. policyengine-us reads the allowance only in its estimate of Maryland income tax withheld, which fills the state income tax part of the federal state and local tax deduction; the one Maryland federal output that estimate moves is excluded (reference_audit/2026-10-05). Wisconsin had published its 2026 brackets and deductions too, but every Wisconsin output they move is excluded for an engine defect, so they enter only those outputs' corrected values. Where no 2026 amount published before the freeze was found, the reference takes the last amount published: California's indexed amounts, whose 2026 factor rests on June 2026 prices; the IRS optional sales tax tables, whose 2026 edition had not been found by September 22; Idaho's zero-rate threshold; Maryland's return deduction; and SNAP's FY2027 figures, published on 2026-08-21. Illinois's exemption and Minnesota's child credit projections matched the published amounts. These corrections change parameter values and no rule, so the affected references were regenerated with the same engine version and stay scored: 23 references, most of them SNAP. With the upstream fixes, the September 22 release regenerated 26 references in 24 households, 13 of them SNAP. The reference sidecar records each regenerated value with the fix that produced it, and the fixes, the recomputation of every reference under each, and the verification reports are committed with the snapshot (reference_audit/2026-09-22).

The SNAP convention as first built also corrected the engine's SNAP arithmetic, and an independent review of the snapshot separated those corrections from the parameter holds. They are engine defects, each fixed in policyengine-us after the freeze (#9162 and #9318), so the SNAP references apply the fixes: the engine kept cents in the allotment, where 7 CFR 273.10(e)(2)(ii)(A) requires a whole-dollar allotment; it returned a minimum benefit of $23.84 a month, where 7 CFR 273.10(e)(2)(ii)(C) rounds 8 percent of the one-person maximum to the nearest dollar ($24); it floored net income, which 7 CFR 273.10(e)(1)(ii) requires a state to round to the nearest dollar at each step or to calculate by its TANF procedure, which may keep cents; and it compared income with unrounded income standards. Given the engine's household size, eligibility and floored net income, the Axiom rules engine's encoding of 7 U.S.C. 2017(a) reproduces the whole-dollar allotment of all 19 households the first SNAP convention module changed, in each month from January to September (171 of 171 monthly comparisons),

and the federal minimum benefit; it did not test New Jersey's state minimum, and it covers the allotment rounding alone, leaving untested the net income rounding the published references also apply. Recomputing SNAP with net income's cents kept (the upstream fix rounds net income to the nearest dollar) moves no scored reference by more than the dollar tolerance. One SNAP defect outlasted the references' first two engine moves: the engine granted the heat-and-eat standard utility allowance to households without an elderly or disabled member, which P.L. 119-21 section 10103 ended in July 2025, until policyengine-us fixed it (#10033). The scored output it moved was excluded, and it stays excluded on 2.60.0, which gives the record's corrected value ($3,240), because the stated facts do not decide it, as the 2026-09-29 recheck found. The prompt calls the head disabled but states none of the routes by which 7 U.S.C. 2012(j) defines a disabled member, such as receiving SSI or Social Security disability benefits (with one, the allowance would stand, at $3,576), and it does not say whether the household's $2,000 energy subsidy is a LIHEAP payment or a State-law payment, which 7 U.S.C. 2014(k)(4) treats differently. One more SNAP defect was found after the September 22 release and fixed upstream on 2026-09-24: the engine's values for the SNAP child support option carried the opposite meaning from the one its formulas read. A state may exclude legally obligated child support paid from gross income (7 CFR 273.9(c)(17)); a state that does not must deduct it from net income (273.9(d)(5)). For 2026 the engine excluded it in 37 jurisdictions that USDA's SNAP State Options Report lists as deducting it, Michigan among them, and deducted it in 9 of the 14 it lists as excluding it. Recomputed under the fix (policyengine-us #9586), one reference moves: a one-person Michigan household that pays child support stayed within the 200 percent gross limit of Michigan's broad-based categorical eligibility only because the engine excluded the payments, so the engine granted it the $24 monthly minimum; counting the payments, as Michigan does, leaves it ineligible, and the regenerated reference is $0.

PolicyBench computes each scored reference with policyengine_us.Simulation. The snapshot also records policyengine.py 6.1.2 for provenance; its certified US bundle carries an older policyengine-us, and policyengine.py does not load beside 2.60.0. On 2026-09-29 PolicyBench rebuilt the references with policyengine-us 2.15.17 in place of 1.755.4 and ported the publication conventions to the new version. The SNAP convention holds each FY2026 figure directly for October to December 2026, because the new version carries USDA's FY2027 figures as published values. policyengine-us 2.15.17 counts Maryland county income tax in its state income tax. PolicyBench's state income tax output leaves local tax out, so an adapter restores that scope, and the county tax stays in the federal state and local tax deduction. policyengine-us #9261 changed from 40 hours to 0 the default of the weekly-hours input that SNAP's work rules read, so the scenario builder also passes each prompt's stated usual weekly hours under that input's name. After the port, PolicyBench grouped every output that still moved by root cause. An investigator settled each root cause from the upstream commits, the engine code of both versions and primary legal sources with their publication dates, and an independent reviewer tried to refute each settlement. The move changes 4 scored references (Table 4). Two take law published before the freeze that policyengine-us 1.755.4 did not encode: New Jersey's child tax credit schedule for 2026 to 2028 (P.L.2026, c.26, approved June 30, 2026), and Arizona's broad-based categorical eligibility limit for SNAP, which Arizona raised from 185

to 200 percent of the poverty guideline starting with benefit month March 2026. Two follow engine corrections: policyengine-us added child support received to school-meal income (7 CFR 245.6(a)(5)(ii)), and policyengine-us #9425 corrected the rounding in New York's Empire State child credit phase-out. The investigation found, and PolicyBench excluded, 3 federal income tax outputs that depend on an unlisted input: the new version counts a listed state and local tax refund as income (policyengine-us #9422), and whether any of it is income depends on whether the refunded tax reduced federal tax in the year the household paid it (26 U.S.C. 111(a)), which the prompt does not say. The re-review of a California household's excluded SNAP output also flagged a scored output the move did not change: the head's Medicaid eligibility. The head's income is above the 138 percent limit for the adult expansion group, so only a disability pathway leads to Medi-Cal, and California's Working Disabled Program requires SSI's definition of disability, the unlisted input that already excludes the household's SNAP. policyengine-us 2.15.17 tests the general disability flag there and gives 1 under either reading; with the program's test reading SSI's definition, the head qualifies only by meeting it, and PolicyBench excluded the output. Another 2 references move by less than the dollar tolerance. PolicyBench re-reviewed the 19 excluded outputs whose values move on 2.15.17; all stay excluded, each with the value it was decided on. The reference sidecar lists every change, and reference_audit/2026-09-28 holds the recomputation of every output, the investigations and the reviews.

A review of release dashboard-data-20260930 on 2026-10-05 found three more inputs that no prompt states behind scored references; no judge had flagged the scored outputs they move (reference_audit/2026-10-05, reference_audit/2026-10-05-medicare-part-b and reference_audit/2026-10-05-payroll). The first is the state income tax in the federal state and local tax deduction, which takes the state income tax a household paid during the year, or general sales tax instead (26 U.S.C. 164(a)(3) and (b)(5)). No prompt states state income tax withheld or paid. policyengine-us fills that input with state_withheld_income_tax, a formula estimate of withholding on the person's whole federal adjusted gross income, which can differ from the household's state liability, and the references take the estimate. Read as the household paying its 2026 state income tax during the year, or as paying none, the federal income tax of five households moves, three of them scored. The second is Medicare enrollment and a Part B premium. policyengine-us treats every Medicare-eligible person as enrolled and adds the modeled Part B premium, $2,434.80 for 2026 where the Medicare Savings Program does not pay it, to medical expenses; no prompt states either. The prompt's instruction to assume program take-up supports the reference, and its rules that unlisted numbers are zero and that unlisted expenses and health coverage are not to be inferred support the other reading. Without the premium, the outputs of one household move: it itemizes, so its federal income tax moves, and Virginia's itemized deductions start from the federal ones, so its Virginia income tax moves too. The third is whether the employer deducts the employee share of a state paid-leave or disability premium. The payroll output counts mandatory employee state payroll taxes. In Minnesota, Colorado, Massachusetts and New York the law lets the employer deduct an employee share of such a premium from wages, up to a cap, but does not require it, so the employee pays from $0 to the cap at the employer's choice. The four payroll references from those states count the

cap, and the prompt does not state the employer's choice; read as the amount the law requires of the employee, each is employee federal payroll tax alone. California's, Connecticut's, New Jersey's and Pennsylvania's employee contributions are owed by the worker and withheld by the employer, and the review left those references as they were. Each audit recomputed every output on policyengine-us 2.15.17 with the publication conventions, under each reading, and each sweep's baseline reproduces all 1,928 references that release scored. Beyond the one-dollar tolerance, the sweeps move eight scored outputs, which PolicyBench excluded, and two federal income tax outputs already excluded for other reasons; they move no output that is still scored by any amount. The eight records are computed on policyengine-us 2.15.17, whose values are the references.

On 2026-10-06 PolicyBench ruled to exclude ten more outputs. Eight came from a reference adversary (reference_audit/2026-10-05-reference-adversary). It flags the scored outputs where many models, or several of the strongest, agree on an answer the reference does not give. A judge that cannot yet see how the engine derived the reference works each flagged output from primary law, and only then reconciles its answer with the derivation. The adversary flagged 61 outputs and ran 52, the other nine belonging to other audits, and it held 45 references. An independent check of its seven other verdicts confirmed four as engine defects: Arizona's 2026 standard deduction left unindexed (A.R.S. 43-1041(H)), Ohio's medical deduction counting health insurance premiums only for a Medicare-eligible filer with no employer-paid plan (R.C. 5747.01(A)(10)), Colorado's 2026 sales tax refund paid without the excess state revenues that C.R.S. 39-22-2003(2) requires, and New York's 2026 child and dependent care credit, which Tax Law 606(c-2) replaced. The check refuted two, the Medicaid eligibility of two children in a North Carolina household, whose references stand. The seventh is a question of definition. A Pennsylvania household's dependent earns $45,000 of wages and must file their own return, and the output definitions name no return, so the household's federal and state income tax can be read with or without that return; a scan for the same question found a Missouri household, and PolicyBench excluded the four outputs. The other two outputs are two Louisiana households' 2026 state income tax (reference_audit/2026-10-05-louisiana). Their references rest on a standard deduction that policyengine-us computed from published price indexes under Louisiana's formula; Louisiana published the return amount itself on 2026-09-28, after the freeze. All ten records are computed on policyengine-us 2.15.17.

On 2026-10-10 PolicyBench moved the references from policyengine-us 2.15.17 to 2.38.6, a release that fixes engine defects behind excluded outputs, and recomputed every output under the same publication conventions and adapter (reference_audit/2026-10-09-engine-upgrade). The move fixes the engine defects behind 18 outputs. Earlier releases excluded 14 of them, which return to scoring at the new version's values; the other four, ruled on 2026-10-06 while still scored, stay scored at those values. Of the 18, 16 land within $1 of the corrected values their exclusion records carry. The other two are held instead to the values the audit's fix modules give on policyengine-us 2.37.2, an engine that still has the defects; each lands within $1 of its own. The fixed outputs take the upstream fixes for the IRA deduction's compensation limit and a dependent's contributions (policyengine-us #10032, #9801; three outputs), the IRA deduction's active-participant phase-out and California's itemized deduction conformity (policyengine-us #10032,

#10034), New Jersey's worker unemployment and workforce contributions (policyengine-us #10031), Arizona's standard deduction indexing (policyengine-us #9928), California's itemized deduction conformity (policyengine-us #10034), Ohio's medical deduction for health insurance premiums (policyengine-us #10020), estate income in gross income (policyengine-us #10027, #9633; two outputs), Colorado's 2026 sales tax refund (policyengine-us #9946), the IRA deduction's active-participant phase-out (policyengine-us #10032; four outputs), New York's 2026 child and dependent care credit (policyengine-us #9948) and the IRA deduction's active-participant phase-out and estate income in gross income (policyengine-us #10032, #10027; two outputs). The move changes one scored reference beyond the dollar tolerance and two within it, and removes two outputs from scoring. The changed scored reference follows Idaho's $10 permanent building fund tax (Idaho Code 63-3082), which policyengine-us now counts in state income tax (policyengine-us #9810). The two outputs that leave scoring depend on an input no prompt states: county of residence (Indiana county income tax).

On 2026-10-13 PolicyBench moved the references again, from policyengine-us 2.38.6 to 2.41.0, the first release with every fix the previous move was waiting on. With the same publication conventions and adapter, it recomputed every output (reference_audit/2026-10-12-engine-upgrade). The move fixes the engine defects behind nine outputs. Earlier releases excluded them, and they return to scoring at the new version's values. Of the nine, six land within $1 of the corrected values their exclusion records carry. The other three are held instead to the values the audit's fix modules give on policyengine-us 2.38.6, an engine that still has the defects; each lands within $1 of its own. The reference sidecar records each target with the upstream fix it names (regenerated_exclusions). The restored outputs take the upstream fixes for Idaho's subtraction for health insurance premiums (policyengine-us #10023; two outputs), Wisconsin's retirement income exclusion before refundable credits and Wisconsin's capital gain subtraction for distributions (policyengine-us #10026; two outputs), the IRA deduction's active-participant phase-out, California's itemized deduction conformity and California's educator expense addback (policyengine-us #10032, #10034, #10052) and elective deferrals in the earned income behind the EITC and the refundable child tax credit (policyengine-us #10030; four outputs). The move changes no scored reference, by any amount, and removes no output from scoring. PolicyBench re-reviewed the 12 excluded outputs whose values move on 2.41.0; all stay excluded, each with the value it was decided on.

On 2026-10-23 PolicyBench moved the references once more, from policyengine-us 2.41.0 to 2.60.0, and recomputed every output under the same publication conventions and adapter (reference_audit/2026-10-22-engine-upgrade). policyengine-us 2.60.0 was the newest release when PolicyBench began sweeping the references on 2026-10-23 (uploaded 2026-10-22 at 22:47 UTC). policyengine-us 2.78.0, the newest release when PolicyBench checked PyPI on 2026-10-29 at 16:52 UTC, gives the same value as 2.60.0 for all 1,935 scored outputs under the same conventions and adapter, and moves one of the excluded outputs, which keep the values they were decided on. No excluded output returns to scoring. The move changes four scored references beyond the dollar tolerance and none within it, and removes no output from scoring. The changed scored references follow the free school breakfast that Pennsylvania (Act 33 of 2023) and Arkansas (Act 123 of 2025) give students at covered schools (policyengine-us #10238)

and California's modified categorical eligibility for CalFresh, which has no net income test even for a household with an elderly or disabled member (policyengine-us #10036). policyengine-us models both breakfast laws for a student at a covered school without their conditions: Pennsylvania reimburses schools to the extent funds are available, and Arkansas serves the breakfast on request. PolicyBench re-reviewed the 16 excluded outputs whose values on 2.60.0 differ from the values they were decided on; all stay excluded. The 49 excluded outputs keep the values they were decided on (29 computed with policyengine-us 1.755.4, 18 with 2.15.17, 2 with 2.38.6).

Federal income tax references apply the section 1 rate schedule to taxable income, to the cent, at every income. The Form 1040 instructions send filers with taxable income under $100,000 to the IRS Tax Table, whose looked-up amounts can differ from the schedule by a few dollars. PolicyBench keeps the schedule: an answer that follows the Tax Table can miss the $1 tolerance on those outputs (reference_audit/2026-10-10-tax-table).

The audit tooling itself required hardening, which we report because it changes how much the annotations can be trusted. The derivation narratives displayed alongside reference values are written by a model from the engine's computation trace for each output. After earlier narratives were found to misattribute eligibility pathways, the writer also receives the recorded basis of each reference change, an engine upgrade's build refuses a rewritten narrative that omits the new reference amount, and narratives that reviewers found wrong were replaced by hand corrections (hand_corrected_narratives in each engine upgrade's audit). No check reads every mechanism a narrative states, so a narrative can still misdescribe how the engine reaches a value it states correctly. The failure-audit judge reads these grounded narratives together with engine facts and must return a per-model diagnosis naming the specific rule or computation the model missed. It is barred from re-adjudicating the reference in prose: hedged or reference-adjudicating verdicts are mechanically rejected and re-judged, and genuine reference doubt is routed to a structured flag that triggers the manual adjudication described above.

After freezing the snapshot and completing response-contract repairs, we annotated the 9,050 rows whose legacy threshold score is below 1 and their scenario-output cases. Table 5 reports the final classifications for that legacy-threshold audit universe. It contains 9,046 of the snapshot's 9,046 exact-match misses and 4 exact hits. The 2,407 other rows with bounded score below 100 fall outside the selection and have no audit annotation. Each annotated row on a scored output resolves to one of two final classes: llm_error (a substantive miscalculation, with a per-model diagnosis naming the rule or computation the model missed) or parse_contract_failure (a missing or unparsable answer, classified deterministically). Three further classes describe outputs the snapshot does not score, and each such output is removed from scoring for every model, so no model gains or loses from it. prompt_ambiguity marks 42 outputs whose reference depends on an input the prompt never lists, so a careful reader could take the stated facts either way; for each, the reference was recomputed under the other reading with the engine version that produced it, and it moved (the California Medicaid output once the engine's Working Disabled Program test read SSI's definition, as the program does). reference_engine_defect

marks the 5 outputs where that engine misapplies the law on stated facts. policyengine-us 2.60.0 computes all five at their corrected values; they stay excluded because each also moves under an input the prompt does not state. reference_later_law marks the two Louisiana outputs whose reference rests on an amount the state published after the freeze. Together they remove 49 outputs in 37 households, and of the 1,949 annotated rows on those outputs, 1,911 carry one of these classes as a description of an unscored output, and the rest are answers that never parsed. A scored reference also follows from law published before the references were frozen on 2026-07-03. USDA published the FY2027 SNAP cost-of-living figures on 2026-08-21, after the freeze and after most rows were answered, so grading any model on them would score it against law it could not have seen; SNAP's October to December months therefore hold the FY2026 figures, and the publication conventions described above apply the same rule to every projected amount. Every model is therefore scored on 1,935 of its 1,984 requested outputs. The exclusion record (reference_exclusions.json) travels with the frozen run and the manifest pins it. No scored row remains classified as a prompt ambiguity, unresolved reference issue, or needs-review item, and zero annotated rows carry a standing reference-suspect flag. The audit pipeline is model-assisted, with developer adjudication of each reference-suspect or prompt-ambiguity flag and no independent external validation; this snapshot records 63 cases of developer adjudication, kept beside the annotations with the judge's verdict and the reasoning. Three judge models produced the verdicts, and all are board rows: GPT-5.6 Sol, called through the Codex CLI, judged the cases whose verdicts predate the September 2026 additions; Claude Opus 5, called through the Claude Code CLI, judged the 92 cases the September 5 additions joined that no later judge re-judged; and Claude Opus 5.5 judged the cases the September 22, September 29, September 30, October 9, October 13, October 14 and October 27 additions joined or a reference revision changed, its own row's cases included, and five cases whose earlier verdicts an audit of the judges' transcripts could not show came from a judge that saw only its prompt (325 of 681 cases in all). Each verdict is recorded in a per-case provenance sidecar in the audit tree. Judge verdicts classify misses after scoring and change no score, and every reference flag a judge raised is settled by the developer adjudication above. The unlisted inputs behind the unlisted-input exclusions are the SSI disability criterion (meets_ssi_disability_criteria), false for every person in the certified June 2026 microdata build; months of SSDI receipt (months_receiving_social_security_disability), never carried by the data and integer-typed, which the prompt builder does not expose; weekly hours worked (weekly_hours_worked_before_lsr), which policyengine-us 1.755.4 defaulted to 40 where the prompt treats unlisted numbers as zero; whether survivor benefits other than Social Security are a pension or annuity, which the prompt does not say and which decides both their taxability and whether they count toward Wisconsin's homestead credit income; who paid for the coverage behind listed employment disability benefits, which decides whether they are taxable; whether listed home mortgage interest is on the home the household occupies, which decides whether SNAP counts it as a shelter cost; whether an adult tax dependent is the claiming filers' child, which decides whose income counts in that dependent's Medicaid household; whether the tax behind a listed state and local tax refund reduced federal tax in the year the household paid it, which decides how much of the refund is income; the state income tax a household withheld or paid during 2026, which the federal state and local tax deduction takes as its state income

tax part and which policyengine-us estimates from federal adjusted gross income; Medicare enrollment and a Part B premium, which policyengine-us assumes for every Medicare-eligible person and counts as a medical expense; whether the employer deducts the employee share of a state paid-leave or disability premium that the law lets it deduct but does not require, which decides whether that share is a mandatory employee state payroll tax; and whether listed interest comes from a Massachusetts bank deposit, which moves only an output already recorded as an engine-defect exclusion. One definition was also open: the output definition of federal income tax before refundable credits does not say whether it includes the net investment income tax, which the reference adds and Form 1040 reports after its line for tax after nonrefundable credits, so the outputs that tax moves are excluded as well. The prompt lists a general is disabled fact and never a program-specific determination (Section 4.2). Every exclusion but the California Medicaid output and the 2026-10-06 records was derived mechanically, without regard to which models missed: each is an output that a sweep of every reference, under an alternative reading or an engine defect's sandbox fix, moved on policyengine-us 1.755.4 or, for the state and local tax refund reading and the readings of the 2026-10-05 review, on 2.15.17; the outputs the 2026-10-10 upgrade excluded are ones policyengine-us 2.38.6 itself moved, onto an input no prompt states. The 2026-10-06 records are the exception by design. The reference adversary selects the outputs where models agree against the reference, and an independent check against primary law then decided each; the two Louisiana records came from a sweep of that state's 2026 standard deduction. On 1.755.4, PolicyBench excluded every output an alternative reading of the September 22 audit moved and every output a defect not fixed upstream moved by more than a dollar, which is why the set includes three SNAP outputs that move through the elderly-or-disabled SNAP rules (in two of those households SSI itself stays at zero; in the third an aged member already receives SSI and the disabled member would add a second benefit). On 2.15.17, PolicyBench re-ran four of those sweeps (the IRA deduction limit fix, the net investment income tax definition, and the readings for mortgage residence and 40 unlisted weekly hours) and ran a new one for the state and local tax refund reading. Beyond the one-dollar exact-match tolerance, the new sweep moves three previously scored federal income tax outputs, and PolicyBench excluded them. Set against the same 2.15.17 calculation without its fix or reading, no sweep moves a scored output by more than the tolerance, and three scored outputs move by less. PolicyBench did not re-run the other sweeps on 2.15.17. On 2.15.17, the California Medicaid output stays at 1 under either reading of SSI's disability criterion, because the engine's Working Disabled Program test reads the general flag in its place. PolicyBench found that output when it re-reviewed the household's excluded SNAP output, and excluded it because the program itself requires SSI's definition of disability.

Source: Article Notebook

Table 4: Discrepancy review across the audit waves.

Discrepancy class	Review outcome
US federal credits	Reviewed cases where explanations omitted EITC or refundable CTC; each discrepancy was a model credit-treatment error, and PolicyEngine's credit values held up.
US health coverage (Medicaid/CHIP)	Reviewed threshold-cluster suspects; all dissolved against 42 CFR 435 pathway rules (aged/disabled income tests, tax-dependent household composition, CHIP's exclusion of Medicaid-eligible children) confirmed in engine traces.
US SNAP, SSI, and school meals	Reviewed false-zero suspects; references confirmed against categorical-eligibility, aged/ blind/disabled, and universal-free-meal rules. One SNAP cell traced to a scenario encoding that omits the disability its prompt implies (disclosed under Limitations).
US state and local income tax	Reviewed state-rule and bracket suspects; most applied wrong schedules or missed conformity elections and forgiveness programs. One genuine defect: New Jersey's filing-threshold floor was bypassed on the benchmarked path (fixed, policyengine-us #8845).
US Head Start eligibility	Two flags reflected a genuine age-window defect (fixed, policyengine-us #8846); references regenerated with the fix.
US payroll and overtime	Reviewed during development; upstream data fixes and prompt clarifications were applied before the frozen snapshot.

Table 4: Discrepancy review across the audit waves.

Discrepancy class	Review outcome
US payroll tax (October 2026 review)	No judge flag: a review of the state part of every payroll reference classified each state program by who owes the contribution. Minnesota's and Colorado's paid leave laws, Massachusetts's paid family and medical leave law and New York's paid family leave and disability law let the employer deduct an employee share of the premium up to a cap but do not require it; the four references from those states, which count the cap, are excluded as unlisted inputs. California's, Connecticut's, New Jersey's and Pennsylvania's employee contributions are owed by the worker and withheld by the employer, and the review left those references as they were.
US federal income tax (September 2026 wave)	Engine defects confirmed: the traditional IRA deduction ignores the compensation limit and the active-participant phase-out (IRC 219(b), 219(g)); estate income never enters gross income; elective deferrals count as EITC and refundable-CTC earned income. Affected outputs excluded. Capital gain distributions reported without Schedule D never reached AGI, a defect fixed upstream after the freeze; the references it alone moves are regenerated with the fix. The taxability of employment disability benefits and whether the output includes the net investment income tax excluded as unlisted inputs. The educator-expense flag dissolved against the $350 2026 cap.

Table 4: Discrepancy review across the audit waves.

Discrepancy class	Review outcome
US state income tax and credits (September 2026 wave)	Engine defects confirmed in New Jersey worker contributions, Wisconsin's elected retirement exclusion, Wisconsin's capital gain exclusion (which leaves out capital gain distributions), Idaho's health-premium subtraction, California itemized-deduction conformity, and Massachusetts dividends taxed without the short-term capital loss offset; affected outputs excluded. The CalEITC's missing second lookup at adjusted gross income and New York's renter cap were fixed upstream after the freeze; the affected references are regenerated with the fixes. Wisconsin homestead income turned on the unlisted type of survivor benefits (excluded as an unlisted input). The New Jersey EITC, New York real property tax credit and Idaho child tax credit flags dissolved against the statute (Idaho's credit expired after 2025).

Table 4: Discrepancy review across the audit waves.

Discrepancy class	Review outcome
US SNAP, SSI and CHIP (September 2026 wave)	Whether listed mortgage interest is on the occupied home, unlisted hours worked, survivor-benefit taxability, and whether an adult tax dependent is the filers' child (three Medicaid outputs) excluded as unlisted inputs. SNAP October-December references regenerated with the FY2026 figures, the last published before the freeze. SNAP references regenerated for every month of 2026 with the engine's SNAP rounding defects fixed as policyengine-us fixed them after the freeze (cents in the allotment, an unrounded minimum benefit, floored net income, unrounded income standards). One Michigan SNAP reference regenerated with the fix for the SNAP child support option, whose engine values carried the opposite meaning (Michigan deducts child support from net income, and the engine excluded it from gross income; fixed upstream in policyengine-us #9586 on 2026-09-24). The heat-and-eat utility allowance granted without an elderly or disabled member, which P.L. 119-21 ended, excluded as an engine defect. The CHIP flag dissolved: employer coverage bars CHIP.
US projected 2026 parameters (September 2026 wave)	No judge flag: a recomputation with every projected 2026 value held found references resting on price-index projections. Wisconsin, Minnesota, Michigan and Missouri amounts and Maryland's withholding allowance had been published before the freeze; California's indexed amounts, the IRS sales tax tables, Idaho's threshold and Maryland's return deduction had not. Scored references regenerated with the published or last-published amounts; Wisconsin's reach only the corrected values of Wisconsin outputs excluded for engine defects.

Table 4: Discrepancy review across the audit waves.

Discrepancy class	Review outcome
US references on policyengine-us 2.15.17 (September 2026 upgrade)	Once PolicyBench had ported the conventions, an investigator traced every output that still moved between policyengine-us 1.755.4 and 2.15.17 to its upstream change and settled it against primary sources and their publication dates. An independent reviewer tried to refute each settlement. The upgrade changes 4 scored references, for law published before the freeze that 1.755.4 did not encode or misapplied: New Jersey's 2026-2028 child tax credit schedule (P.L.2026, c.26), Arizona's 200 percent SNAP categorical eligibility limit from March 2026, child support received counted as income for school meals (7 CFR 245.6(a)(5)(ii)), and New York's Empire State child credit phase-out rounding. The investigation found, and PolicyBench excluded, 3 federal income tax outputs that depend on an unlisted input: 2.15.17 counts a listed state and local tax refund as income, which is right only if the refunded tax reduced federal tax in the prior year (26 U.S.C. 111(a)). PolicyBench re-reviewed the 19 excluded outputs that move; all stay excluded. The re-review of a California household's excluded SNAP output flagged its head's Medicaid eligibility, which turns on the same unlisted input, SSI's disability criterion, through the Working Disabled Program; PolicyBench excluded it.

Table 4: Discrepancy review across the audit waves.

Discrepancy class	Review outcome
US references on policyengine-us 2.38.6 (October 10 upgrade)	The move from policyengine-us 2.15.17 fixes the engine defects behind 18 outputs (14 that earlier releases excluded return to scoring at the new version's values; the 4 ruled on 2026-10-06 while still scored stay scored). The move changes one scored reference beyond the dollar tolerance and two within it, and removes two outputs from scoring. PolicyBench re-reviewed the 17 excluded outputs whose values move; all stay excluded. The fixed outputs take the upstream fixes for the IRA deduction's compensation limit and a dependent's contributions (policyengine-us #10032, #9801; three outputs), the IRA deduction's active-participant phase-out and California's itemized deduction conformity (policyengine-us #10032, #10034), New Jersey's worker unemployment and workforce contributions (policyengine-us #10031), Arizona's standard deduction indexing (policyengine-us #9928), California's itemized deduction conformity (policyengine-us #10034), Ohio's medical deduction for health insurance premiums (policyengine-us #10020), estate income in gross income (policyengine-us #10027, #9633; two outputs), Colorado's 2026 sales tax refund (policyengine-us #9946), the IRA deduction's active-participant phase-out (policyengine-us #10032; four outputs), New York's 2026 child and dependent care credit (policyengine-us #9948) and the IRA deduction's active-participant phase-out and estate income in gross income (policyengine-us #10032, #10027; two outputs). The changed scored reference follows Idaho's $10 permanent building fund tax (Idaho Code 63-3082), which policyengine-us now counts in state income tax (policyengine-us #9810). The two outputs that leave scoring depend on an input no prompt states: county of residence (Indiana county income tax).

Table 4: Discrepancy review across the audit waves.

Discrepancy class	Review outcome
US references on policyengine-us 2.41.0 (October 13 upgrade)	The move from policyengine-us 2.38.6 fixes the engine defects behind 9 outputs (earlier releases excluded them, and they return to scoring at the new version's values). The move changes no scored reference, by any amount, and removes no output from scoring. PolicyBench re-reviewed the 12 excluded outputs whose values move; all stay excluded. The restored outputs take the upstream fixes for Idaho's subtraction for health insurance premiums (policyengine-us #10023; two outputs), Wisconsin's retirement income exclusion before refundable credits and Wisconsin's capital gain subtraction for distributions (policyengine-us #10026; two outputs), the IRA deduction's active-participant phase-out, California's itemized deduction conformity and California's educator expense addback (policyengine-us #10032, #10034, #10052) and elective deferrals in the earned income behind the EITC and the refundable child tax credit (policyengine-us #10030; four outputs).
US references on policyengine-us 2.60.0 (October 23 upgrade)	The move from policyengine-us 2.41.0 returns no excluded output to scoring. The move changes four scored references beyond the dollar tolerance and none within it, and removes no output from scoring. PolicyBench re-reviewed the 16 excluded outputs whose values move; all stay excluded. The changed scored references follow the free school breakfast that Pennsylvania (Act 33 of 2023) and Arkansas (Act 123 of 2025) give students at covered schools (policyengine-us #10238) and California's modified categorical eligibility for CalFresh, which has no net income test even for a household with an elderly or disabled member (policyengine-us #10036).

Table 4: Discrepancy review across the audit waves.

Discrepancy class	Review outcome
US federal and Virginia income tax (October 2026 review)	No judge flag: a review of release dashboard-data-20260930 swept every reference on policyengine-us 2.15.17 under readings of two inputs no prompt states. The state income tax part of the federal state and local tax deduction is policyengine-us's formula estimate of state income tax withheld on federal adjusted gross income; read as the household paying its 2026 liability during the year, or paying none, three scored federal income tax outputs move, and PolicyBench excluded them; the readings also move two federal outputs already excluded. policyengine-us treats every Medicare-eligible person as enrolled and counts a modeled Part B premium as a medical expense; without it, one household's federal and Virginia income tax move, and both are excluded.

Source: Article Notebook

Table 5: Annotation outcomes for the legacy-threshold audit universe in the frozen snapshot.

Country	Legacy-threshold rows audited	LLM response errors	Parse-contract failures	Unresolved prompt/ reference rows	Missing row annotations	Missing case notes
US	9050	8381	669	0	0	0
Total	9050	8381	669	0	0	0

Source: Article Notebook

5 Results

5.1 United States leaderboard

The US leaderboard for the frozen manuscript snapshot is shown in Table 6. The top three models in that snapshot are GPT-6 Sol (95.2 exact, 95% CI 93.4–96.7; 95.7 within-1%), Claude Opus 5.5 (94.1 exact, 95% CI 92.1–95.8; 95.5 within-1%), and GPT-6.1 Terra (93.1 exact, 95% CI 91.1–94.9; 93.9 within-1%).

Source: Article Notebook

Table 6: Top US benchmark models in the frozen manuscript snapshot, ranked by the headline exact-match rate with the within-1% near-miss companion alongside. Exact 95% CIs and rank ranges are household-resampling bootstrap intervals (see Uncertainty).

	Model	Exact %	Exact 95% CI	Exact rank	Within 1% %	Within 10%	Bounded score
0	GPT-6 Sol	95.2	93.4–96.7	1–2	95.7	97.2	98.1
1	Claude Opus 5.5	94.1	92.1–95.8	1–4	95.5	97.1	97.9
2	GPT-6.1 Terra	93.1	91.1–94.9	2–6	93.9	96.0	97.2
3	GPT-5.6 Sol	93.1	90.5–95.3	2–8	93.9	95.7	97.0
4	GPT-6 Astra	91.7	89.3–94.0	3–11	92.3	94.3	96.9

Source: Article Notebook

5.2 Uncertainty

Each leaderboard score is a mean over a sample of 100 households, so the rankings carry sampling uncertainty. We quantify it with a household-level bootstrap: the 100 households are resampled with replacement 10,000 times, and each model's population-weighted exact-match rate is recomputed on every resample. The resampling unit is the household, carried with its full output vector, so within-household correlations across outputs are preserved. The same resampled set is used for every model in each replicate (a paired bootstrap), so the rank ranges in Table 6 reflect genuine ordering uncertainty and carry no independent per-model noise.

Sampling uncertainty is substantial at this sample size. In the frozen snapshot, 7 of the other 50 models have an exact-match 95% interval that overlaps the leader's. Small differences in the leaderboard ordering should therefore be read as ties, and the rank-range column makes the set of plausible positions explicit. These intervals are a manuscript artifact: the public leaderboard and app report point estimates only, because the benchmark sample is the dominant source of uncertainty here, ahead of run-to-run model variation.

5.3 Cost and latency

Accuracy is one axis; the price and speed of a no-tools answer are another. Table 7 reports each model's estimated cost to compute one household's full output vector and its median per-household latency, next to the headline exact rate. Per-household cost spans $0.001 (Claude Haiku 5.5) to $0.541 (Claude Fable 5), and median per-household latency 3 s (Gemini 3.5 Flash-Lite) to 1023 s (Kimi K3). Neither dimension cleanly tracks accuracy: the top-scoring model, GPT-6 Sol at 95.2% exact, sits mid-pack at $0.027 per household and 23 s.

Source: Article Notebook

Table 7: Estimated cost and median latency per household for the frozen US snapshot, alongside the headline exact-match rate. Each row uses its recorded per-call cost: reconstructed from token counts at the list price configured at request time, or the provider-reported charge where no reconstruction was available. A list-price override applies to requests made after it is set and leaves recorded costs unchanged. Claude Fable 5's cost and latency come from its original run's per-call rows, which its board rows do not carry. Latency is the median across households of each household's summed request time. A cost dash marks a provider-reported charge of zero on every call (GLM-5.3-Flash (preview)). A latency dash marks rows that record no request time (GPT-5.5, through OpenAI's batch API) or a median household answered from the harness's response cache without a provider call (GLM-5.2, Kimi K2.6, MiniMax M3, and Qwen3.7-max; see text).

Model	Exact (%)	Cost / household	Latency (median)
GPT-6 Sol	95.2	$0.027	23 s
Claude Opus 5.5	94.1	$0.067	24 s
GPT-6.1 Terra	93.1	$0.013	22 s
GPT-5.6 Sol	93.1	$0.090	36 s
GPT-6 Astra	91.7	$0.126	32 s
Claude Sonnet 5.5	91.7	$0.034	20 s
GPT-6 Luna	91.5	$0.002	29 s
Claude Fable 5.1	90.8	$0.257	49 s
Kimi K3	90.5	$0.470	1023 s
GPT-6.1 Sol	90.2	$0.023	34 s
Muse Wonder	90.0	$0.079	26 s
GPT-5.6 Luna	88.7	$0.019	14 s
Gemini 4 Argon	88.6	$0.028	22 s
GLM-5.3-Flash (preview)	88.6	—	242 s
GPT-5.5	88.0	$0.262	—
Inkling	87.7	$0.091	250 s
Grok 4.7	87.6	$0.260	547 s
GPT-5.6 Terra	87.6	$0.039	19 s

Model	Exact (%)	Cost / household	Latency (median)
Muse Spark 1.3	87.2	$0.021	41 s
DeepSeek V4.1 Flash	86.3	$0.024	85 s
Grok 4.6	85.8	$0.087	219 s
Gemini 3.8 Flash	85.6	$0.068	56 s
Grok 4.5	84.7	$0.073	112 s
Claude Opus 5	83.6	$0.067	16 s
Claude Fable 5	83.3	$0.541	98 s
DeepSeek V4 Pro 0813	83.0	$0.199	808 s
DeepSeek V4 Flash 0731	82.9	$0.003	195 s
Gemini 3.7 Flash	82.7	$0.014	9 s
Gemini 3.6 Flash	82.6	$0.055	26 s
Gemini 3.5 Flash-Lite	81.6	$0.003	3 s
Gemini 3.1 Pro Preview	81.2	$0.085	45 s
GLM-5.3	81.1	$0.169	134 s
Gemini 3 Flash Preview	80.6	$0.048	58 s
Claude Opus 4.7	80.5	$0.293	53 s
Grok 4.3	80.3	$0.015	22 s
Claude Sonnet 4.6	79.9	$0.186	114 s
Grok Build 0.1	79.9	$0.045	135 s
Gemini 3.5 Flash	79.8	$0.056	24 s
Gemini 3.1 Flash Lite Preview	79.3	$0.002	3 s
DeepSeek V4 Pro	79.3	$0.011	170 s
Claude Haiku 5.5	79.2	$0.001	5 s
Qwen3.7-max	76.9	$0.139	—
GLM-5.2	76.5	$0.082	—
Claude Opus 4.8	75.8	$0.255	55 s
MiniMax M3	75.4	$0.007	—
Qwen3.8-Max	74.7	$0.224	861 s
Claude Haiku 4.5	74.5	$0.055	48 s
GPT-5.4 mini	73.3	$0.006	5 s
Claude Sonnet 5	72.4	$0.172	64 s
Kimi K2.6	66.9	$0.391	—
GPT-5.4 nano	65.3	$0.002	6 s

Source: Article Notebook

Cost preserves each call's recorded total: a request-time reconstruction from token counts where one was available, and the provider-reported charge otherwise. Configured list-price overrides do not reprice recorded calls. Claude Fable 5's board rows carry no usage or timing fields, because they were rebuilt from release dashboard-data-20260701 without those columns; its cost, tokens and latency come from the per-call rows of its original run, which that release

publishes. Latency is wall-clock request time as observed by the harness, summed within each household and taken as the median across the 100 households. The timer spans one request as the provider handles it, including anything that happens inside that call. When the harness retries a failed request, only the attempt that succeeded is timed, without the failed attempts or the wait between them. The median is reported in preference to the mean, which a few slow calls inflate. Latency is not comparable across serving regimes. A model served one or three outputs per request (Table 2) sends a household's calls one after another, and its household time is their sum. GPT-5.5 ran through OpenAI's batch API, which records no per-call request time, so it shows a dash.

The harness also caches provider responses on disk, keyed on the request and the prompt contract version. A request identical to an earlier one is answered from the cache in hundredths of a second, and the timer records that as if it were a call. A replayed response's cost is computed from that response when its row is written: its token counts at the list prices then in effect (PolicyBench's own price for the model where it sets one, otherwise litellm's price table), or the provider's charge embedded in it where no reconstruction exists. A household's cost therefore counts each of its responses once, whether or not the board run paid for it. GLM-5.2, Kimi K2.6, MiniMax M3, and Qwen3.7-max were finished over several passes, each of which replayed the earlier passes' successful responses. Each of them records a replay on 53 to 93 of its 100 households, so its median falls under a second and it shows a dash as well. In 11 other models, 29 households sum to under a second. The audit traces 27 of them to an earlier request on the same household, such as an onboarding smoke or a restarted chunk; the other two come from runs that predate per-call timestamps and left no cache or ledger behind. Dropping them would raise no model's median by more than 2.5 seconds. A further 33 households replay their first recorded call and then reach the provider, so their times leave out that first call. The benchmark card summarizes the replays, and the repository's latency replay audit lists each household. Both cost and latency are properties of this run's providers, prompts, and serving configurations at evaluation time, and they can change for the same model.

5.4 Simple baselines

Simple baselines help interpret score levels in a zero-heavy benchmark. Table 8 reports an always-zero response and a median-reference-by-output response on the same frozen household sample, scored with the household-impact weighting. These are not model competitors; they show how much of each metric can be earned without household-specific policy calculation. The always-zero baseline scores 69.6% on the headline exact-match rate — below the 85% of outputs that are exact zeros because the weighting concentrates mass in high-dollar output groups that are rarely zero (Section 3.1). The top model clears the baseline by 25.5 points, and the within-1% companion shows the same separation with near-miss tolerance.

Source: Article Notebook

Table 8: Simple baseline scores on the frozen manuscript snapshot, household-impact-weighted. The always-zero exact rate sits below the unweighted zero share because the weighting concentrates on high-dollar output groups that are rarely zero.

	Baseline	Exact %	Within 1% %	Bounded score
0	Always zero	69.6	69.6	69.6
1	Median reference by output	63.6	63.6	66.5

Source: Article Notebook

5.5 Sensitivity to benchmark view

The primary leaderboard gives each household equal weight. Table 9 reports several alternative views, including the older equal-output-group score. The top model is stable across several views, but lower ranks move across sensitivity views. Scores are therefore best read with the weighting choice visible.

Source: Article Notebook

Table 9: Leaderboard sensitivity to alternative scoring views.

	View	Rank 1	Rank 2	Rank 3
0	Exact-match headline	GPT-6 Sol (95.2)	Claude Opus 5.5 (94.1)	GPT-6.1 Terra (93.1)
1	Within-1% companion	GPT-6 Sol (95.7)	Claude Opus 5.5 (95.5)	GPT-5.6 Sol (93.9)
2	Bounded score	GPT-6 Sol (98.1)	Claude Opus 5.5 (97.9)	Muse Wonder (97.5)
3	Equal-output-group score	Claude Opus 5.5 (98.1)	GPT-6 Sol (98.0)	GPT-6 Astra (97.9)
4	Amount outputs only	Claude Opus 5.5 (97.8)	GPT-6 Astra (97.5)	GPT-6.1 Sol (97.5)
5	Positive reference cases only	GPT-6 Astra (93.1)	GPT-6.1 Sol (90.7)	Claude Opus 5.5 (90.5)
6	Zero reference cases only	Claude Opus 5.5 (99.5)	GPT-6 Sol (99.5)	GPT-6.1 Terra (99.2)

Source: Article Notebook

The bounded score is a global-output-weighted average of row scores. Amount outputs use max(0, 1 − |pred − ref| / |ref|) when the reference is nonzero and 1{pred = 0} when the reference is zero. Binary outputs use exact 0/1 matching. The exact-match rate uses the same weighting and aggregation but replaces the amount row score with the headline indicator: 1{|pred − ref| 1} (one currency unit). Each output group's default household-impact weight is computed from a full weighting population, the certified PolicyEngine US populace dataset; the 100 benchmark households do not enter it. For each source household, the share is |ref_ij| / max(|household_net_income_i|, Σ_k |ref_ik|); shares are averaged with calibrated household weights and renormalized so output weights sum to one. The max(...) denominator anchors

per-household shares to net income when net income is the dominant flow and falls back to the gross tax-benefit flow only when programs cancel each other out, so a $1 benefit to a high-earner contributes essentially zero weight and per-household shares never exceed one. Booleans carry weight through PolicyEngine's paired per-capita value (for example medicaid_value), so eligibility calls are graded as accuracy but weighted by the dollar stake. Table 10 compares the bounded-score ranking against two opt-in alternatives shown for transparency: equal weighting (every output the same) and budget-weighted (each output's share of total absolute reference dollars in the same full weighting population).

Source: Article Notebook

Table 10: Top country ranks under the bounded secondary metric across three weightings (household, equal, budget).

	Weighting	Rank 1	Rank 2	Rank 3
0	Household-weighted (default)	GPT-6 Sol (98.1)	Claude Opus 5.5 (97.9)	Muse Wonder (97.5)
1	Equal weights	Claude Opus 5.5 (98.1)	GPT-6 Sol (97.9)	GPT-6 Astra (97.7)
2	Budget-weighted	GPT-6 Sol (97.9)	Claude Opus 5.5 (97.3)	Muse Wonder (97.2)

Source: Article Notebook

5.6 Hardest benchmark targets

The lowest-scoring US variables are multi-step tax quantities and sparse positive-dollar benefits. As shown in Table 11, state income tax before refundable credits, federal income tax before refundable credits, state refundable credits, SNAP, and federal refundable credits are the hardest US outputs by bounded score in the frozen manuscript snapshot.

Source: Article Notebook

Table 11: Hardest US variables by bounded score (the secondary tracking metric, used here because it orders the hardest variables differently from the hit rates and separates them more finely; on their positive-reference cases the hit rates collapse toward single digits — see the zero-versus-positive table).

	Variable	Score	Exact	Within 1%	Within 10%
15	state_income_tax_before_refundable_credits	83.1	60.7	63.0	73.9
0	federal_income_tax_before_refundable_credits	83.6	61.5	64.1	72.8
16	state_refundable_credits	85.8	82.1	82.2	83.0
13	snap	88.9	83.5	84.8	87.4
1	federal_refundable_credits	91.3	85.7	86.9	89.5

Source: Article Notebook

5.7 Zero and positive cases

Overall hit rates can overstate performance on sparse programs because correct zeros are common. Table 12 therefore reports within-10% performance for all cases, positive-reference cases, and zero-reference cases. The gap is large for several outputs: models often identify that a program or tax does not apply, but miss the amount when the reference value is positive. Figure 1 plots the same split by output group.

Source: Article Notebook

Table 12: US within-10% performance by zero versus positive reference cases.

	Variable	All cases	Positive cases	Zero cases
0	federal_income_tax_before_refundable_credits	73.6	46.3	93.9
1	state_income_tax_before_refundable_credits	74.7	48.2	91.2
2	state_refundable_credits	84.0	17.7	98.6
3	snap	87.8	34.9	97.1
4	federal_refundable_credits	90.6	53.1	96.2
5	payroll_tax	92.4	88.7	98.6
6	person_medicaid_eligible	93.0	83.3	95.8
7	free_school_meals_eligible	95.4	58.5	99.9

Source: Article Notebook

5.8 Open-weight time lag

Source: Article Notebook

Source: Article Notebook

The leaderboard also measures time. Following the lag construction the UK AI Security Institute applied to cyber capabilities (UK AI Security Institute 2026), the open-weight lag is the time between the closed frontier first reaching a score and the leading open-weight model matching it. Two cuts of the frozen roster give the trajectory. Among models released before July 2026, the leading open-weight model is DeepSeek V4 Pro at 79.3% exact, whose weights shipped on April 24, 2026 (DeepSeek 2026); the earliest-released closed model above that score is Gemini 3 Flash Preview, released December 17, 2025 (TechCrunch 2025) — a lag of 128 days, about 4.2 months. On the full roster, Kimi K3 at 90.5% exact trails 8 closed models; the earliest released of them, GPT-5.6 Sol, generally available July 9, 2026 (TechCrunch 2026),



Figure 1: Output-level performance on zero-reference and positive-reference cases.

puts the lag at 7 days by API availability, or 18 days dated to Kimi K3's July 27 weight release on Hugging Face (Tech Times 2026).

Both figures are lower bounds in one direction: a closed model outside the evaluated roster that crossed either score earlier would lengthen the corresponding lag. They also carry the uncertainty of any single-benchmark, single-domain estimate — the AISI analysis reports substantially longer lags on cyber tasks — and a lag measured at a moving frontier compresses mechanically when a strong open-weight release lands days after a frontier release.

6 Failure modes

The benchmark surfaces a few recurring failure patterns.

First, models miss positive tax and benefit quantities more often than zero cases. The lowest-scoring outputs by bounded score (Table 11) require the model to choose the right income concepts, exclusions, program thresholds, and sequencing before applying any final subtraction. Positive benefit cases remain difficult, so the result should not be read as a general claim that benefits are easy.

Second, joint accuracy across interacting components can be lower than marginal accuracy on either component. Table 13 shows within-10% accuracy for federal_refundable_credits, state_refundable_credits, and the conjunction of both within the same household. The joint hit rate can be no higher than either marginal and is strictly lower than both for every model except GPT-6.1 Sol, Claude Fable 5.1, Claude Opus 5.5, GPT-6 Astra, Muse Wonder, and GPT-6 Sol. Leaderboard scores that average across outputs therefore overstate how often a model gets a single household's federal/state credit allocation jointly correct.

Source: Article Notebook

Table 13: US within-10% accuracy on federal vs state refundable credits and the household-level joint.

	Model	Federal within 10%	State within 10%	Joint within 10%
35	GPT-6.1 Sol	98.0	92.9	92.9
1	Claude Fable 5.1	100.0	91.9	91.9
7	Claude Opus 5.5	100.0	91.9	91.9
32	GPT-6 Astra	99.0	87.9	87.9
33	GPT-6 Luna	97.0	88.9	87.9
47	Muse Wonder	99.0	87.9	87.9
34	GPT-6 Sol	97.0	86.9	86.9
40	Grok 4.7	97.0	87.9	86.9
30	GPT-5.6 Sol	97.0	86.9	85.9
28	GPT-5.5	93.9	87.9	84.8

Table 13: US within-10% accuracy on federal vs state refundable credits and the household-level joint.

	Model	Federal within 10%	State within 10%	Joint within 10%
14	DeepSeek V4.1 Flash	93.9	86.9	83.8
22	Gemini 3.8 Flash	93.9	86.9	83.8
23	Gemini 4 Argon	96.0	85.9	83.8
36	GPT-6.1 Terra	98.0	84.8	83.8
39	Grok 4.6	94.9	85.9	83.8
13	DeepSeek V4 Pro 0813	93.9	83.8	81.8
20	Gemini 3.6 Flash	93.9	82.8	81.8
21	Gemini 3.7 Flash	89.9	85.9	80.8
42	Inkling	93.9	83.8	80.8
0	Claude Fable 5	91.9	84.8	79.8
15	Gemini 3 Flash Preview	90.9	84.8	79.8
44	Kimi K3	92.9	81.8	79.8
46	Muse Spark 1.3	94.9	81.8	79.8
10	Claude Sonnet 5.5	97.0	79.8	78.8
12	DeepSeek V4 Pro	89.9	82.8	78.8
29	GPT-5.6 Luna	92.9	81.8	78.8
31	GPT-5.6 Terra	93.9	80.8	78.8
38	Grok 4.5	92.9	80.8	78.8
17	Gemini 3.1 Pro Preview	89.9	82.8	77.8
48	GLM-5.3-Flash (preview)	89.9	83.8	77.8
18	Gemini 3.5 Flash	89.9	81.8	76.8
3	Claude Haiku 5.5	86.9	81.8	75.8
6	Claude Opus 5	87.9	82.8	75.8
8	Claude Sonnet 4.6	91.9	81.8	75.8
11	DeepSeek V4 Flash 0731	87.9	81.8	74.7
16	Gemini 3.1 Flash Lite Preview	84.8	81.8	74.7
19	Gemini 3.5 Flash-Lite	85.9	81.8	74.7
27	GPT-5.4 nano	85.9	81.8	74.7
41	Grok Build 0.1	89.9	80.8	73.7
5	Claude Opus 4.8	83.8	81.8	72.7
37	Grok 4.3	83.8	81.8	72.7
26	GPT-5.4 mini	80.8	81.8	71.7
25	GLM-5.3	85.9	77.8	70.7
9	Claude Sonnet 5	79.8	80.8	69.7
45	MiniMax M3	79.8	81.8	69.7
4	Claude Opus 4.7	84.8	78.8	67.7
2	Claude Haiku 4.5	74.7	81.8	66.7
24	GLM-5.2	79.8	74.7	66.7

Table 13: US within-10% accuracy on federal vs state refundable credits and the household-level joint.

	Model	Federal within 10%	State within 10%	Joint within 10%
49	Qwen3.7-max	77.8	81.8	64.6
50	Qwen3.8-Max	66.7	81.8	58.6
43	Kimi K2.6	46.5	50.5	31.3

Source: Article Notebook

Third, structured-output reliability is part of the benchmark contract. Missing or unparseable numeric values are not dropped. Appendix A documents parser recovery and the supervised runner's bounded retry rounds. The 669 cells (0.7%) that still returned no parsable value after retries remain in the canonical files and score zero as parse-contract failures. Appendix A gives the per-model breakdown.

7 Limitations

PolicyBench is not a substitute for a production tax-and-benefit calculator. Several caveats matter:

- output-contract reliability required a repair workflow; raw provider responses are retained within the canonical prediction files wherever the transport exposed them (no raw payload is retained for Claude Fable 5 (1,984 rows), GLM-5.3 (78 rows), and Kimi K3 (64 rows)), and cells that never parsed score zero
- request shape is not identical across models: 10 of 51 models answered the prompt template in several requests, each over a subset of the requested outputs (Table 2), an accommodation that, if it moves scores at all, should favor those models; later additions must answer the whole-scenario request or are listed as not scorable
- zero-heavy outputs require separate positive-case interpretation
- the choice of headline metric (exact match), near-miss companion (within-1%), and secondary tracking metric (bounded score) is one benchmark view; all three are reported, but downstream evaluations should also consider error-magnitude metrics (mean absolute error, mean absolute percentage error) where appropriate
- the public scenario explorer exposes the current test set and reference outputs, so open-set leakage is a prominent limitation
- the 100-household manuscript snapshot should be treated as a preview until larger frozen runs are published
- the annotations cover the 9,050-row legacy-threshold audit universe, including 9,046 exact-match misses and 4 exact hits, but not 2,407 additional rows with bounded score

below 100; the review is developer-adjudicated and not an independent validation of each reference value

- one frozen scenario encodes household inputs that do not match its prompt narrative (a South Carolina household whose prompt implies a qualifying disability the inputs omit), and two input-semantics conventions (disability, which the prompt states as one general fact where each program applies its own determination, and adult tax-dependent relationships) are under-specified in ways a scheduled scenario refresh will address (Section 4.2); outputs whose reference moves under the other reading of either convention are excluded, while the mis-encoded scenario's cells score models against its encoded inputs
- the reference engine can carry defects the audits have not found: policyengine-us 2.60.0 no longer has any defect the audits recorded behind an excluded output; the judge that finds defects reviews only the cases a new model joins, each engine-upgrade review covered only the outputs its upgrade moved, and on 2.15.17 PolicyBench re-ran only four of the September 22 alternative-reading and defect sweeps, so defects that no reviewed case, move or re-run sweep exposes can remain
- the 2026-10-05 review of release dashboard-data-20260930 found eight scored references that rested on inputs the prompt never states and that no judge had flagged, so other such references can remain scored: the exclusions cover the inputs the audits found
- the references were frozen on 2026-07-03 for tax year 2026, so amounts published later reach the benchmark only through a stated convention: where policyengine-us projected a 2026 amount, the reference takes the amount published before the freeze or, failing that, the last one published (SNAP's FY2026 figures in place of USDA's FY2027 figures of 2026-08-21; California's 2025 indexed amounts; the IRS's 2025 sales tax tables), so a model that knew later law is scored against the earlier law, and the projection audit sourced only the amounts whose projection moved an output; law published before the freeze reaches a reference only once policyengine-us encodes it, and policyengine-us encoded New Jersey's 2026 child tax credit schedule and Arizona's SNAP categorical eligibility limit only after 1.755.4
- benchmark success should not be interpreted as policy-advice readiness

The current paper therefore evaluates model performance under a specific structured-output benchmark and certifies no general tax or benefit competence.

8 Conclusion

PolicyBench shows a consistent pattern across the US benchmark. Models often identify non-applicability, but positive tax and benefit amounts remain difficult, especially for multi-step income tax and positive benefit cases. In the frozen manuscript snapshot, GPT-6 Sol is the top-scoring US model.

These results support a narrow conclusion: unaided frontier models still struggle to reproduce selected household-level microsimulation outputs under a structured public benchmark. They do not show that models cannot assist policy analysis, and they do not validate PolicyEngine outputs as administrative truth. They suggest that future evaluations should separate no-tool estimation, tool-using system design, and reference-output validation more explicitly.

Next steps are to expand country coverage, increase frozen sample sizes, and add protected or rotating evaluation sets so that public rankings are less exposed to open-set leakage. The benchmark should also continue reporting sensitivity views, because scores are useful summaries only when their weighting choices are visible.

Appendix A: Structured-output audit

The benchmark requires one numeric value and one explanation for every requested output. Parse failures are benchmark failures and stay in the denominator. Before publishing the manuscript snapshot, we audited missing numeric values and explanations, extended the parser only where explicit variable-keyed value and explanation fields were recoverable, and retried failing scenarios under the supervised runner's bounded rounds. Table 14 summarizes that sequence.

Source: Article Notebook

Table 14: Structured-output parser audit and repair sequence.

Step	Finding
Initial parse-contract audit	The source run was audited for rows missing a parsed numeric value or non-empty explanation; those rows were not dropped from the denominator.
Parser repair	The parser was extended to recover explicit value and non-empty explanation blocks from nested, escaped, or partially truncated provider JSON without scraping prose numbers.
Retry policy	Later waves ran under a resumable supervised runner that retries failed or timed-out scenarios in bounded rounds; every model's canonical file covers all 100 households. No separate whole-response retry artifacts exist for this snapshot.

Table 14: Structured-output parser audit and repair sequence.

Step	Finding
Final parse coverage	669 of the 98,685 canonical model-output rows (0.7%) have no parsable numeric value after retries. The failures comprise Kimi K2.6 (397), GLM-5.2 (136), GLM-5.3 (75), and Kimi K3 (61). They score zero as parse-contract failures. A further 59 rows (Grok 4.3 (55), Kimi K2.6 (3), and Claude Haiku 4.5 (1)) carry a parsed numeric value but no explanation; they are scored on the numeric value and counted here as explanation-contract violations, so 728 rows in all fall short of the numeric-plus-explanation contract. Every other row has both.
Preservation rule	The snapshot retains raw provider responses in the compressed source-run prediction files wherever the transport exposed them; no raw payload is retained for Claude Fable 5 (1,984 rows), GLM-5.3 (78 rows), and Kimi K3 (64 rows) (batch-adapter rows carry no per-row payload, and those parse failures captured none).

Source: Article Notebook

Model waves after June ran under a resumable supervised runner that retries failed or timed-out scenarios in bounded rounds. No separate retry artifacts exist; repairs are folded into each model's canonical prediction file. After retries, 669 cells (0.7% of canonical rows) still returned no parsable numeric value. They comprise Kimi K2.6 (397), GLM-5.2 (136), GLM-5.3 (75), and Kimi K3 (61). These cells remain in the canonical files, with their raw responses preserved wherever the transport exposed one (no raw payload is retained for Claude Fable 5 (1,984 rows), GLM-5.3 (78 rows), and Kimi K3 (64 rows)). They score zero as parse-contract failures, and the denominator is never reduced.

Competing interests

The author leads PolicyEngine, which develops the microsimulation software used to produce the benchmark reference outputs, and is a founder of the Axiom Foundation, a nonprofit developing

deterministic rules-as-code infrastructure. Both organizations' missions are advanced by the finding that unaided language models do not reliably reproduce rule calculations.

Source: Article Notebook

Abbood, Auss, Zaiqiao Meng, and Nigel Collier. 2025. "Time to Revisit Exact Match." Findings of the Association for Computational Linguistics: EMNLP 2025 (Suzhou, China), 11903–26. https://doi.org/10.18653/v1/2025.findings-emnlp.637.

Bock, Michael R., Kara Molisee, Zachary Ozer, and Sumit Shah. 2025. "TaxCalcBench: Evaluating Frontier Models on the Tax Calculation Task." arXiv Preprint arXiv:2507.16126, ahead of print. https://doi.org/10.48550/arXiv.2507.16126.

Chen, Zhiyu, Wenhu Chen, Charese Smiley, et al. 2021. "FinQA: A Dataset of Numerical Reasoning over Financial Data." arXiv Preprint arXiv:2109.00122. https://arxiv.org/abs/2109.00122.

Cobbe, Karl, Vineet Kosaraju, Mohammad Bavarian, et al. 2021. "Training Verifiers to Solve Math Word Problems." arXiv Preprint arXiv:2110.14168. https://arxiv.org/abs/2110.14168.

DeepSeek. 2026. DeepSeek-V4 Preview Release. DeepSeek API news, April 24, 2026. https://api-docs.deepseek.com/news/news260424/.

Federal Reserve Bank of Atlanta. 2026. Policy Rules Database. Federal Reserve Bank of Atlanta. https://www.atlantafed.org/what-we-study/workforce-development/advancing-careers-for-low-income-families/policy-rules-database.

Feenberg, Daniel R., and Elisabeth Coutts. 1993. "An Introduction to the TAXSIM Model." Journal of Policy Analysis and Management 12 (1): 189–94.

Ghenis, Max. 2026. PolicyEngine Powers Rapid Policy Analysis at No 10 Downing Street. PolicyEngine. https://www.policyengine.org/us/research/policyengine-10-downing-street.

Ghenis, Max, and Pavel Makarchuk. 2025. PolicyEngine and Atlanta Fed Sign Memorandum of Understanding for Policy Rules Database Validation. PolicyEngine. https://www.policyengine.org/us/research/policyengine-atlanta-fed-mou-prd.

Guha, Neel, Julian Nyarko, Daniel E. Ho, et al. 2023. "LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models." arXiv Preprint arXiv:2308.11462. https://arxiv.org/abs/2308.11462.

Hendrycks, Dan, Collin Burns, Saurav Kadavath, et al. 2021. "Measuring Mathematical Problem Solving with the MATH Dataset." NeurIPS Datasets and Benchmarks Track.

https://arxiv.org/abs/2103.03874.

Holzenberger, Nils, Benjamin Van Durme, Sarah Lawsky, and Kyle Richardson. 2021. "Factoring Statutory Reasoning as Language Understanding Challenges." arXiv Preprint arXiv:2105.07903. https://arxiv.org/abs/2105.07903.

Nava Labs. 2025. Experimenting with AI-Powered Tools in Public Benefits. Case study. https://www.navapbc.com/case-studies/ai-tools-public-benefits.

Nava Labs. 2026. Evaluating a GenAI-Powered Assistive Chatbot for Caseworkers. Case study. https://www.navapbc.com/case-studies/evaluating-ai-assistive-chatbot-caseworkers.

OpenRouter. 2026. "Ox Alpha: API Pricing & Providers." August. https://openrouter.ai/stealth/ox-alpha.

PolicyEngine. 2024. PolicyEngine Launches State Income Tax Modeling Nationwide. PolicyEngine. https://www.policyengine.org/us/research/state-tax-model-beta.

Shanahan, Catherine, Emma McCarthy, Yan Zhao, et al. 2025. "Performance of LLMs on VITA Test: Potential for AI-Assisted Tax Returns for Low Income Taxpayers." Artificial Intelligence and Law, ahead of print. https://doi.org/10.1007/s10506-025-09465-7.

Shorten, Connor, Charles Pierse, Thomas Benjamin Smith, et al. 2024. "StructuredRAG: JSON Response Formatting with Large Language Models." arXiv Preprint arXiv:2408.11061, ahead of print. https://doi.org/10.48550/arXiv.2408.11061.

Sutherland, Holly, and Francesco Figari. 2013. "EUROMOD: The European Union Tax-Benefit Microsimulation Model." International Journal of Microsimulation 6 (1): 4–26. https://doi.org/10.34196/ijm.00075.

Tech Times. 2026. "Kimi K3 Open Weights Arrive Sunday: Self-Hosting Cuts China Data Risk the API Never Can." July. https://www.techtimes.com/articles/321551/20260725/kimi-k3-open-weights-arrive-sunday-self-hosting-cuts-china-data-risk-api-never-can.htm.

TechCrunch. 2025. Google Launches Gemini 3 Flash, Makes It the Default Model in the Gemini App. TechCrunch, December 17, 2025. https://techcrunch.com/2025/12/17/google-launches-gemini-3-flash-makes-it-the-default-model-in-the-gemini-app/.

TechCrunch. 2026. OpenAI Launches Its New Family of Models with GPT-5.6. TechCrunch, July 9, 2026. https://techcrunch.com/2026/07/09/openai-launches-its-new-family-of-models-with-gpt-5-6/.

UK AI Security Institute. 2026. How Far Behind the Frontier Are Leading Open-Weight

Models on Cyber? Blog post. https://www.aisi.gov.uk/blog/how-far-behind-the-frontier-are-leading-open-weight-models-on-cyber.

Vals AI. 2026. Public Benefits Bench. Benchmark report, in partnership with the Center for Civic Futures and Code for America. https://www.vals.ai/benchmarks/public-benefits-bench.

Willison, Simon. 2026. Kimi K3, and What We Can Still Learn from the Pelican Benchmark. Simonwillison.net, July 16, 2026. https://simonwillison.net/2026/Jul/16/kimi-k3/.

Woodruff, Nikhil. 2026. Informing Policy Using Micro-Simulations. No10 Innovation Fellowship. https://fellows.ai.gov.uk/articles/nikhil-woodruff-micro-simulation.

ZenML LLMOps Database. 2025. Building and Automating Comprehensive LLM Evaluation Framework for SNAP Benefits. Industry writeup. https://www.zenml.io/llmops-database/building-and-automating-comprehensive-llm-evaluation-framework-for-snap-benefits.

Zhou, Ruiwen, Wenyue Hua, Liangming Pan, et al. 2025. "RuleArena: A Benchmark for Rule-Guided Reasoning with LLMs in Real-World Scenarios." Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Vienna, Austria), 550–72. https://doi.org/10.18653/v1/2025.acl-long.27.

Zhu, Fengbin, Wenqiang Lei, Youcheng Huang, et al. 2021. "TAT-QA: A Question Answering Benchmark on a Hybrid of Tabular and Textual Content in Finance." arXiv Preprint arXiv:2105.07624. https://arxiv.org/abs/2105.07624.