POSITION PAPER

Framework for Evaluating LLMs on Complex Data Sets

mathematica.org

Executive Summary

Evaluating the performance of large language models (LLMs) on realistic data analysis tasks is increasingly important for determining their reliability in policy research and applied analytics. Most public LLM evaluations rely on simplified or synthetic data, which do not test a model's ability to filter, weight, and compute valid population estimates—core tasks in statistical practice. To close that gap, Mathematica developed a **prototype LLM Evaluation Framework**, a cloud-native process that measures how accurately and transparently different LLMs generate and execute analytic code processing of complex, survey-based data in response to prompts that have varying levels of detail.

We tested whether the different LLMs could select the right variables, apply survey weights, compute valid statistics, and run basic regression designs. We did this by comparing the LLM outputs to verified, human-developed, gold-standard estimates and documenting the standardized error from the results of applying other performance measures. The framework can run parallel evaluations across model families and versions[1] in two kinds of workflows: **reasoning-only** (which relies solely on the model's internal knowledge to logically decompose and address problems) and **code-execution** (which enables the model to generate an analysis plan, execute code against actual data, and iteratively improve its results).

Taking advantage of a controlled environment with real-world data and tasks used in Mathematica's data analytics work, this initiative demonstrated success in creating an initial process for consistently assessing an LLM's output on statistical data—providing a path toward trustworthy, evidence-based assessment of LLM analytic capability. Moreover, this initiative provides a scalable foundation for future benchmarking efforts and a transparent reference point for understanding the strengths, limitations, and constraints of using LLMs in applied research. Mathematica's approach for evaluating LLMs on frequently used public data sets (for example, the American Community Survey [ACS]) enables evidence-based judgments of their capability, transparency, and accuracy for applied research and policy analysis.

Problem Statement

Despite rapid advances in LLMs, there is currently no standardized method for evaluating their ability to perform statistical reasoning on complex, policy-relevant data sets. In addition, most public LLM benchmarks such as Massive Multitask Language Understanding (MMLU) and GSM-8K use text-based tasks to test factual recall or logical reasoning. These benchmarks offer limited insight into a model's ability to handle structured data, apply survey design concepts, or generate valid statistical estimates.

[1] We assessed four frontier models from two model families: (1) Sonnet 4 (released May 22, 2025) and (2) Sonnet 4.5 (released September 29, 2025) from Anthropic's Claude; and (3) GPT 4o (July 18, 2024) and (4) GPT-5.1 (released November 12, 2025) from OpenAI. We tried to test other LLMs available through AWS Bedrock such as DeepSeek and Llama, but the Bedrock API did not support tool and function calling for these versions.

Social scientists and policymakers study questions whose answers depend on the models' ability to perform specific tasks such as filtering microdata, applying sample weights, and computing statistical estimates of the population sampled in the data. Without systematic evaluation of how a model performs on these tasks, organizations risk overstating a model's readiness for data analysis, undermining both research quality and trust in AI-assisted methods.

Recent research underscores the importance of grounding LLM evaluation in real analytic workflows. The DiscoveryBench study[2] demonstrated that even advanced reasoning models perform inconsistently when asked to replicate scientific analyses, achieving only modest accuracy on real data sets. The Mathematica team wanted to establish a process in which we could prompt the LLM with descriptive questions like "What is the total number of benefit-eligible veterans in Alabama who faced employment barriers in 2021?" and then be able to systematically confirm the quality of the output.

To achieve this, we define a lightweight evaluation framework for assessing LLM performance on applied statistical tasks. The framework separates the conceptual components of evaluation (what is being tested) from the specific implementation choices used in this study (how it is tested). In this paper, we describe the framework briefly and then report results from applying it to real ACS-based analytic tasks.

Our Framework

Our evaluation framework consists of:

- A data set with survey design features and a set of descriptive and causal analytic questions
- Quantitative and qualitative evaluation metrics
- Explicit tooling constraints (**reasoning-only** versus **code execution**)
- A controlled execution environment

Together, these components focus evaluation on the kinds of inferential reasoning that underlie applied research and provide a structured way to assess whether model outputs align with statistical best practice when applied to real-world survey data.

Curating the data set

For the data set and analytic task components of the framework, we built 35 validated question-and-answer pairs using the ACS Public Use Microdata Sample (PUMS) for the years 2010 through 2024.[3] We chose the ACS because it provides a rich and policy-relevant data set containing hierarchical person- and household-level records with sampling weights and replicate weights.

[2] Majumder, Bodhisattwa Prasad, et al. "DiscoveryBench: Towards data-driven discovery with large language models." *arXiv preprint arXiv:2407.01725* (2024)

[3] We evaluated 24 descriptive questions and 11 validated difference-in-difference estimates.

We drew on Mathematica's deep expertise using ACS data to answer a variety of real-world descriptive and causal analytic questions such as estimating individual eligibility for program benefits, and identified clean, quality-assured data sets that we used to generate question-and-answer pairs for population estimates and difference-in-difference estimates.[4]

Prompt design and workflows

For the **tooling-constraint component** of the framework, we ran parallel evaluations across two core workflows, each designed to isolate different dimensions of model reasoning:

- **Reasoning-only.** The model answers an analytic question without data access, relying solely on its internal knowledge. This baseline helps detect data leakage (when the model's answer reflects memorized information rather than true reasoning) and assess how much improvement comes from genuine computation used in the code-execution workflow. For causal inference questions, the prompts specified a difference-in-difference design, including the definition of comparison groups, pre/post periods, and required clustering and weighting choices. This allowed us to assess not only whether models could compute a treatment effect, but whether they could correctly implement each component of the causal design.
- **Code execution.** The model generates executable Python code to answer an analytic question using data accessed through a controlled application programming interface (API), iteratively refining the analysis as needed. This simulates how an analyst would interact with a data set and reason through the steps of loading data, filtering relevant rows and columns, making necessary data transformations, and computing results.

Both workflows were orchestrated through a fully serverless pipeline using Amazon Web Services (AWS) Lambda, Bedrock, OpenAI API, and LangChain. All runs were executed in a restricted cloud environment with controlled data access interfaces.

Evaluation metrics

For the evaluation metrics component of the framework, Mathematica compared each model's output against verified ACS benchmark estimates using a mix of quantitative and qualitative measures.[5] The goal was to assess not only how close model-generated numbers were to the true values, but also whether models followed valid statistical steps to get there. For numeric performance metrics, we center this position paper's discussion on the exact match rate for simplicity; Appendix Exhibit A.1 presents other metrics that we looked at.

We also qualitatively analyzed whether the model correctly applied survey weights, filtered data appropriately, and followed valid statistical methods and conducted error analysis to identify patterns of failure and/or confusion. For causal inference tasks requiring regression models, we reviewed model-generated code to assess whether regressions followed the specified formula; used the expected Python libraries; and applied the correct weighting, clustering, and filtering operations.

[4] To reduce this effort, we replicated the existing workflows and point estimates in Python because LLMs such as ChatGPT run Python code in their environments and not Stata or R.

[5] This analysis does not assess run-to-run variability or within-model variance across repeated executions of identical prompts; results therefore reflect point accuracy rather than stability metrics.

Our Implementation

Mathematica's LLM Evaluation Framework runs entirely in the cloud and was designed to be both secure and reproducible. It processes questions through two workflows: one in which the model can query the data and writes and executes Python code (code execution) and another in which it reasons using the model's knowledge base without access to the data (reasoning-only).

The tool uses AWS Lambda to run each test in isolation and LangChain to manage interactions between models, data, and prompts. All data access takes place through a controlled interface that allows models to load only approved variables and return structured outputs, ensuring consistency and transparency. Results are stored automatically for comparison and documentation. Each evaluation follows a consistent pattern:

1. A set of analytic questions is uploaded to the tool.
2. The framework sends each question to one or more models as specified.
3. Models either generate executable code to analyze the ACS data or produce a direct answer.
4. Code runs in a secure cloud environment using local copies of the data set.
5. Models are given the opportunity to iterate and test their approach a fixed number of times.
6. The resulting estimates, explanations, and metadata are saved for review.

This approach allows many models to be tested in parallel and ensures that every step of the reasoning and computation can be traced and replicated.

In addition to evaluating analytic reasoning, the framework was designed with secure-environment controls enabling safe use of microdata that may include sensitive or personally identifiable information. The Mathematica LLM Evaluation Framework's architecture supports sandboxed execution, role-based access, versioned logging and restricted data flows, offering an early building block for using LLMs with sensitive microdata in trusted environments.

Results

Our evaluation across 35 human-validated questions revealed clear performance patterns tied to workflow design, model family, model edition, and the statistical complexity of the task.

Reasoning-only. In the reasoning-only workflow—in which models attempt to answer questions purely based on reasoning and prior knowledge—none of the models achieved an exact match. For causal questions, models frequently produced results that differed by orders of magnitude from the validated results. In this setting, models rely on broad heuristics such as multiplying state-level population guesses by assumed participation rates rather than attempting to reconstruct the required population universe. As a result, this workflow serves as a useful data leakage check but cannot support policy-relevant analysis. Although not unexpected given that this task requires access to structured data, sampling weights, filters and variable definitions, the result underscores the baseline gap between reasoning-only and data-driven workflows.

Code execution. In the code-execution workflow, in which the model generates and runs analytic code on the ACS data, performance improved notably compared with the reasoning-only setting. However, improvements varied by model and by type of statistical task.

All frontier models achieved high accuracy on descriptive tasks in this setting, but with meaningful differences across models:

- Claude Sonnet 4, Sonnet 4.5 and GPT-5.1 performed similarly well, with exact-match rates between 88–92 percent.
- GPT-4o consistently trailed the other frontier models, with an exact-match rate of 75 percent.

For causal inference questions, performance diverged more notably by model:

- GPT-5.1 achieved near-perfect accuracy with an exact-match rate of 100 percent.
- Claude Sonnet 4 and Sonnet 4.5 also achieved relatively high accuracy with 82 percent and 73 percent exact-match rates, respectively.
- GPT 4o achieved a 64 percent exact-match rate, considerably lower than that of the other models.

Unlike descriptive tasks, for which all models have relatively straightforward instructions, causal inference requires several coordinated decisions: selecting the correct comparison groups, defining pre/post periods, applying weights, and estimating a treatment effect through regression.

Tooling. This evaluation also drew attention to tooling and framework design as key enablers: the agent-tooling paradigm (querying data via custom tools, sandboxed execution, and standardized prompts) proved essential for reliably measuring performance differences and tracing methods. Likewise, causal inference tasks (in this case, difference-in-differences) surfaced unexpected model behavior: for example, some models did not cluster standard errors on the specified cluster and instead defaulted to clustering at the level of fixed effects (for example, state-year); others managed the correct clustering but used regression libraries that were different from our own solutions (for example, statsmodels versus linear models). However, when different package choices resulted in differences in the output estimate, this was usually due to inappropriate modeling options or inappropriate filtering of the data.

Together, these findings affirm that the combination of structured data set access, analytic tooling, and reproducible workflows yields measurable improvements, but also that substantial room remains for models to handle the full complexity of applied statistical estimation.

Lessons Learned

Testing large language models (LLMs) on real survey data revealed both the promise and current limits of their analytic reasoning. Across dozens of expert curated questions drawn from the ACS microdata, models exhibited clear differences in how they assess metadata, employ tooling, interpret prompt structure, and execute statistical tasks.

Data and metadata matter

Model accuracy depended on how clearly variables were labeled and documented. When variables were not sufficiently documented and clearly defined, the models tended to select the wrong outcome variables and apply the wrong filters on the data. This occurred in part because the prompts described outcomes in plain, non-technical language, and the model had to infer which technical variable in the data set matched that description. When data sets contained multiple columns with similar names, the model sometimes mapped the plain-language description to the wrong variable. Overall, this finding underscores the notion that data usability and metadata clarity are foundational for AI readiness in analytic workflows.

Tool-calling improves reliability

Access to code execution substantially improved accuracy for descriptive estimates across all models, reducing errors by several orders of magnitude relative to reasoning-only. However, code execution did not guarantee success for causal inference: although GPT-5.1 consistently produced correct difference-in-difference estimates, other models generated code that was syntactically valid but methodologically incorrect. Some models dropped required groups or years, added unintended fixed effects, or mis-specified the interaction term. Tool-calling is necessary but not sufficient for valid causal estimation—models must also reliably adhere to econometric design constraints.

Prompt structure shapes outcomes

Well-structured prompts that clearly defined available variables (including names, types, weights) and articulated the desired output format led to sharper and more accurate responses. In contrast, prompts that left variable definitions implicit or skipped specification of weights resulted in degraded performance. This suggests that prompt engineering remains a critical lever for applying LLMs in statistical workflows. In addition, causal inference prompts revealed that models interpret instructions differently depending on internal reasoning styles. Claude Sonnet 4.5, for example, tended to introduce additional data transformations or modeling choices that were not requested, leading to lower accuracy. These findings suggest that future prompting strategies may require more rigid templates or guardrails, particularly for multi-step causal designs.

Statistical competence remains uneven

Statistical competence varied by model family and task type. All frontier models performed well on descriptive statistics when granted data access, but only GPT-5.1 consistently implemented the correct causal design and achieved near-perfect accuracy on difference-in-difference estimates. Claude Sonnet 4.0 and 4.5 achieved partial success, with errors driven by mis-specification of interaction terms and incorrect filtering. GPT-4o underperformed on both descriptive and causal tasks relative to the other frontier models. These results highlight that small differences in model reasoning style can produce significantly different analytic outcomes.

Reproducibility is achievable

Despite varied accuracy levels, every analysis run was logged, versioned, and traceable, from prompt version and model type to code executed, data set snapshot used, and output logs. That demonstrates the potential for reproducible and automated evaluation of LLMs on complex data sets. This represents a scalable foundation for future benchmarking efforts as models continue to evolve.

In Conclusion

Testing LLMs on federal statistical data used to generate population estimates and measure outcomes revealed both the promise and current limits of their analytic reasoning. By using a data set based on ACS microdata with its complex weights, filters, subgroup definitions, and metadata, the framework surfaces meaningful differences in model design, tooling, prompt structure, and data usability.

Advancing analytic readiness

As noted, one of the most important implications of this exercise is that reasoning alone is insufficient for credible statistical estimation in policy-relevant settings. Our results show that models with access to data and code-execution capability deliver substantially higher accuracy for descriptive tasks. This suggests that any organization considering LLM-powered analysis should treat data set access, tool invocation, and workflow automation as prerequisites for trustworthy output, not optional add-ons.

Implications for prompting, metadata, and workflow design

We found that metadata clarity, prompt structure, and tooling architecture matter as much as or more than the model version used. In practical terms, this means that before placing an LLM into an analytic workflow, organizations should:

- Ensure that data sets include clear data element labels (such as table and column names and metadata) and that documentation used as context for the LLM include unambiguous variable definitions
- Provide prompts that explicitly define analysis tasks, variables, weights, sample design, output format, and error structure
- Use frameworks that support model code-calling, sandboxed execution, version control, and traceability

By combining tooling, metadata clarity, and workflow controls, the tool helps shift an LLM from a "black box" output to one whose analysis can be more closely governed, reviewed, and traced.

Appendix

Exhibits A.1 and A.2 present model performance metrics for the descriptive and the difference-in-difference questions, respectively. We report the following metrics:

- **Mean absolute error (MAE).** On average, how far off were the model's answers from the verified numbers?
- **Root mean square error (RMSE).** If the model makes very large errors, RMSE weights them more heavily.
- **Relative absolute error (RAE).** Expresses the model's error as a share of the verified value.
- **Symmetric mean absolute percentage error (SMAPE).** Handles miniscule or near-zero numbers better than RAE by balancing the difference between estimates and verified values.
- **Normalized root mean square error (NRMSE).** How much, in relative terms, the model tends to miss the mark on big numbers.
- **Exact match rate (EMR).** How often did the model get the number exactly right?

Exhibit A.1. Performance metrics for frontier LLMs, descriptive questions only

	Mean absolute error (MAE)	Root mean square error (RMSE)	Exact match rate (EMR)	Number of nulls	Normalized root mean square error (NRMSE)	Symmetric mean absolute percentage error (SMAPE)
Reasoning-only						
Claude Sonnet 4	8,476,246	20,704,111	0%	0	593	104%
Claude Sonnet 4.5	8,019,081	19,441,147	0%	0	557	92%
GPT 4o	11,362,903	26,997,163	0%	0	773	111%
GPT 5.1	4,271,274	12,485,188	0%	2	344	100%
Code execution						
Claude Sonnet 4	26,414	87,677	88%	0	3	3%
Claude Sonnet 4.5	360,227	1,733,129	92%	0	50	4%
GPT 4o	424,604	1,915,660	75%	1	54	14%
GPT 5.1	360,364	1,733,129	88%	0	50	4%

Note: We evaluated LLMs with 24 validated descriptive questions, using a custom cleaned data set to assess program eligibility.

Exhibit A.2. Performance metrics for frontier LLMs, difference-in-difference questions

Diff-in-diff questions performance	Mean absolute error (MAE)	Root mean square error (RMSE)	Exact match rate (EMR)	Number of nulls	Normalized root mean square error (NRMSE)	Symmetric mean absolute percentage Error (SMAPE)
Reasoning-only						
Claude Sonnet 4	0.024	0.029	0%	5	2750	188%
Claude Sonnet 4.5	0.025	0.039	0%	2	3086	179%
GPT 4o	0.051	0.051	0%	8	6031	199%
GPT 5.1	0.006	0.007	0%	2	596	196%
Code execution						
Claude Sonnet 4	0.001	0.002	91%	0	160	18%
Claude Sonnet 4.5	0.003	0.005	73%	0	400	54%
GPT 4o	0.001	0.004	64%	3	327	25%
GPT 5.1	0.000	0.000	100%	0	1	0%

Note: We evaluated LLMs with 11 validated difference-in-difference estimates, using a custom cleaned data set to assess program eligibility.