From Simple to Impossible: How Task Complexity Limits AI Research Assistants

Nathaniel Lovin and Scott Wallsten

November 2025

2001 L Street NW ● Suite 500 ● Washington, DC 20036
Tel: (202) 828-4405 ● Email: info@techpolicyinstitute.org ● www.techpolicyinstitute.org

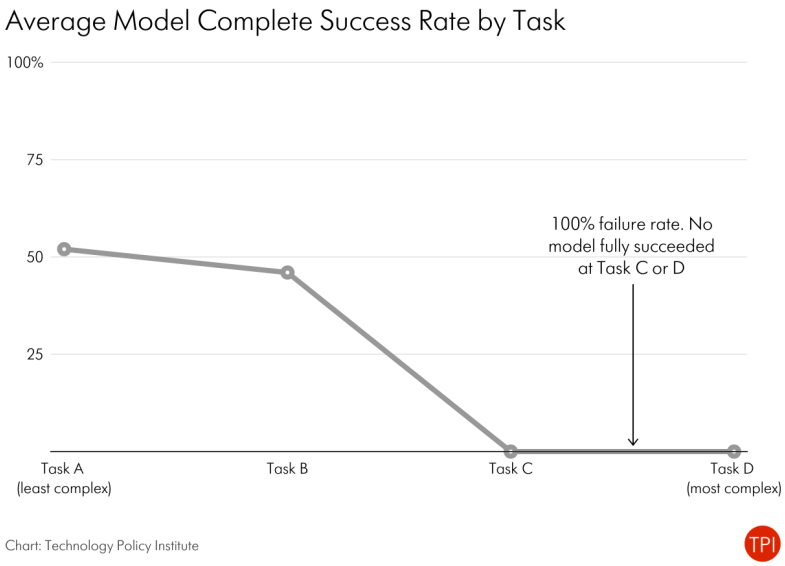
From Simple to Impossible:
How Task Complexity Limits AI Research Assistants

Nathaniel Lovin[*]
Scott Wallsten[**]

November 2025

Abstract

We develop a benchmark to evaluate AI agents' ability to acquire data and perform analysis tasks common in social science research. Testing six state-of-the-art agents (OpenAI, Anthropic Claude 3.5-4.5, and Google Gemini) on progressively complex tasks using the FCC's broadband database, we find agents achieve moderate success on simple single-file downloads (52% average) but fail completely when required to download multiple data files (0% success) or perform basic analysis (0% success). Logistic regression analysis reveals that task complexity, specifically multi-year data requirements and analysis requirements, drives these failures, while instruction specificity has minimal and statistically insignificant effects. Despite dramatic year-over-year improvements in individual model performance (Claude's failure rate dropped from 76% to 4% on the simplest task), fundamental capability barriers remain: agents cannot reliably complete straightforward data workflows that take humans under a minute.



[*] Lead programmer and senior research analyst, Technology Policy Institute
[**] President and senior fellow, Technology Policy Institute

Introduction

Artificial intelligence "agents" are the year's buzzword. They are supposed to autonomously decide the tasks and workflow necessary to achieve a given outcome, promising to perform complex tasks for us (i.e., acting as our agents). A prototypical example would be finding and booking flights without any human intervention beyond being given the acceptable parameters.

In the last year, AI Agents have gone from "highly experimental" to "experimental" thanks, in part, to "thinking"/"reasoning" models like OpenAI's o3 and DeepSeek's r1[1] and targeted training on tool-use. Of particular interest is "computer use" agents like those from OpenAI and Anthropic. These agents are able to use a browser to perform actions that a human could, like clicking on links and downloading files. ChatGPT, for example, shows the website the agent is browsing as it does so.

AI agents are in their infancy and, just as certain instruments track the development of AI models, it is similarly possible to track the progress of agents in order to understand their effectiveness and development.

A few agent benchmarks already exist.[2] They show generally that the longer a task takes humans to complete, the less successful AI agents tend to be at it.[3] Additionally, agent capabilities are improving rapidly: every seven months, agents can successfully complete tasks that take humans twice as long as before.

Those tests, however, focused only on software engineering. Only a few tests focused on other fields, and those yielded conflicting results.[4] Additionally, benchmarks should include methods of testing increasingly more complex tasks.[5]

In this article, we develop a simple benchmark test for a data analytic task unrelated to software development. We use it to compare AI agents to each other and to humans.

We find that the agents performed reasonably well on simple, well-specified tasks (52% average success rate) but failed completely when asked to download all available data files (0% success) or perform basic

---

[1] LLMs trained in part with Reinforcement Learning from Verifiable Rewards that produce a hidden "chain of thought" "internal monologue" before generating a final output

[2] OSWorld evaluates simple computer use tasks that would take an average person only a few minutes. OpenAI's o3 with Computer Use hit 42.9% with 200 steps (Xie et al 2024). SWE-Bench Verified (https://openai.com/index/introducing-swe-bench-verified/) is based around real-world software engineering tasks collected from github, and METR's RE-Bench (https://metr.org/blog/2024-11-22-evaluating-r-d-capabilities-of-llms/), a benchmark of long time (2hrs+) AI research tasks. Currently, the best model on SWEBV is Claude Sonnet 4.5Opus 4.1, at 65% (https://epoch.ai/benchmarks/swe-bench-verified). METR has not published recent RE-Bench data separate from their time horizon's work, however, GPT-5 was able to do 2 hr 15 tasks correctly 50% of the time on a combination of SWEBV, Re-Bench, and shorter tasks (https://evaluations.metr.org/gpt-5-report/, https://arxiv.org/abs/2404.07972).

[3] https://metr.org/blog/2025-03-19-measuring-ai-ability-to-complete-long-tasks/

[4] https://metr.org/blog/2025-07-14-how-does-time-horizon-vary-across-domains/

[5] Or as close as possible with underlying errors in the evals - see for example this discussion of possible issues with GPQA Diamond https://epoch.ai/gradient-updates/gpqa-diamond-whats-left

analysis (0% success). While Claude models showed dramatic year-over-year improvement, with failure rates on the simplest task dropping from 76% to 4%, even the best agents cannot reliably complete straightforward data acquisition workflows that take humans under a minute. Regression analysis reveals that task complexity, not instruction specificity affects agent success.

Method

Social science tasks seem to be disproportionately represented among AI uses.[6] Researchers tend to use varied datasets for different analyses and, outside of large, well-funded organizations, are unlikely to be able to afford purpose-built models or agents. As a result, agents that can find, clean, merge, and otherwise assemble datasets, particularly publicly available datasets, could be useful to practitioners.

To that end, we develop a benchmarking approach to assess how well models/agents can do these tasks. Specifically, we focus this test around how well agents can obtain and work with data on U.S. broadband deployment from the Federal Communication Commission's (FCC's) national broadband map.[7]

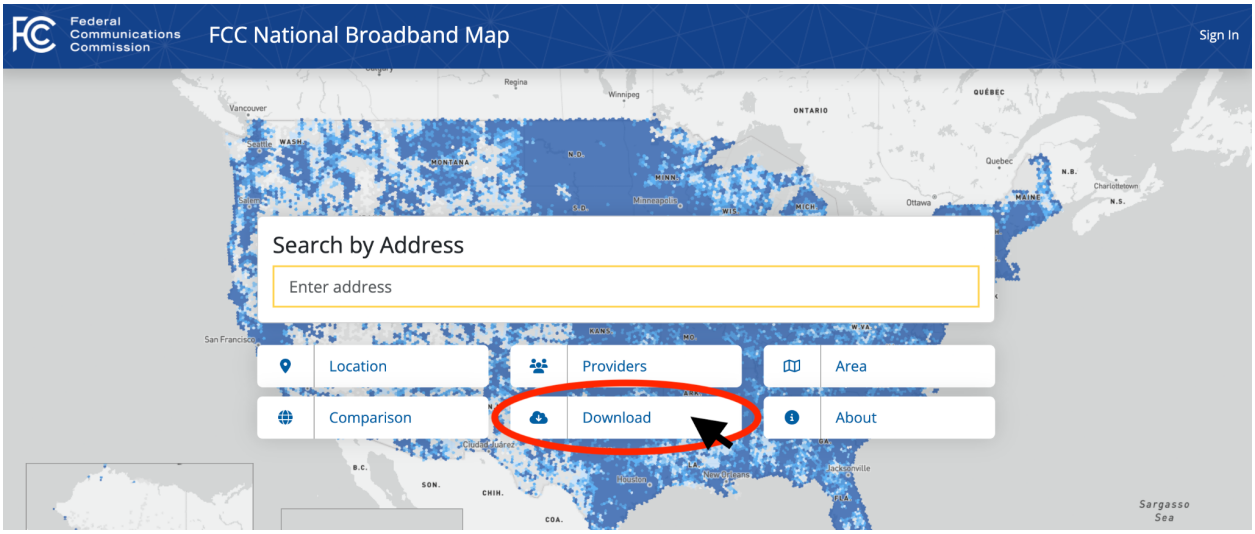
We measure the agents against four prompts of varying complexity of task and completeness of directions. Both matter. An agent becomes more useful as it is better able to complete complex tasks and to do so without detailed instructions. That is, an agent can be useful if it can complete complex tasks if the user provides detailed instructions, and can also be useful if it can complete simple tasks without being told how to do it, but it is most useful if it can do complex tasks without being told how to do them.

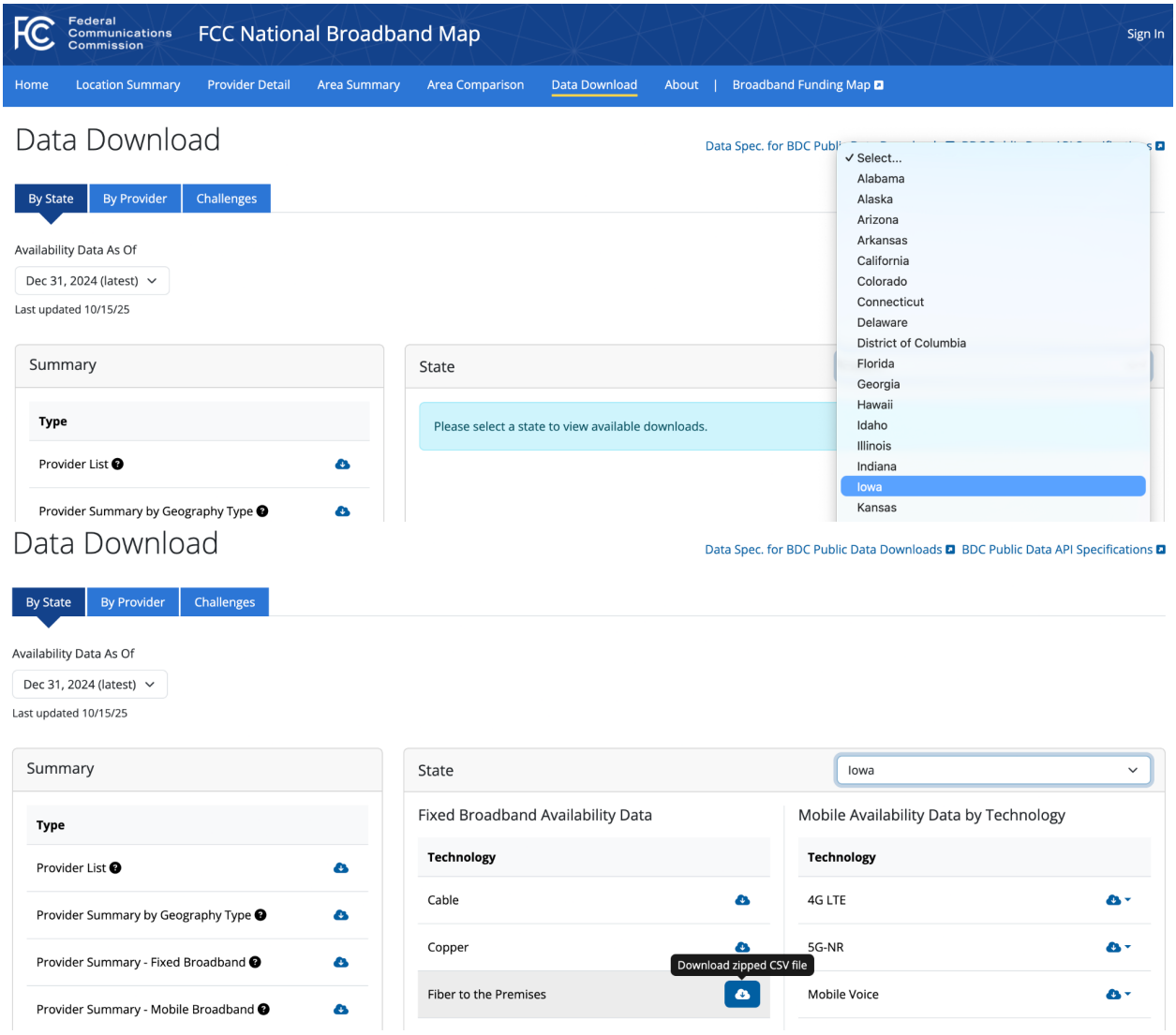
For this benchmark we give an agent tasks of increasing complexity and fewer instructions on how to complete the task. The hardest test here is for an agent to answer a specific question by downloading the data needed to answer the question and figure out how to run a simple analysis. The conceptual simplicity masks significant complexity. An agent needs to know how to find the data, navigate various web interfaces, make decisions about what data is required, download the data, and then perform calculations using those downloaded datasets.

We ask the agents to download certain data on broadband from the Federal Communication Commission's (FCC's) broadband map database and then perform some simple calculations. We built up the test with the following prompts.

Task Type A: Exact instructions, Single year download
Prompt: "Go to https://broadbandmap.fcc.gov, click the 'Download' button, and download the most recent fiber data for Iowa"

[6] See Wallsten, Scott. "Measuring AI Intensity by Occupation: Adjusting for Workforce Size." November 2025. https://techpolicyinstitute.org/publications/artificial-intelligence/ai-intensity-by-occupation-adjusting-for-workforce/
[7] I use the FCC data because it is well-defined, available at a specific online location, and updated regularly. Additionally, because I have worked with these data extensively I know the structure of the datasets and the website as well as other details that make it easier for me to understand the tasks I assign. Iowa because it is my home state. Go Hawkeyes.





Task Type B: Vague instructions, Single year download

Prompt: “Go to https://broadbandmap.fcc.gov/home and download the most recent fiber data for Iowa”

Task Type C: Vague instructions, Multi-year download
Prompt: “Go to https://broadbandmap.fcc.gov/home and download all the fiber data for Iowa”

Task Type D: More specific instructions, Multi-year download, and some analysis
Prompt: “Download the data from https://broadbandmap.fcc.gov/ to tell me how many locations in Iowa gained fiber availability each year between Dec 2022 and Dec 2024”

The four tasks increase in complexity along three dimensions: instruction specificity (exact vs. vague), data scope (single year vs. multiple years), and task requirements (download only vs. download + analysis). This design allows us to isolate the separate effects of each factor.

We tested models from Google, OpenAI, and Anthropic using their APIs. OpenAI and Google both offer one model trained to do computer use tasks, while Anthropic offered four versions of Claude with computer use. The oldest Anthropic model, Claude 3.5 (new), was discontinued midway through this experiment.[8]
We ran each prompt 25 times per model. Each model run continued until the model failed to return a call to the computer use tool or for fifty steps, whichever came first. If the model asked a question about what to do containing “Proceed”/”continue”/”next step”/”like me to”/”Should I” I replied “Proceed with the next step.” If the model required any other human-in-the-loop step, the run terminated.[9]

Results[10]

Broadly speaking, the results are as one would expect: agents performed worse as instructions became less specific and tasks became more complex. Notably, none of the models were able to successfully

[8] We tested the following models: OpenAI’s “computer_use_preview”; Anthropic’s “claude-3-5-sonnet-20241022”, “claude-3-7-sonnet-20250219”, “claude-4-sonnet-20250514”, and “claude-sonnet-4-5-20250929”; and Google’s “gemini-2.5-computer-use-preview-10-2025.” For each model We built a simple agent scaffolding around the API to control a standardized E2B sandbox# with Chrome installed.
E2B is a startup offering cloud sandboxes for AI Agents to run programs with. https://e2b.dev/
At each step, the screenshot of the current screen is sent to the model. The model then returns call to the computer use tool, an action to be taken. After the action is performed, a new screenshot is taken and the loop repeats. As the screenshots of the current position take up many tokens in the context window, for both cost and performance reasons the message history must be condensed as new steps are taken. The OpenAI API automatically condenses the model responses as more messages get added to the “conversation”/message history. For both Anthropic and Google I sent the message history to a separate instance of the given model every 5 responses to condense it.
[9] OpenAI’s Computer Use is extremely eager to confirm what it’s doing is right - “Should I proceed to download the file?” “Should I proceed to extract the zip?” etc. Ensuring human-in-the-loop is important, but for simple, safe tasks like these it’s a lot, and I’m a bit worried we’ll end with a Homer Simpson’s Drinking Bird situation, mindlessly hitting yes over and over.
As the models were running in secure, isolated, E2B sandboxes with no personal data, slightly automating the human in the loop seemed low-risk.
[10] Full results, along with a link to a ZIP file of the model outputs, are available in the appendix.

analyze the data, as instructed in Task D. This section provides an overview of the results, detailed summary tables, and finally regressions that explore the role of specific instructions and complexity of the request.

Figure 1: Success Rates

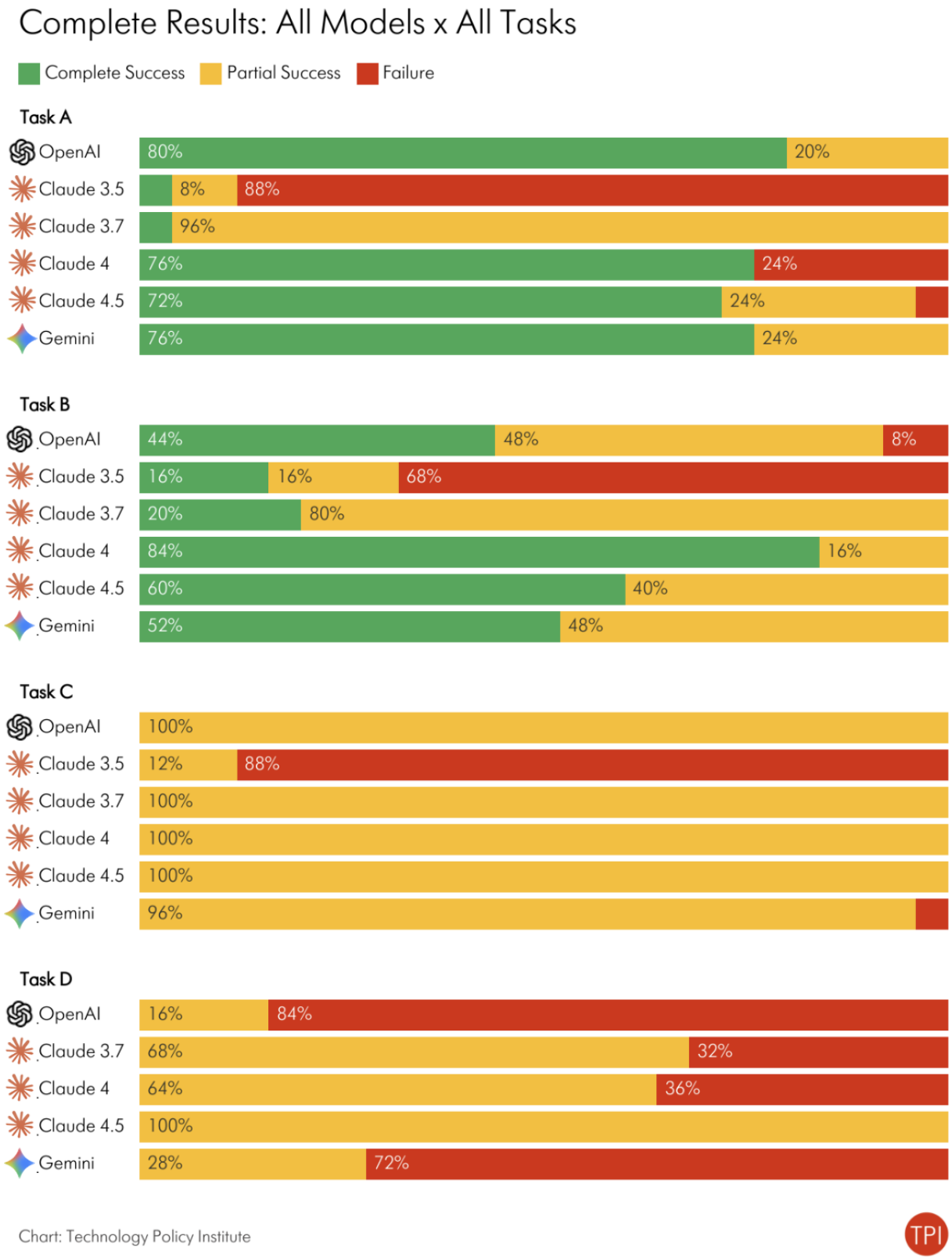
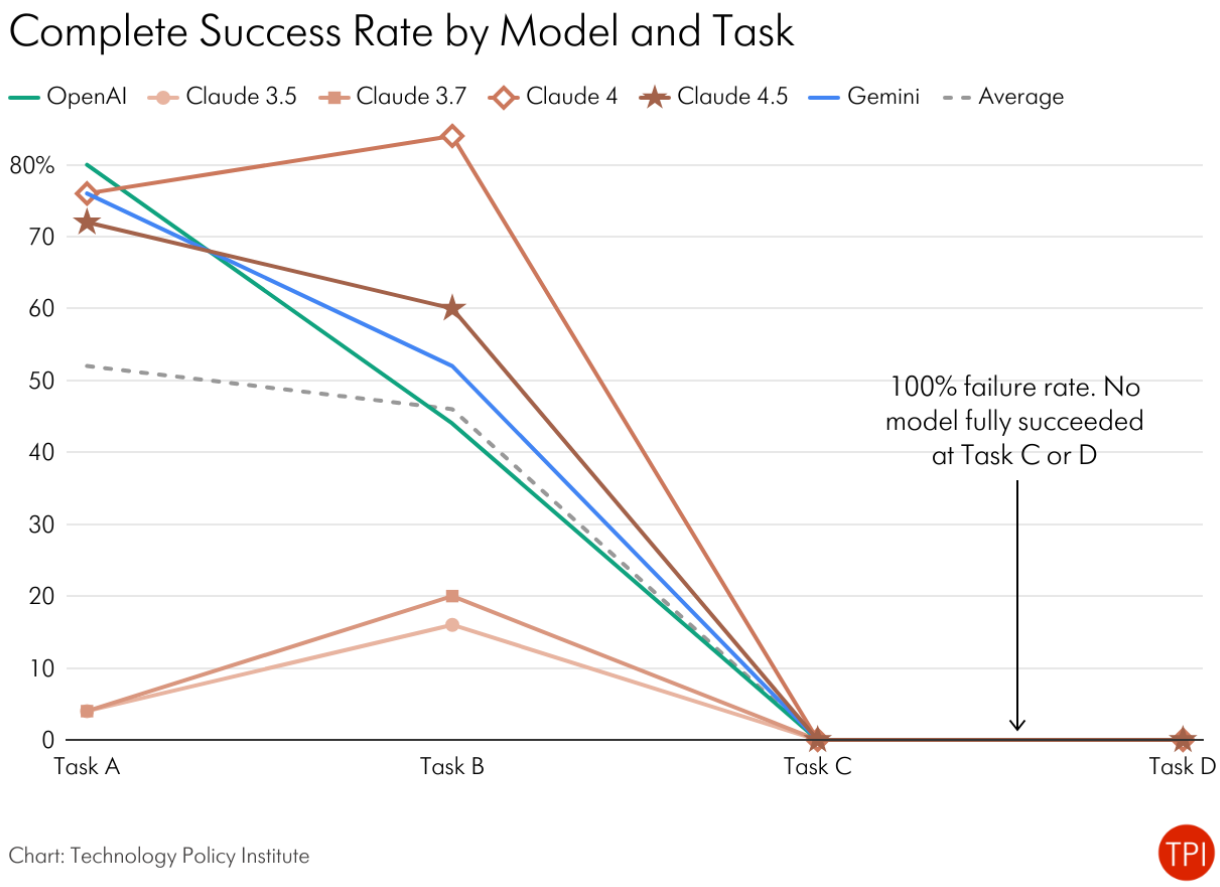
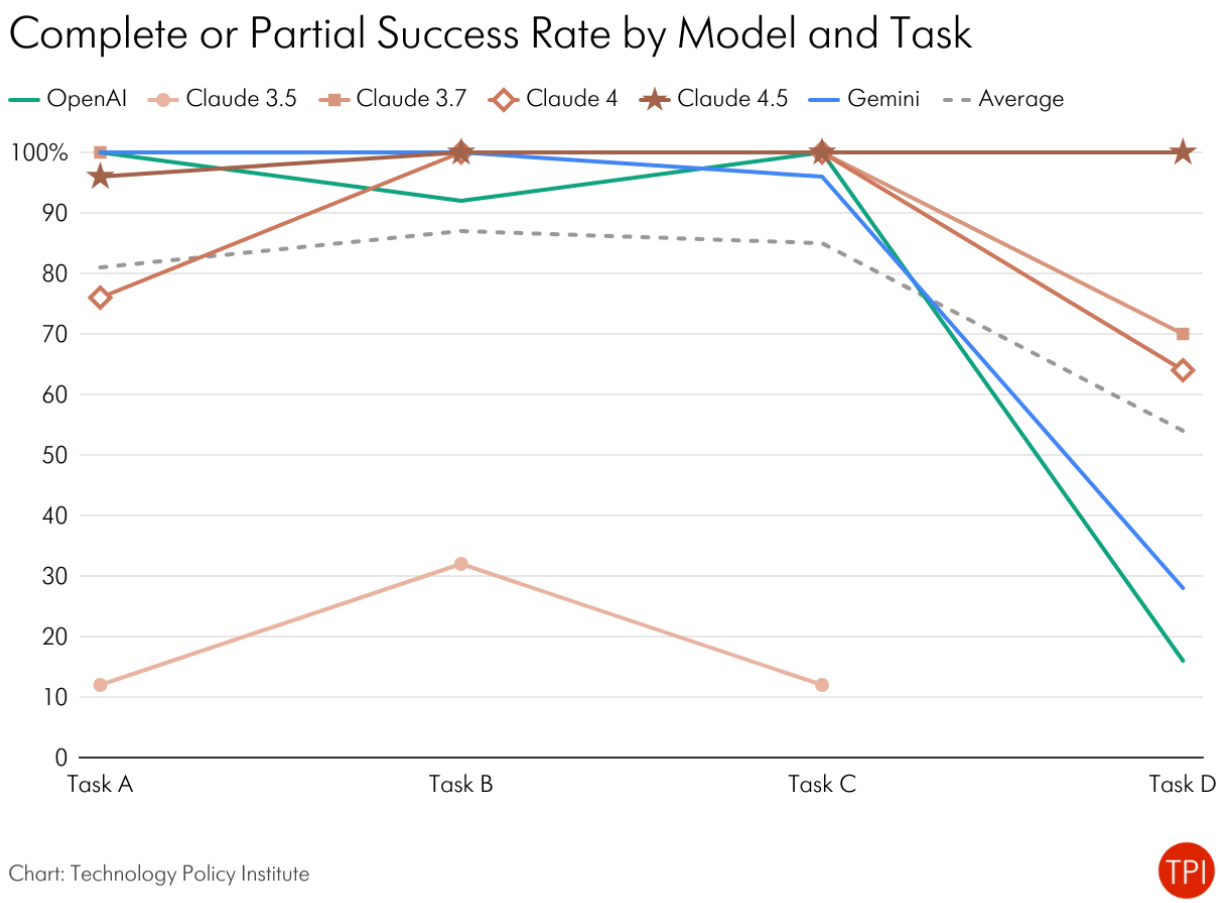


Figure 2: The Capability Cliff





The tables below show more detailed results.

Table 1 presents overall performance patterns. On simple download tasks(A and B), models achieve 46-52% complete success on average.[11] However, complete success drops to zero when tasks require downloading multiple years of data (Task C) or performing analysis (Task D). Measuring 'any progress' reveals that most models make partial progress on Task C (85% average) but struggle even to attempt Task D (59% average), suggesting two distinct capability barriers.

Table 1: Complete Success Rate by Model and Task Type

	Task A (Exact instructions, Single year download)	Task B (Vague instructions, Single year download)	Task C (Vague instructions, Multi-year download)	Task D (Exact instructions, Multi-year download, and some analysis)
OpenAI	80%	44%	0%	0%

[11] Descriptive averages are unweighted across models (e.g., 69% for Task A); regression-based averages weight all runs equally (e.g., 52% for Task A). The difference reflects model heterogeneity.

Claude 3.5	4%	16%	0%	—
Claude 3.7	4%	20%	0%	0%
Claude 4	76%	84%	0%	0%
Claude 4.5	72%	60%	0%	0%
Gemini	76%	52%	0%	0%
Average	52%	46%	0%	0%

Notes: Based on 25 runs per model per task (n=150 per task for Tasks A-C; n=120 for Task D due to missing Claude 3.5).

Table 2
Task A Results

	Model					
	OpenAI	Claude 3.5	3.7	4	4.5	Gemini
Success rates						
Correctly downloaded	80%	4%	4%	76%	72%	76%
Semi-Success rates						
Downloaded and processed	20%	8%	0%	0%	0%	0%
Downloaded multiple times	0%	0%	96%	0%	24%	24%
Failure rates						
Incorrect download	0%	12%	0%	0%	0%	4%[12]
Full failure	0%	76%	0%	24%	4%	0%

The most common way the models failed with the direct instruction prompt was downloading the dataset multiple times, not realizing that they had already successfully downloaded it. While not a strict problem

[12] Downloaded multiple times and also downloaded wrong file once

for this use, it could be problematic in the case of larger datasets or bandwidth limitations as well as additional token costs.

Claude 3.5 struggled to use the dropdown menu on the FCC site, correctly selecting Iowa but then attempting to scroll down the page without clicking out of the selector, leading to it selecting "Louisana" instead of scrolling down to the fiber data button. It then got stuck in a loop of the same failure for most of the runs. Claude 3.5 was also the only model, plus one run of Gemini, to download the wrong data (Iowa's cable data, North Dakota's fiber data, or trying to download Centurylink's data).

Claude 3.5 and OpenAI were the only models that attempted to do anything with the data once downloaded even though it was not instructed to. These actions mainly consisted of opening the downloaded zip file, although OpenAI also opened the resulting csv file twice in LibreOffice.

Table 3
Task B Results

	Model					
	OpenAI	Claude 3.5	3.7	4	4.5	Gemini
Success rates						
Correctly downloaded	44%	16%	20%	84%	60%	52%
Semi-success rates						
Downloaded multiple times	8%	0%	80%	16%	40%	48%
Downloaded and processed	40%	16%	0%	0%	0%	0%
Failure rates						
Incorrect download	0%	8%[13]	0%	0%	0%	16%[14]
Full failure	8%	60%	0%	0%	0%	0%

Task B performance is similar to Task A. A logistic regression predicting complete success on Tasks A and B (discussed in detail below) shows that vague instructions reduce success by 8.6 percentage points,

[13] Downloaded wrong file as well as correct file
[14] Downloaded multiple times and also downloaded wrong files

but this effect is not statistically significant (p=0.214) and is dwarfed by model fixed effects. None of the models went to the FCC's map instead of the download page, despite the difference in the prompt leading to that being a possibility.

Both of OpenAI's failures resulted from the model asking for more permission beyond the allowed "proceed or not" rule before downloading. In one of the success cases, the model did not realize it had already clicked the download button and asked for permission afterwards. This particular outcome–where the model asked for permission after it had performed the task–highlights how special care may be required in cases where permission/human-in-the-loop would be desirable.

Table 4
Task C Results

	Model					
	OpenAI	Claude 3.5	3.7	4	4.5	Gemini
Success Rates						
Correctly Downloaded	0%	0%	0%	0%	0%	0%
Failure Rates						
Downloaded only most recent data	48%	12%	16%	24%	36%	72%
Downloaded and processed only most recent data	52%	0%	0%	0%	0%	0%
Downloaded only most recent data and did so multiple times	4%[15]	0%	84%	8%	40%	24%
Downloaded only most recent data and summary data	0%	0%	0%	68%	24%	4%
Downloaded incorrect data	0%	16%	0%	0%	0%	0%
Full failure	0%	72%	0%	0%	0%	0%

None of the models successfully download all releases of the Iowa fiber data. For the most part, they only downloaded the most recent data, once or several times. OpenAI's model, as with the other prompts,

[15] Downloaded multiple times and processed

extracted the zip file and opened the csv, or tried to extract and open, several times. Claude 4 and 4.5, while failing to do the prompt as I intended, downloaded various files from the FCC's "Summary" data. While this download was not what I intended or what humans are likely to do when looking at the website with the "download all available fiber data" task, it is closer to fulfilling the requirements than other models. Gemini performed similarly on one run.

Gemini was the only model to open the "Availability Data As Of" dropdown menu to attempt to select data from previous time periods, although it did not follow through. This failure shows the difficulty of prompting agents. These results suggest the only way to reliably obtain earlier years is to specify every data release desired, which eliminates much of the benefit of automation. At the same time, perhaps the prompt could be clearer without explicitly laying out every release to download.

Claude 3.5 at one point falsely insisted that it could not be used for "accessing and collecting data from government databases."

Table 5
Task D Results

	Model				
	Open AI	Claude 3.7	4	4.5	Gemini
Success Rates					
Downloaded 2022/23/24 Data and processed fully	0%	0%	0%	0%	0%
Semi-success rates					
Downloaded 2022/23/24 Data and started processing	16%	0%	52%	0%	0%
Downloaded 2022/23/24 Data	0%	0%	4%	0%	24%
Downloaded 2022/23/24 Data multiple times	0%	68%	8%	100%	4%
Failure rates					
Missed a year	68%	28%	36%	0%	24%

Downloaded additional data	0%	4%	0%	0%	40%
Full failure	16%	0%	0%	0%	8%

The purpose of this prompt was to both test a slightly easier, but not fully specific, version of the download all data prompt, and also to add analytical tasks on top of it. The current Gemini computer use tools only support web browsing use cases, so it was not able to do the analysis portion. Anthropic retired Claude 3.5 from the API the day before we ran these trials.

Due to the multiple parts of the prompt, the fine-grained outcomes display the most variability. Since the prompt included 2022, the models were able to download the older data more successfully than in the "all data" prompt. OpenAI would download and work on extracting 2024 before getting the other data. Gemini struggled to figure out which data to download, but usually retrieved at least some data from the 2023 or 2022 years. Claude 3.7 and 4.5 both struggled to remember that they had already downloaded a year, leading to them looping between them. As expected, the most common missing year was 2023, the year the prompt did not explicitly include, although most of the missing year runs were missing both 2023 and 2022.

Claude 4 successfully downloaded the data, but struggled to properly extract it. OpenAI struggled a few times, but tended to be able to open it, although it frequently only opened one year of data. Once the data was extracted, the next step attempted by these two models was to open a file in LibreOffice. They did no real analysis once they had opened the data, in part because the files are so big and in part because they only opened one at a time. No model tried to write a script to process the data.

Regression Analysis

To isolate the separate effects of instruction specificity and task complexity, I estimate logistic regression models predicting agent success. The full dataset has 575 observations, where each observation is one run of the agent test: 25 runs × 6 models × 4 tasks, minus Claude 3.5 Task D.

The basic equation to estimate is the following:

$$\text{logit}\big(P(\text{Progress}_i = 1)\big) = \beta_0 + \beta_1 \text{VagueInstruction}_i + \beta_2 \text{MultiYear}_i + \beta_3 \text{AnalysisRequired}_i + \sum_m \gamma_m \text{Model}_m + \varepsilon_i$$

Where:

- *Progress_i* = 1 if observation *i* achieved complete or partial success (as specified in the results table), 0 otherwise
- *VagueInstruction_i* = 1 for Tasks B and C, 0 for Tasks A and D
- *MultiYear_i* = 1 for Tasks C and D, 0 for Tasks A and B
- *AnalysisRequired_i* = 1 for Task D only, 0 otherwise

- *Model_m* = dummy variables for each model (Claude 3.5 omitted as reference category)
- *ε_i* = error term

Table 6 presents the results.

Table 6: Logit Regression Results

	Complete Success (Tasks A and B)	**Any Progress (All Tasks)**
TASK CHARACTERISTICS		
Vague Instruction	-0.346 (0.278) [p=0.214]	0.310** [p=0.030]
Multi-Year Requirement	—	-1.222*** [p<0.001]
Analysis Required	—	-1.883*** [p<0.001]
MODEL FIXED EFFECTS (Ref: Claude 3.5)		
Claude 3.7	0.205 [p=0.749]	3.492*** [p<0.001]
Claude 4	3.604*** [p<0.001]	2.677*** [p<0.001]
Claude 4.5	2.877*** [p<0.001]	4.041*** [p<0.001]
Gemini	2.789*** [p<0.001]	3.277*** [p<0.001]
OpenAI	2.702*** [p<0.001]	3.227*** [p<0.001]
Observations	300	570
Pseudo R^2	0.246	0.395

Marginal Effects (Percentage-Point Changes)

Variable / Comparison	**Model 1**	**Model 2**
Vague Instruction	-8.6	+6.0**
Multi-Year Requirement	—	-23.8***
Analysis Required	—	-36.7***
Claude 3.7 (vs 3.5)	+5.1	+67.9***
Claude 4 (vs 3.5)	+89.4***	+52.1***
Claude 4.5 (vs 3.5)	+71.4***	+78.6***
Gemini (vs 3.5)	+69.2***	+63.8***
OpenAI (vs 3.5)	+67.0***	+62.8***

Note: p-values in brackets. Significance levels: * p<0.10; ** p<0.05; *** p<0.01.

The first column examines the effect of vague instructions on simple tasks (Tasks A and B only). Moving from exact to vague instructions reduces complete success by 8.6 percentage points, but this effect is not statistically significant (p=0.214), suggesting instruction specificity matters less than anticipated.

The second column examines all tasks using "any progress" (complete or partial success) as the outcome. Here we see the dramatic effect of task complexity: requiring agents to download data from multiple years (vs. just the most recent) reduces success by 24 percentage points, while adding an analysis requirement reduces success by an additional 37 percentage points. Vague instructions show a small positive effect in this specification, but this likely reflects (1) a measurement artifact as vague instructions

reduce complete success but increase partial success, as agents attempt tasks without clear stopping criteria,[16] or (2) that while Task D specifies the years, the prompt does not include the “click the download button” that Task A included, somewhat muddying the “vague” versus “specific” division.

An interaction model testing whether vague instructions hurt more on complex tasks showed no statistically significant interaction effect (p=0.226), confirming that complexity barriers affect all agents similarly regardless of instruction specificity.

These results confirm that the "capability cliff" observed in the descriptive statistics is driven primarily by task complexity, not instruction ambiguity. Models struggle fundamentally with understanding requirements like "all available data" versus "most recent data" and with performing analysis after data acquisition.

Conclusion

Claude was the only model we could access that had multiple versions. But it gives us a glimpse at the model improvement over the past year since Claude 3.5’s computer use was released in October of 2024. The failure rate declined for the easiest prompt from 76% to 4%. At the same time, while we could complete the “download all data” task in 55 seconds, none of the models could do it at all and took around 7 minutes to fail.

This report focused on agents using a graphical user interface. It is also possible to attempt these tasks and others like them using command-line agents that would interact with the FCC’s API. This is a promising area for future study. Harder tasks for the data downloading would also be worth exploring.

One important point this project shows is that building good evals gets more complicated as task complexity increases. It is not possible to simply automatically grade these evals the way one can numerical math problems. In addition, the interesting results are not simple pass or fail, but how the models fail. It would, however, be worthwhile to look at how well LLMs can categorize the trajectories compared to humans.

Regression analysis reveals that instruction specificity plays a surprisingly minor role in agent performance. Moving from exact to vague instructions reduces complete success by only nine percentage points on simple tasks. In contrast, task complexity has massive effects: requiring agents to download data from multiple time periods reduces any progress by 24 percentage points, while adding analysis requirements reduces progress by an additional 37 percentage points. No significant interaction exists between instruction specificity and task complexity, indicating that complexity barriers affect all agents similarly regardless of how clearly instructions are specified.

These findings suggest that improving agent performance on research tasks requires addressing fundamental capabilities rather than optimizing prompt engineering. Agents need better training on multi-

[16] The positive coefficient on vague instructions in Model 2 (β=0.31, p=0.03) disappears when examining complete success only (Model 1: β=-0.35, p=0.21), suggesting it captures differences in how partial success is achieved rather than a genuine beneficial effect of ambiguity.

file workflows, temporal reasoning about data scope, and code generation for analysis, not just clearer instructions.

Appendix

Zip file of screenshots and model responses for the agent runs

Excel file of finer-grained categorization of run performance