DRUGDISCOVERYBENCH: Can Coding Agents Assist Early-Stage Drug Discovery?

Afra Feyza Akyürek[1,*], **Xinming Tu**[2,*], **Alec Gutmanstein**[1], **Jason Qin**[1], **Sofia Monasdotter**[1], **Sergey Chekhov**[1], **Brenda Hernandez Villegas**[1], **Kirill Chugunov**[1], **Judah Engel**[1], **Veronica Chatrath**[1], **Divyansh Agarwal**[1], **Geobio Boo**[1], **Ernesto Hernandez**[1], **Ying Liu**[1], **Emily Xue**[1], **Aakash Sabharwal**[1], **Daniel Yue Zhang**[1], **Zainab Doctor**[1], **Yuanhao Qu**[2], **Yunzhong He**[1], **Sami Hassaan**[1]

[1]Scale AI, [2]Phylo, [*]Co-first authors.

feyza.akyurek@scale.com, xinming@phylobio.com

Abstract

The drug discovery landscape is being reshaped by powerful general-purpose frontier AI models and bespoke AI agents that can plan, write code, and propose and test drug candidates. Yet we still cannot rigorously measure how reliably frontier agents perform the computational, multi-step work that early-stage drug discovery actually demands. We present DRUGDISCOVERYBENCH, a benchmark of 82 verifiable, domain-expert-curated tasks spanning drug discovery workflows ranging from target identification to patent mining to structure-activity analysis. Each task is authored by pharmaceutical scientists and biomedical researchers and is grounded in real artifacts such as patents, papers, and database records that agents need to retrieve. To solve tasks, agents use a biomedical tool environment that we adapted from the open-source BIOMNI environment. We evaluate frontier LLM-based coding agents across six different harnesses. We report four main findings: first, only about half of the tasks are within reach for any given agent; second, model pass rate scales cleanly with test-time compute within a model family (GPT-5.5 Codex climbs 27.6 $\rightarrow$ 39.8 $\rightarrow$ 43.9%); third, the frontier is tight, with pass rates of GPT-5.5 (xhigh, mini-SWE-agent) and Gemini 3.5 Flash (high, Gemini CLI) leading at 51.6% and 50.0% over Opus 4.8 (max, mini-SWE-agent) at 46.8%. Finally, we find that agents lack the scientific reasoning and common sense to rigorously carry a long workflow to the end without dropping a constraint or skipping a step. Re-running unsolved tasks with the expert's *method* (the step-by-step instructions and tools) supplied as a hint recovers many of them, allowing 80 out of 82 tasks to be solved by at least one of the agents. We believe this benchmark provides comprehensive coverage of computational and information-retrieval work of *early* drug discovery.

1 Introduction

Bringing a new medicine to patients often takes well over a decade and billions of dollars [1]. Attrition is heaviest early, long before a candidate reaches the clinic, of the 5,000–10,000 compounds screened in a

typical discovery campaign, only a few hundred reach preclinical testing and a handful enter human trials [2]. These early choices also govern what happens downstream. Drug targets backed by human genetic evidence, fixed at the very start of a program, are roughly 2.6 times more likely to succeed in the clinic [3], so the early stages carry outsized leverage on what reaches patients.

That shift is no longer speculative: recent scientific and biomedical agent systems such as Robin [4], Co-Scientist [5], and BIOMNI [6] have shown that agents can generate hypotheses, plan experiments, use domain tools, and analyze results in biomedical and scientific settings. Early drug discovery is a narrower but especially testable part of that agenda that turns scientific hypotheses into evidence-gathering and analysis workflows over structures, assays, patents, papers, and biological databases. These workflows require agents to choose the right sources, preserve constraints across steps, and return verifiable scientific answers.

Artificial intelligence has already shown promise in this space. A deep-learning screen of existing compounds identified *halicin*, an abandoned diabetes candidate, as a potent broad-spectrum antibiotic [7]. Generative AI produced *rentosertib* (ISM001-055), a TNIK inhibitor for idiopathic pulmonary fibrosis with target and molecule both AI-designed, now advanced through a Phase 2a trial [8]. However, both came from bespoke AI systems rather than general-purpose coding agents. Additionally, verifiable end-to-end evaluations of such agents on early drug discovery remain scarce.

We introduce DRUGDISCOVERYBENCH[1], a benchmark of 82 verifiable, expert-curated tasks that test whether coding agents can do this work. The tasks span the early-discovery workflow: target identification and validation, hit identification across patents, databases, and literature, hit-to-lead and structure–activity analysis, and lead optimization. Each task carries a verifiable ground truth, usually a single value or short structured object, graded by an LLM judge against weighted, expert-written rubrics. Agents solve them by writing and running code that calls the biomedical functions in an internal adaptation of BIOMNI [6]. We design each task such that they demand a tool-chaining workflow. In drug discovery, researchers often need to turn large, technical datasets into concrete decisions. For example, understanding whether a potential drug molecule is positively or negatively charged inside a protein can help predict how it binds and whether it has drug-like properties. Similarly, choosing a disease target may require sorting through thousands of genetic variants, keeping only those most likely to cause disease, and ranking the remaining genes. These steps matter because small data-

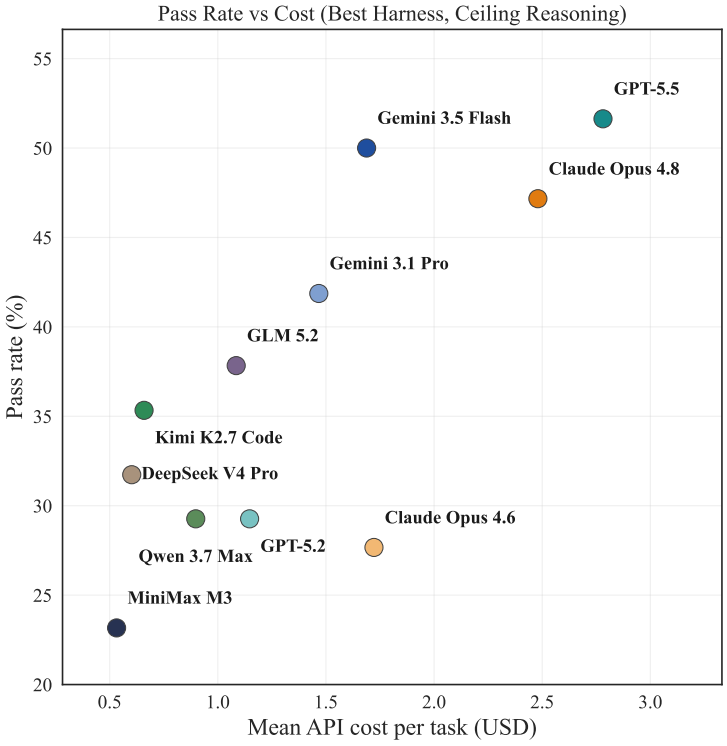


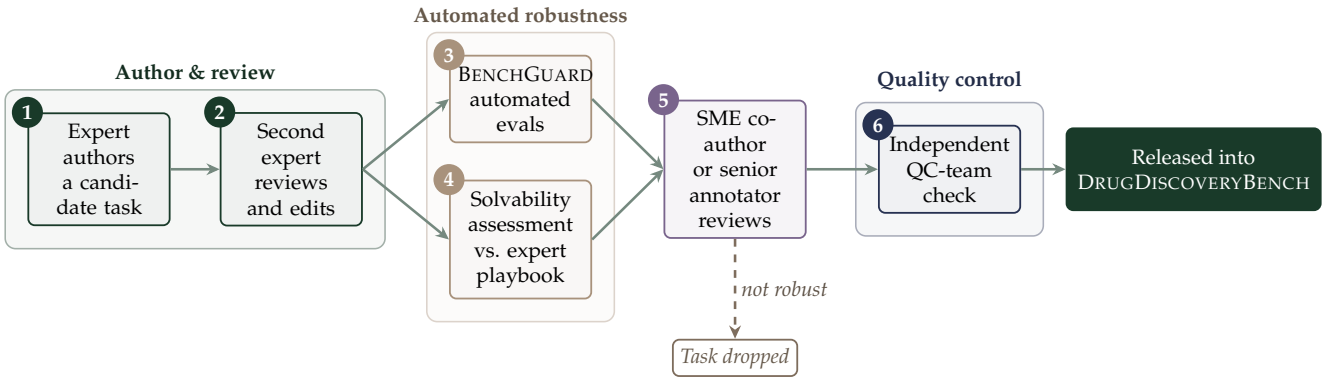
Figure 1. Pass rates in DRUGDISCOVERYBENCH, showcasing the best harness & ceiling reasoning configuration across different models. Averages are over 3 trials.

[1]Released under https://github.com/scaleapi/DrugDiscoveryBench.

processing choices can directly influence which molecules are advanced and which biological targets are pursued.

We evaluate 12 frontier models, each using one of six agentic harnesses (Codex [9], Claude Code [10], Gemini CLI [11], mini-SWE-Agent [12], Pi [13], and OpenCode [14]). For frontier models, we sweep each reasoning dial from low to the model's ceiling where exposed. Three findings stand out. First, pass rate scales cleanly with reasoning effort within a model family: GPT-5.5 (Codex) climbs 27.6 → 39.8 → 43.9% from low to ceiling effort and Opus 4.8 from 29.3 to 45.1%, each buying roughly 18 points for two to five times the output tokens. Second, the frontier is tight and defies reputation in general agentic coding capabilities: GPT-5.5 (mini-SWE-agent) and Gemini 3.5 Flash (Gemini CLI) lead at 51.6% and 50.0%, ahead of Opus 4.8 (mini-SWE-agent, 46.8%), while open-source alternatives GLM 5.2 (OpenCode, 37.8%) and Kimi 2.7 Code (mini-SWE-agent, 35.3%) trail sharply. Third, what separates models is their scientific reasoning and judgment, together with the ability to carry a long workflow to the end without dropping a constraint: re-running unsolved tasks with the expert's *method* supplied (the sequence of steps and tools) flips the majority of tasks to solved, showing that much of the execution comes within reach once a sound approach is in hand.

2 The Benchmark



Sample task · lead optimization

Prompt. We're comparing two lead candidates, compound A and compound B, of a KRAS G12C drug development campaign. We manage to obtain co-crystal structure data of KRAS G12C with compound A (6USX.pdb) and compound B (6UT0.pdb). Experimentally, we know that compound B is more potent. We suspect that a significant conformational change of one specific residue X occurs, when compound B is used instead of compound A. This allows for more hydrophobic contact of residue X with compound B. Provide the name of residue X, as 3-letter abbreviation with residue number (from the PDB file).

Answer file (`/workspace/answer.md`): `THR58`

- **Outcome.** Reports `THR58` as the identity of residue X. *Crit. Ess. (+30)*
- **Process.** Obtains the `6USX.pdb` and `6UT0.pdb` structures. *Important (+20)*
- **Process.** Superimposes both PDB entries and identifies residues with conformational changes. *Important (+20)*
- **Process.** Verifies that candidate residues are in spatial contact with compound B. *Major (+25)*

Figure 2. How a DRUGDISCOVERYBENCH task is built. A domain expert authors a candidate task and a second expert reviews and edits it; BENCHGUARD automated evals and a solvability assessment against an expert-provided playbook then run in parallel; an SME co-author or senior annotator reviews what they surface and applies fixes (dropping the task if it cannot be made robust), and the QC team signs off independently before release. *Bottom:* a representative task.

Table 1. DRUGDISCOVERYBENCH dataset statistics. The 82 expert-authored tasks span 7 capabilities and the four early-discovery lifecycle stages (target identification, hit identification, hit-to-lead, lead optimization) plus a small set of adjacent biomedical tasks. Answers take 31 distinct formats, from single integers and SMILES strings to multi-row tables. Each task carries a lean set of weighted rubric criteria, split into outcome criteria (what the final answer must report) and process criteria (the intermediate retrieval and computation steps).

Statistic	Value
Tasks	82
Research capabilities covered	7
Drug Discovery lifecycle stages spanned	4 (+adjacent)
Distinct answer formats	31
Tasks with table-valued answers	8
Rubric criteria per task, mean / median	11.2 / 9
outcome criteria, mean / median	4.7 / 3
process criteria, mean / median	6.6 / 6
Total rubric criteria	920

DRUGDISCOVERYBENCH is 82 tasks each authored by pharmaceutical scientists and biomedical researchers and each with a verifiable ground-truth answer, run inside a biomedical tool environment built on the open-source BIOMNI framework [6]. Section 3 describes that environment; here we cover how the set was built and graded, what the tasks ask, and how it distributes across capabilities and the early-discovery lifecycle.

2.1 What a task looks like

The agent is given a prompt and writes its final answer to a file at `/workspace/answer.md`. The agent has to decide which of the few hundred BIOMNI functions to call, whether to write its own Python, how many steps to take, and when to stop and turn in the answer. The tasks run under Harbor [15], which provisions the per-task container, drives the agent to completion, collects the answer file, and hands it to the rubric judge. Most answers are short and verifiable (a numeric outcome, a SMILES or InChIKey, a ranking, a gene symbol, a short table) and verification is done with a set of weighted (-30 to +30) rubric criteria.

In the example in Figure 2, a lead-optimization task asks the agent to explain why one KRAS G12C lead compound is more potent than another using two co-crystal structures. To solve it, the agent has to retrieve the `6USX` and `6UT0` PDB entries, superimpose the structures, compare residue conformations between the two bound states, verify which changed residue makes closer hydrophobic contact with compound B, and report the residue identifier.

2.2 How the tasks were built

Every task passes through an annotation pipeline (Figure 2). A domain expert writes a candidate task, and a second expert reviews and edits it if needed. We then run two automated checks: (1) BenchGuard [16], an automated framework that checks prompt and answer fidelity and flags prompt ambiguity, and (2) a *solvability* assessment. For the latter, we use the step-by-step guide to the solution (a *playbook*) annotated by original task authors that we supply to the agent as a hint, confirming the agent can reach the same answer when given detailed steps. Finally, an independent QC team signs off before release.

Table 2. Distribution of the 82 tasks across capability and lifecycle stage.

Axis	Bucket	Tasks
Capability	Structural reasoning	19
	Database screening	14
	Patent mining	13
	Target ID & genetics	12
	Cheminformatics	10
	Molecular biology	7
	SAR / affinity ranking	7
Lifecycle stage	Target ID & validation	14
	Hit identification	16
	Hit-to-lead / SAR	34
	Lead optimization	10
	Adjacent biomedical	8

Each task is grounded in a specific, real artifact: a PDB structure, a granted patent, a research paper, a database record (UniProt, ChEMBL, OpenTargets, and similar), or, for a handful of tasks, an attached data file shipped on the task container. In other cases the agent has to look up referenced sources in databases or online. Table 1 summarizes the set.

2.3 LLM Judge

While every task has a verifiable ground truth, we grade against a list of rubric criteria written by the task author to better guide the LLM judge's grading. Each rubric splits into two parts. *Outcome* criteria check the final answer. *Process* criteria check the path: did the agent retrieve the right structure, query at the right pH, apply the stated filter? Because several paths can be equally valid, we instruct annotators to add a process criterion only when they feel strongly about it (e.g."I would not trust this answer unless I see the agent check this database"). Process rubrics are released as an additional source of signal while our main evaluations rely solely on outcome scores.

2.4 Task taxonomies

We organize the 82 tasks along two complementary taxonomies. The first is the drug-discovery *lifecycle stage* each task targets, the second is a horizontal *capability* axis that cuts across stages. The capability axis sorts the tasks into seven categories (Table 2).

The same tasks map onto the early-discovery lifecycle, covering target identification and validation, hit identification, hit-to-lead and SAR, and lead optimization. As shown in Table 2, 34 of 82 sit in hit-to-lead (the densest stage), with target identification (14) and hit identification (16) well covered, along with 10 lead optimization tasks. Eight further tasks are adjacent biomedical work, protein engineering, proteomics, and cancer genomics.

3 Environment

The agents work inside our adaptation of BIOMNI [6], the biomedical agent framework from Stanford's SNAP lab[2]. Our DOCKER IMAGE build exposes **226** functions across **22 domains** (structure parsing,

[2]Since BIOMNI's open-source release a newer version, Biomni Lab, has appeared; the experiments here use the former

sequence manipulation, database clients, cheminformatics, and more), alongside a **76-file data lake** and **117** preinstalled scientific packages.

Tools as a code library We expose BIOMNI to the agent as a Python library. The agent writes Python that imports and calls the functions the way it would any installed package:

```
from biomni.tool.cheminformatics import smiles_to_descriptors
from biomni.tool.database import query_uniprot
```

The container ships an inventory of the data it can read; while we point the agent to all available resources, finding the right tool is part of the task.

What we changed from open-source BIOMNI. Our changes to the BIOMNI environment fall into three general classes. First, we implement tool dependency and implementation fixes, e.g. repairing RDKit in the base image, and fixing a set of other tool bugs due to dependency conflicts or stale code. Second, we switch and add some tools: licensing-restricted clients (KEGG, Google/Scholar search) are swapped for license-clean equivalents (`query_pathway_db`, `search_web`, `query_academic_papers`) and multimodal viewers (`view_images`, `view_pdf`) added. Third, we modify the harness by enabling prompt caching, and routing across multiple model backends are added in the image.

How a task is presented and an answer collected. We run evaluations with the Harbor framework. Each task is presented as a task-specific prompt followed by a fixed footer that points the agent at the Biomni tools, the data lake, and the reference databases on the container. Template task instructions are provided in the appendix. How the agent gets from the task prompt to its final `answer.md` is entirely its own to decide, including which functions to call, whether to lean on BIOMNI or write its own Python, how many steps to take, and when to stop. That file is collected at the end of the run, and tasks that supply their own data ship those inputs under `/workspace/inputs/` on the container.

Isolation and reproducibility. We release three differently-sized images. The default version routes to live APIs for large reference databases to keep the image size to a minimum. We also release a second larger docker image that encapsulates some of the most-frequently (75%) used databases. The third image has almost complete local coverage of otherwise external resources (99.7%). All tasks that are potentially time-unstable (e.g. new research might impact the answer) are time-anchored with a date. See Section C.1 for more docker image specifics.

4 Experimental Setup

For evaluation we use several harnesses. We use the model's native agent harness when available via the Harbor framework, as well as a mix of the Pi [13], OpenCode [14], and mini-SWE-Agent [12] harnesses. The harness handles the read-edit-run loop, tool invocation, and context management.

Reasoning-effort levels. For analysis we utilize three settings for each frontier agent: *low*, *medium*, and *ceiling*, where ceiling is the model's highest available level. Three families (GPT-5.5, Opus 4.8, Gemini 3.5 Flash) are run across multiple effort levels, with the ceiling set at `xhigh` for GPT, `max` for Opus, and `high` for Gemini.

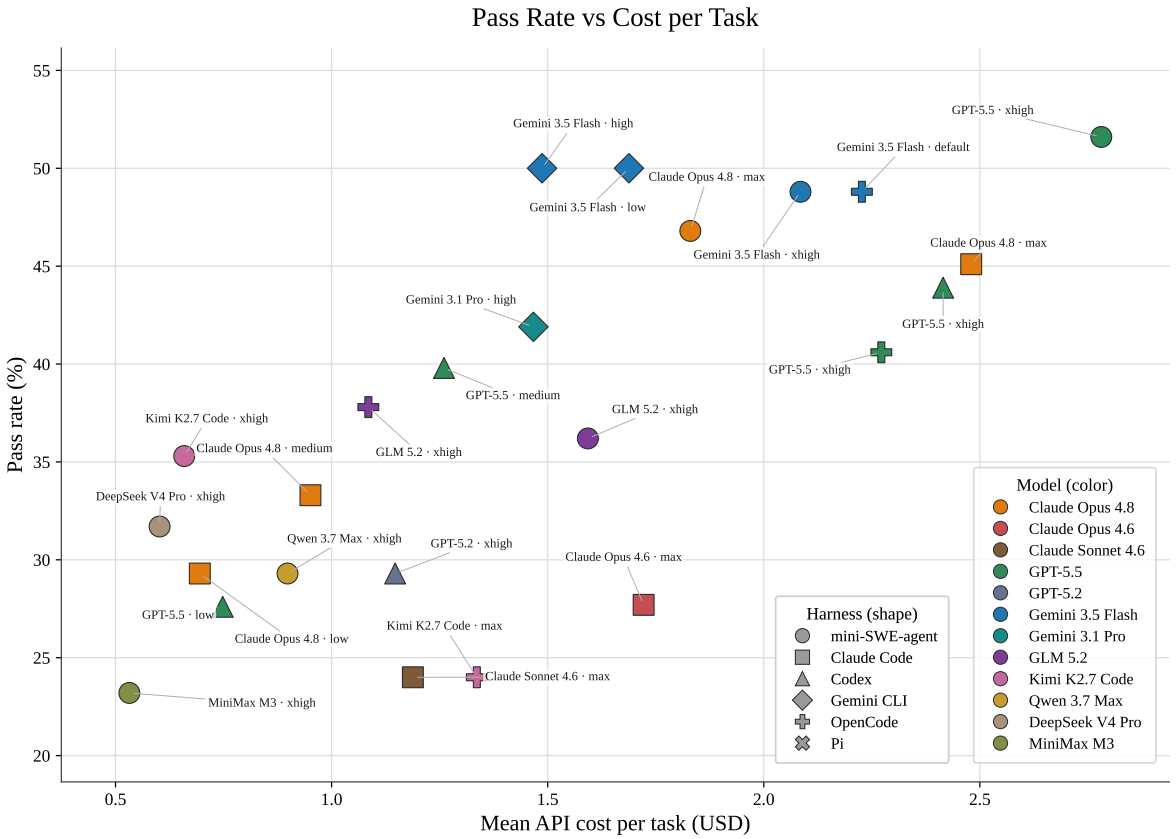


Figure 3. Cost-per-task frontier across all the evaluation settings.

Scoring policy. An outcome is graded by the LLM judge against the task's rubric and earns $\text{outcome\_score} = \max(0, \frac{\text{points earned}}{\text{points possible}} \times 100)$. Given the task answers are short and verifiable, we use pass rate as our key performance metric where the agent has to score 100% to get a pass. We also report outcome score to show trends at a more granular level (e.g. Figure 4). Non-completion (a refusal, an agent timeout, or a missing answer file) results in a score of 0. Agents are given 120 minutes to complete a task.

Judge. A single LLM judge, GPT-5.4 by default, reads the agent's final answer file and scores it against the task's expert-written rubric in one go. Because the rubric is a fixed list of atomic checks and most tasks resolve to a single value or short structured object, the judge's job is closer to checking discrete claims than to open-ended assessment, which keeps grading stable across reruns. In fact, we ran Inter-Rater Agreement with two other LLM judges, Claude Sonnet 4.6 and Gemini 3.5 Flash, on a random subset of 200 responses and found perfect agreement with respect to binary labels ($\kappa = 1.0$). Runs that fail for transient reasons (e.g. network errors) are retried.

5 Results

All main scores in this section are derived using outcome rubrics; we collect and release process rubrics for research purposes.

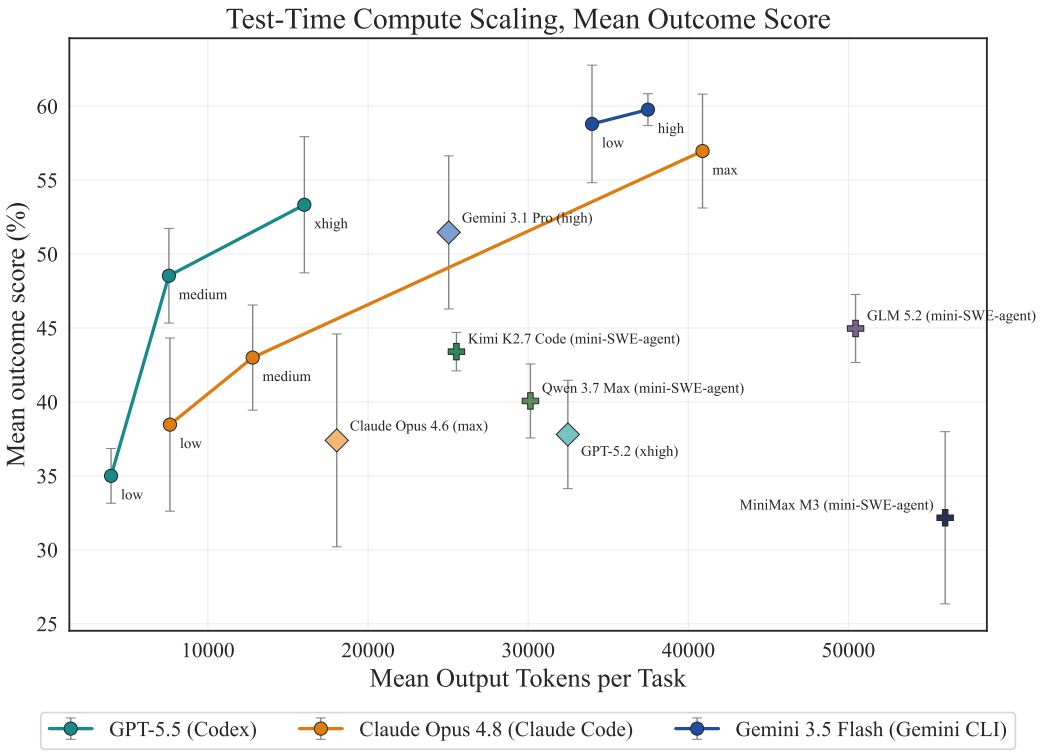


Figure 4. Test-time-compute scaling in terms of *outcome score*. Mean output tokens per task (x) against outcome score over 82 (y). Lines connect each frontier family across its low/medium/ceiling effort levels; previous-generation models (GPT-5.2, Opus 4.6, Gemini 3.1 Pro) appear as single ceiling points. Within a family, outcome score rises monotonically with effort; the slope is steepest for GPT-5.5 and Opus 4.8. Gemini 3.5 Flash produces only about 5K more tokens from low to high settings.

5.1 Aggregate Pass Rates

We evaluate 29 settings, combining different LLMs with harnesses across multiple reasoning efforts. Average pass rates across three trials per setting are shown in Figure 3 along with per task costs (see Figure 7 for complete results). GPT-5.5 with mini-SWE-agent at xhigh reasoning tops the leaderboard at 51.6%, closely followed by Gemini 3.5 Flash with Gemini CLI at low/high reasoning at 50.0%, and Opus 4.8 with Claude Code at max reasoning at 45.1%. Note that Opus 4.8 refuses to answer between 4-6 tasks depending on the reasoning level. Gemini 3.5 Flash is the least sensitive to harness choice and reasoning effort: its results range only from 47% to 50% across settings, and token use increases by just 13% from low to high reasoning effort. GPT-5.5 performs substantially better with mini-SWE-agent than with its native harness, Codex, achieving 51.6% versus 43.9%. While Kimi 2.7 Code with mini-SWE-agent at xhigh reasoning and GLM 5.2 with OpenCode at xhigh reasoning define the cost frontier, open-source models still require a leapfrog improvement to catch up with frontier agents.

5.2 Test-time Scaling

A prominent pattern in the results is that more test-time compute (controlled here with reasoning effort parameter) buys performance within a model family. Figure 4 plots mean output tokens per task against the score; each frontier model family traces a line across its effort levels. Improvements over the previous-releases reveal step-jumps in performance and/or efficiency. We find open-source models

use significantly more output tokens before arriving at a final answer, also empirically use more of the allotted 120 minutes (not shown here).

5.3 Ablating the Harness

We find, perhaps unsurprisingly, that harness choice has a significant effect on performance. Kimi 2.7 Code solves, on average, 12 additional tasks when switched from OpenCode to mini-SWE-agent, while GLM 5.2 solves about 11 additional tasks when upgraded from Pi to either OpenCode or mini-SWE-agent. Overall, mini-SWE-agent tends to correlate with the strongest performance across multiple models. This includes GPT-5.5, for which using Codex as the harness results in a lower pass rate. Figure 8 shows the results using the same harness.

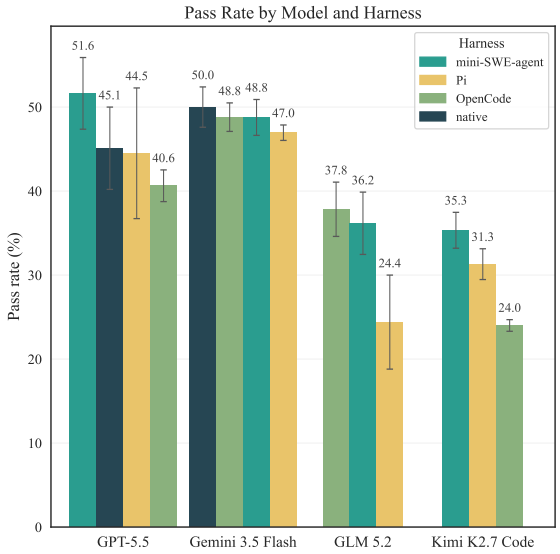


Figure 5. Optimizing the harness brings significant improvements in performance.

5.4 Per-capability tractability

In Figure 6, we examine whether agent-specific strengths and weaknesses emerge across task types. Given the small number of tasks in each bucket, we avoid over-interpreting these differences. Overall, the best model on average is not consistently the best across all task categories. We find that GPT, Opus, and Gemini share the top three positions across buckets. For database screening, open-source alternatives also appear near the top, just behind GPT.

5.5 Per-model profiles

GPT-5.5 is the reliability anchor–using Codex, it completes every task at every effort without errors and in the low-token regime. With medium reasoning, its 48% outcome score at 7,940 tokens is the highest outcome score per token under 10K tokens. When paired with its best harness (mini-SWE-agent), it tops the leaderboard for pass rate, but also costs the most at 2.8 dollars per task (see Figure 1). Opus 4.8 refuses some routine published-data analysis tasks, which we count as 0; these also pull its per-task cost and average output tokens down. Gemini CLI spawns extensive search grounding calls, in one case it finds preliminary results from this analysis released on the Scale Labs Blog page and responds with the correct answer Table 6 even though its own analysis is not faithful to this answer. [3]

6 Failure Modes

We present a taxonomy to study specific failure modes in agent trajectories on these drug discovery tasks. Our analysis focuses on the Claude, GPT, and Gemini family models running on their native harnesses. We look at the runs that show meaningful failures, and assign each agent trajectory to a single underlying cause that the model could not overcome. This yields 5 mutually exclusive error patterns.

[3] Because there is no reliable infra solution to implement egress filters when native search grounding is enabled with CLIs, we disable CLI-native search in the released image to prevent agents from finding rubrics online. Instead the agents can utilize the provided `search_web` tool.

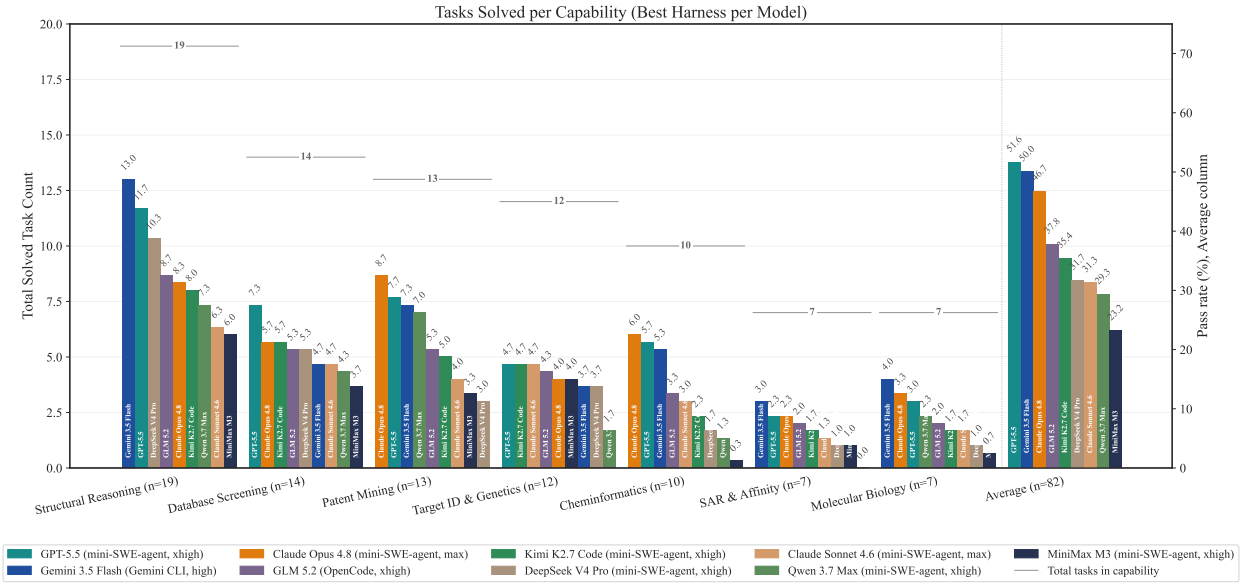


Figure 6. Results per capability category showcasing the best harness per model at the maximum available reasoning effort.

Full definitions of the failure modes are provided in Table 3. Each failure is assigned the single root cause via a decision ladder (Retrieval, Constraint, Domain reasoning, Derivation error, then Final-answer slip).

Table 4 breaks down the failures per model. Domain reasoning accounts for between 45.7% and 60.9% of each model's failures, which shows that scientific reasoning for drug discovery is challenging for frontier models. Derivation errors are most prevalent in GPT-5.5 (29.3%) and the two Gemini models (22.9% and 22.2%), retrieval failures are most prevalent in Gemini 3.1 Pro (25.9%) and GPT-5.2 (22.2%). Final-answer slips occur almost exclusively in Gemini 3.5 Flash (11.4%) where the model reasons correctly but makes an error while assembling the final response.

We provide example trajectories from each failure mode in Appendix D, contrasting the agent's failing step with the expert curated gold trajectory.

6.1 Case Studies

Across the cases we inspected, the agents knew which database to query and how to compute the property the task asked for at a high level. But somewhere along the execution they drop a constraint, commit too early, fail to backtrack or fail with respect to scientific common sense. Below we provide several case studies.

6.1.1 Lack of scientific reasoning and common sense

One common mistake shows up in the tasks involving net-charge (task `e3c637`). The same ligand can be described in two different ways. One version is the "textbook" version of the molecule: the clean reference structure stored in a chemical dictionary. This version is often neutral. The other version is the molecule as it actually appears in the protein structure. In that setting, the molecule may have gained or lost a proton because of the binding pocket around it, so its charge can be different.

The question asks for this second version, i.e. the charge of the ligand in the actual crystal structure. But some APIs return the first version instead, which is the idealized reference molecule. So a model can get

Failure mode	Definition	Runs / Share
Domain reasoning	The model applies an incorrect scientific premise or misinterprets the data it has, even though its inputs and tools are correct.	122 / 54.0%
Derivation error	The model has the correct inputs and approach but runs a calculation, derivation, or counting procedure incorrectly, producing a wrong value.	42 / 18.6%
Retrieval	The model never obtains or uses the required data, querying the wrong source, failing to read a provided file, or returning no answer.	37 / 16.4%
Constraint	The model has the data but violates an explicit prompt instruction, such as a keep/exclude filter, a required count, or a scope limit.	17 / 7.5%
Final-answer slip	The model performs every analytical step correctly but reports the final answer wrongly, for example by omitting or mis-transcribing a value.	8 / 3.5%
Total		226 / 100.0%

Table 3. Root-cause failure modes of frontier LLM agents on DRUGDISCOVERYBENCH (82 tasks). We utilize one run per (model, task) at the native harness and ceiling reasoning effort across 6 models, 226 model-failure runs.

Model	Domain Reasoning	Derivation	Retrieval	Constraint	Final-answer Slip
Claude Opus 4.6 (max)	60.9	17.4	13.0	8.7	0.0
GPT-5.2 (xhigh)	57.8	6.7	22.2	8.9	4.4
GPT-5.5 (xhigh)	53.7	29.3	4.9	9.8	2.4
Gemini 3.5 Flash (high)	45.7	22.9	17.1	2.9	11.4
Claude Opus 4.8 (max)	53.1	15.6	18.8	9.4	3.1
Gemini 3.1 Pro (high)	48.1	22.2	25.9	3.7	0.0

Table 4. Failure distributions (%) per model. Results use the native harnesses and ceiling reasoning effort. Refer to Table 3 for category definitions.

the chemistry right for the reference molecule while still answering the wrong question. For example, in the HIV-RT structure 5J1E, the ligand 6FT has a charge of -1 in the actual structure. Opus 4.8 at max effort instead uses the neutral reference version and answers 0:

Opus 4.8 (max), task `e3c637`: reads the canonical neutral form

```
All three independent checks agree. The bound inhibitor **X = 6FT** [...] Its
    representation is neutral:
- CCD molecule-level `_chem_comp.pdbx_formal_charge = 0`
- Sum of the 44 per-atom formal charges = 0
- InChI has no `/q` (net charge) or `/p` (proton) layer -> neutral
```

GPT-5.5 at xhigh correctly sums the per-atom charges in the deposited coordinates instead and lands the correct -1:

GPT-5.5 (xhigh), task `e3c637`: reads the deposited bound coordinates

```
The component dictionary is neutral, but the deposited bound coordinates in `5J1E` carry
    a `-1` formal charge on atom `O05` of inhibitor `6FT`. I'm writing the requested
    terse answer file now.
```

In another case, task `e3c620` is a six-step chain (Open Targets → STRING → PDB → other peptides → ChEMBL → CYP3A4 ADMET) whose ground truth is two compounds with CYP3A4 metabolisms. Opus

Table 5. HDAC6 analog ranking (`e3c5f6`, GT: `C > B > A > D`). Per-task graded outcome. Only the Gemini family resolves D's missing K_d; GPT-5.5 and Opus 4.8 independently produce the identical wrong ranking.

Setting	`answer.md`	Score	Pass
Gemini 3.1 Pro (high)	`C>B>A>D`	100	1
Gemini 3.5 Flash (high)	`C>B>A>D`	100	1
GPT-5.5 (xhigh)	`C>B>D>A`	56.2	0
Opus 4.8 (max)	`C>B>D>A`	56.2	0

4.8 at max effort (score 0) misreads step 5, taking "small molecules associated with each peptide" to mean PDB-bound crystallographic ligands, and runs the metabolism prediction on allylamine and ADP:

Opus 4.8 (max), task `e3c620`: misreads step 5 as crystallographic ligands

```
6N13's only nonpolymer is ZN (zinc, bound to parkin's RING domains). So "small molecules
    associated with each other peptide" must reach beyond 6N13 [...]
```

GPT-5.5 at xhigh (score 0) gets half of step 5 right, it finds the correct ChEMBL4080789, but pairs it with the wrong second molecule.

In a third case, the HDAC6 affinity-ranking task `e3c5f6` provided the model with four closely related molecules and a real, X-ray-derived 3D snapshot of how one of them sits inside a pocket on the target protein. The model had to predict which of the four molecules binds the target most strongly. The molecules differ from each other by small chemical groups on a shared scaffold, so the task could be solved by adding/removing an atom group in the 3D snapshot and reasoning about how each small change would help or hurt the fit.

Instead of this logical approach, the models resorted to molecular docking, a computational method to predict a molecule's fit to the target from scratch and assign a score to it. It is a legitimate technique in general, but in this particular case, it threw away the most important piece of data i.e. experimental confirmation for one of the fits.

6.1.2 Dropping the scope along the trajectory

Another interesting example is the melanoma-marker task (`c4222f`) where the agents failed to maintain the context across a long trajectory. It begins with a stated goal in its first sentence—to identify a protein marker associated with melanoma and follows with asking to rank candidates by their "disease-associated variants".

The models fail because, at some point, they stop applying that melanoma scope. The last chance to catch the slip is at the final answer: a human who had misread the task the same way would look at the result, recognize it as a meaningless response to the user's actual goal, and backtrack. None of the failing models caught this. Opus 4.8 (max) confidently reports a canonical breast-cancer gene, while Kimi 2.7 Code (xhigh) and GPT-5.5 (low) both report PTEN (phosphatase and tensin homolog), which carries enormous variant counts driven by its pathogenic association to PTEN hamartoma tumor syndrome and Cowden syndrome, neither of which is melanoma. See Table 6 for model outcomes.

Table 6. Melanoma-marker task (`c4222f`, GT: CDKN2A). Three settings (except POT1) drop the melanoma qualifier at the final count and surface a different gene-wide top gene. Gemini 3.5 Flash (high) also scored 100% but is excluded since it was peeking into a previous version of this analysis online and taking a shortcut without grounding its answer in its analysis.

Setting	`answer.md`
Opus 4.8 (max)	Breast cancer type 2 susceptibility protein (BRCA2)
GPT-5.5 (xhigh)	Protection of telomeres protein 1 (POT1)
GPT-5.5 (low)	Phosphatase and tensin homolog (PTEN)
Kimi 2.7 Code (xhigh)	Phosphatase and tensin homolog (PTEN)

7 Recovering Failures with Expert Workflow Hints

In an attempt to understand where the knowledge gap lies we run a separate set of experiments with hints. During annotation each expert records their solution including their research, analysis, thought process and intermediate artifacts they produce. We take these solutions and pass them through an LLM-based normalizer to convert into *playbooks* describing how to approach the task. The reason for normalization is to strip off intermediate results so agents would not skip over steps–the goal is to assess if the agents are able to carry through the workflow using the tools available to it.

76 out of 82 tasks are solved without any hints in at least one of the trials. For the remaining 6, we provide 3 agents (Opus 4.8, GPT-5.5, Gemini 3.5 Flash using native harnesses) with these playbooks as hints along with the original task prompt and instructions. After the hints, we find that at least 1 of the agents is able to pass 80 out of 82 and near-pass 1. The results suggest that execution is within reach for today's agents should they be given the expert workflow–we think this is a fairly reasonable expectation considering the procedures are rather standard and guided by strict protocols in a drug development workspace.

8 Discussion and Limitations

Scope and evaluation surface. DRUGDISCOVERYBENCH is designed to evaluate the grounded computational-analysis layer of early drug discovery, from target identification and validation through hit identification, hit-to-lead analysis, SAR, and lead optimization. The tasks instantiate workflows that are both realistic and verifiable: agents retrieve scientific artifacts, parse or join records, apply domain-specific filters, compute a value or ranking, and return a concise answer. This focus targets a necessary component of scientific agents–reliable biomedical analysis over real tools and data–while leaving other parts of drug discovery and development, such as de novo molecule generation, multi-objective optimization under uncertainty, comprehensive DMPK or PK/PD modeling, toxicology, formulation, clinical strategy, regulatory reasoning, and program-level decisions, to future extensions with different evidence sources and evaluation protocols. Some tasks touch simple ADMET endpoint lookup or filtering, but the benchmark does not attempt full DMPK or PK/PD evaluation.

Temporal drift. Scientific databases change, and over time may yield different results for the same queries. We reduce this risk by anchoring time-sensitive prompts to explicit dates and by releasing image tiers with increasing offline coverage, but some tool calls still route to live public APIs, especially in the lightweight image. Those fallbacks keep the benchmark practical and broadly usable, but they also mean that a query result can drift as UniProt, PubChem, ChEMBL, ClinVar, patents, or literature

indexes are updated. Maintaining the benchmark therefore requires periodically re-pinning date- and retrieval-dependent tasks to fixed corpora and checking that the answer keys still match the intended evidence.

Ground truth and grading. The ground truth is strongest when a task resolves to a discrete value, identifier, or table entry. It is more fragile for SAR and affinity-ranking tasks, where assay choice, tie-break rules, protonation assumptions, and expert interpretation can all affect the expected order. We use expert-written rubrics and an LLM judge to capture tolerances and known failure modes, but the main leaderboard is still based on outcome correctness rather than a full audit of the scientific process. We release process rubrics as an additional signal because a correct final answer can be reached by a weak path, and a scientifically reasonable path can still miss a formatting detail or final value.

9 Related Work

Execution-graded agent benchmarks. The benchmarks that matter for agents grade what a model *does*, not what it can recite. SWE-bench [17] set the template with real GitHub issues whose fixes are checked by running the project's own test suite, and OpenAI's hand-audited SWE-bench Verified subset [18] showed how much curation can shift estimates on the same model. SWE-agent [19] closed the loop by handing a model a structured computer interface for repository-scale software work. GAIA [20] broadened agent evaluation to open tool use and web browsing, while Terminal-Bench [15] grades command-line workflows by inspecting final container state. τ-bench and τ^2-bench [21, 22] further emphasize objective tool-mediated outcomes and reliability across repeated trials. We inherit this lineage directly by using a real tool environment, a checkable outcome, and a headline score that counts failures rather than hiding them.

Scientific and ML-engineering benchmarks. A second line grades open-ended scientific and engineering work. ScienceAgentBench [23] draws data-driven discovery tasks from peer-reviewed papers and validates them with domain experts. MLAgentBench [24] and MLE-bench [25] evaluate agents that read files, run code, and iterate on machine-learning experiments, the latter over 75 Kaggle competitions scored against human leaderboard baselines. PaperBench [26] pushes rubric-based evaluation to full AI-research replication, asking agents to reproduce ICML papers from scratch and grading attempts with author-developed hierarchical rubrics. AstaBench [27] makes cost-aware scientific-agent evaluation explicit, organizing a broad scientific-research suite around a capability taxonomy and reporting performance on a Pareto frontier rather than as accuracy alone. GDPval [28] grounds valuable work in U.S. Bureau of Labor Statistics work activities across 44 occupations and defines success as parity with a real expert's deliverable. Together, these benchmarks move agent evaluation toward execution, replication, expert rubrics, and cost-aware comparison. DRUGDISCOVERYBENCH uses the same evaluation philosophy, but targets the narrower workflow of early drug discovery by chaining structure parsing, cheminformatics, biomedical databases, and literature or patent lookup into checkable answers.

Biomedical agents and benchmarks. Closest to our domain are tool-augmented agents for biology and chemistry. Coscientist [29] plans and runs chemistry experiments, while ChemCrow [30] and CheMatAgent [31] connect language models to chemistry tool suites, the latter paired with ChemToolBench for tool selection and parameter filling. BIOMNI [6], the framework our agents run inside, generalizes the

same idea to a few hundred biomedical functions spanning structure parsing, sequence work, database clients, and cheminformatics.

On the evaluation side, broad biology and biomedical suites–LABBench2 [32], BiomniBench [33], GeneBench [34], BixBench [35], BioMysteryBench [36], and BioXArena [37]–cover practical research tasks from literature and data access to bioinformatics, genomics inference, and executable biomedical ML pipelines; BiomniBench is especially close in scoring full trajectories against process-level expert rubrics. Drug-discovery-specific suites are closer still: SMDD-Bench [38] evaluates multi-turn small-molecule design tasks, while TxBench-PP [39] targets small-molecule preclinical pharmacology decisions from assay data. DRUGDISCOVERYBENCH differs by centering executable early drug-discovery workflows with 82 tasks from target identification through lead optimization, weighted expert rubrics, reasoning-effort sweeps, and an expert-method recovery protocol that separates planning failures from domain-knowledge gaps.

General-purpose coding agents. The agents we evaluate descend from tool-use and language-agent work that gave models an action loop and external calls. ReAct [40] interleaved reasoning with acting, Toolformer [41] showed that models can learn when external calls help, and coding-agent CLIs such as Claude Code [10], OpenAI Codex CLI [9], and Gemini CLI [11] make this abstraction operational for everyday software work. We take that general-purpose interface as given and hand it BIOMNI as an ordinary Python library inside the provided Scale-Biomni environment and no bespoke biomedical agent scaffold or MCP wrapper beyond that environment, just a Bash and Python shell. The question is whether an agent that was never specialized for the domain can plan and execute biomedical work anyway.

Test-time compute and reasoning effort. Our central variable is the compute an agent spends at inference. Snell et al. [42] show that allocating more test-time compute can, under the right strategy, outperform scaling model parameters, and the o1 system card [43] reports a reasoning model whose accuracy improves with additional inference-time reasoning compute. Recent agent benchmarks have also begun reporting compute alongside accuracy: AstaBench scores on a cost frontier, and Terminal-Bench reports input and output tokens next to resolution rate. We make this axis explicit by sweeping selected model families across low, medium, and ceiling reasoning effort and plotting accuracy against output-token budget per task (Section 5). Prior work shows that inference compute can improve reasoning accuracy; we ask whether the same holds when the bottleneck is long-horizon planning over real biomedical tools.

10 Conclusion and Future Work

DRUGDISCOVERYBENCH exposes a central tension for general-purpose coding agents in early drug discovery. These agents can write code, call biomedical tools, query real databases, and solve a substantial fraction of early-discovery tasks. Yet their failures are seldom about missing biological facts, more often they involve losing the scientific thread of the task: a disease qualifier drops out of a ranking step, a bound-state molecular question is answered with a reference-dictionary entry, or a long pipeline continues after an intermediate result has become scientifically meaningless. These failures are not solved simply by giving the model a longer context window or a larger parameter budget. A central missing capability is sound scientific reasoning and judgment, including keeping the original scientific objective active while executing many local tool calls and intermediate decisions.

The recovery experiments support this diagnosis. When agents are given an expert playbook–the sequence of steps and tools, but not the answer–many previously unsolved tasks become reachable. This points to a future that is less about asking a model to rediscover a workflow from scratch each time, and more about turning expert scientific practice into reusable procedures. In real drug discovery, much of the value sits in tacit knowledge: which database to trust for a particular question, which assay makes a comparison meaningful, when a docking score should be ignored, and when a result should trigger backtracking. Future systems should make that knowledge executable, inspectable, and reusable, with scientists remaining in the loop to author, validate, and refine the workflows that agents follow.

This shifts attention from the model alone to the science-native harness around it. A useful biomedical agent needs deterministic access to biomedical data, clear tool interfaces, pinned sources where possible, and built-in verification at intermediate steps. Test-time compute should not merely lengthen trajectories, it should support scientific checking by verifying that the current branch still answers the original question, comparing independent evidence sources, recording provenance, and backtracking when an intermediate result is biologically implausible. In this view, harness design is part of the scientific method, as it determines what evidence the agent can access, how that evidence is checked, and when the workflow should recover from a mistaken branch.

Finally, DRUGDISCOVERYBENCH argues against a single-model view of biomedical automation. Different models show different strengths across capabilities, and their performance is further shaped by the harnesses through which they access tools, manage context, and recover from errors. No one model dominates every task type. The natural next step is model-agnostic orchestration: route structural reasoning, database retrieval, patent mining, target genetics, and verification subproblems to the model-tool stack best suited for each biomedical operation. Future biomedical agents are therefore unlikely to be monolithic models. They are more likely to be routed, tool-grounded, human-supervised systems that combine deterministic infrastructure, reusable expert workflows, and model specialization into reproducible research environments.

References

[1] Olivier J. Wouters, Martin McKee, and Jeroen Luyten. Estimated research and development investment needed to bring a new medicine to market, 2009–2018. *JAMA*, 323(9):844–853, 2020. doi: 10.1001/jama.2020.1166. URL https://doi.org/10.1001/jama.2020.1166.

[2] J. P. Hughes, S. Rees, S. B. Kalindjian, and K. L. Philpott. Principles of early drug discovery. *British Journal of Pharmacology*, 162(6):1239–1249, 2011. doi: 10.1111/j.1476-5381.2010.01127.x. URL https://doi.org/10.1111/j.1476-5381.2010.01127.x.

[3] Eric Vallabh Minikel, Jeffery L. Painter, Coco Chengliang Dong, and Matthew R. Nelson. Refining the impact of genetic evidence on clinical success. *Nature*, 629(8012):624–629, 2024. doi: 10.1038/s41586-024-07316-0. URL https://doi.org/10.1038/s41586-024-07316-0.

[4] Ali Essam Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J. Szostkiewicz, Dmytro Shved, Gavin J. Gyimesi, Jon M. Laurent, Samantha M. Wright, Muhammed T. Razzak, Andrew D. White, Silvia C. Finnemann, Michaela M. Hinks, and Samuel G. Rodriques. A multi-agent system for automating scientific discovery. *Nature*, 2026. doi: 10.1038/s41586-026-10652-y. URL https://doi.org/10.1038/s41586-026-10652-y. Advance online publication.

[5] Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky,

Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, Anil Palepu, Keran Rong, Ryutaro Tanno, Khaled Saab, Fan Zhang, Jacob Blum, Andrew Carroll, Kavita Kulkarni, Nenad Tomašev, Dina Zverinski, Ivor Rendulic, Elahe Vedadi, Florian Hasler, Luka Rimanic, Marina Boia, Ivan Budiselic, Ben Feinstein, Mathias Bellaiche, Tom Sheffer, Jan Freyberg, Jeremy Ratcliff, Ottavia Bertolli, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Yuan Guan, Vikram Dhillon, Eeshit Dhaval Vaishnav, Byron Lee, Tiago R. D. Costa, José R. Penadés, Gary Peltz, Yossi Matias, James Manyika, Demis Hassabis, Yunhan Xu, Pushmeet Kohli, Annalisa Pawlosky, Alan Karthikesalingam, and Vivek Natarajan. Accelerating scientific discovery with Co-Scientist. *Nature*, 2026. doi: 10.1038/s41586-026-10644-y. URL https://doi.org/10.1038/s41586-026-10644-y. Advance online publication.

[6] Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Junze Zhang, Di Yin, et al. Biomni: A general-purpose biomedical AI agent. *bioRxiv*, 2025. doi: 10.1101/2025.05.30.656746. URL https://www.biorxiv.org/content/10.1101/2025.05.30.656746v1.

[7] Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

[8] Zuojun Xu, Feng Ren, Ping Wang, Jie Cao, Chunting Tan, Dedong Ma, Li Zhao, Jinghong Dai, Yipeng Ding, Haohui Fang, Huiping Li, Hong Liu, Fengming Luo, Ying Meng, Pinhua Pan, Pingchao Xiang, Zuke Xiao, Sujata Rao, Carol Satler, Sang Liu, Yuan Lv, Heng Zhao, Shan Chen, Hui Cui, Mikhail Korzinkin, David Gennert, and Alex Zhavoronkov. A generative AI-discovered TNIK inhibitor for idiopathic pulmonary fibrosis: a randomized phase 2a trial. *Nature Medicine*, 31(8):2602–2610, 2025. doi: 10.1038/s41591-025-03743-2.

[9] OpenAI. OpenAI Codex CLI, 2025. URL https://github.com/openai/codex.

[10] Anthropic. Claude Code, 2025. URL https://claude.com/product/claude-code.

[11] Google. Gemini CLI, 2025. URL https://github.com/google-gemini/gemini-cli.

[12] John Yang, Kilian Lieret, et al. mini-SWE-agent, 2025. URL https://github.com/SWE-agent/mini-swe-agent.

[13] Mario Zechner. Pi Agent Harness, 2025. URL https://github.com/earendil-works/pi.

[14] Anomaly. opencode: The open source coding agent, 2026. URL https://github.com/anomalyco/opencode. GitHub repository.

[15] Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=a7Qa4CcHak.

[16] Xinming Tu, Tianze Wang, Yingzhou Lu, Kexin Huang, Yuanhao Qu, and Sara Mostafavi. BenchGuard: Who guards the benchmarks? automated auditing of LLM agent benchmarks. *arXiv preprint arXiv:2604.24955*, 2026. URL https://arxiv.org/abs/2604.24955.

[17] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.06770.

[18] OpenAI. Introducing SWE-bench verified, 2024. URL https://openai.com/index/introducing-swe-bench-verified/. OpenAI blog announcement, August 13, 2024.

[19] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2405.15793.

[20] Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2311.12983.

[21] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024. URL https://arxiv.org/abs/2406.12045.

[22] Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025. URL https://arxiv.org/abs/2506.07982.

[23] Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.05080.

[24] Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2310.03302.

[25] Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.07095.

[26] Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI's ability to replicate AI research. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 56843–56873. PMLR, 2025. URL https://proceedings.mlr.press/v267/starace25a.html.

[27] Jonathan Bragg, Mike D'Arcy, Nishant Balepur, Dan Bareket, Bhavana Dalvi, Sergey Feldman, et al. AstaBench: Rigorous benchmarking of AI agents with a scientific research suite. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2510.21652.

[28] Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, et al. GDPval: Evaluating AI model performance on real-world economically valuable tasks. *arXiv preprint arXiv:2510.04374*, 2025. URL https://arxiv.org/abs/2510.04374.

[29] Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023. doi: 10.1038/s41586-023-06792-0. URL https://www.nature.com/articles/s41586-023-06792-0.

[30] Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6(5):525–535, 2024. doi: 10.1038/s42256-024-00832-8. URL https://www.nature.com/articles/s42256-024-00832-8.

[31] Mengsong Wu, YaFei Wang, Yidong Ming, Yuqi An, Yuwei Wan, Wenliang Chen, Binbin Lin, Yuqiang Li, Tong Xie, and Dongzhan Zhou. CheMatAgent: Enhancing LLMs for chemistry and materials science through tree-search based tool learning. *arXiv preprint arXiv:2506.07551*, 2025. URL https://arxiv.org/abs/2506.07551.

[32] Jon M. Laurent, Albert Bou, Michael Pieler, Conor Igoe, Alex Andonian, Siddharth Narayanan, James Braza, Alexandros Sanchez Vassopoulos, Jacob L. Steenwyk, Blake Lash, Andrew D. White, and Samuel G. Rodriques. LABBench2: An improved benchmark for AI systems performing biology research. *arXiv preprint arXiv:2604.09554*, 2026. URL https://arxiv.org/abs/2604.09554.

[33] Yuanhao Qu, Yingzhou Lu, Xinming Tu, Serena Zhang, Tianwei She, Alexander Glenn Shaw, Jou-Ho Shih, Bingqing Zhao, Minjie Shen, Haochen Yang, Jielin Yan, Rongchuan Zhang, Xinze Wu, Tingting Li, Bin Zhou, Ning Wang, Adam Ma, Le Cong, Xiaobo Hu, Yuan Jiang, Jiayun Dong, Tao Peng, Jure Leskovec, and Kexin Huang. BiomniBench: Process-level evaluation of LLM agents for real-world biomedical research. *bioRxiv*, 2026. doi: 10.64898/2026.05.12.724604. URL https://www.biorxiv.org/content/10.64898/2026.05.12.724604v2.

[34] Jeremy Li and Andrew Ho. GeneBench: Assessing AI agents for multi-stage inference problems in genomics and quantitative biology. *bioRxiv*, 2026. doi: 10.64898/2026.04.22.720113. URL https://www.biorxiv.org/content/10.64898/2026.04.22.720113v1.

[35] Ludovico Mitchener, Jon M. Laurent, Alex Andonian, Benjamin Tenmann, Siddharth Narayanan, Geemi P. Wellawatte, Andrew White, Lorenzo Sani, and Samuel G. Rodriques. BixBench: A comprehensive benchmark for LLM-based agents in computational biology. *arXiv preprint arXiv:2503.00096*, 2025. URL https://arxiv.org/abs/2503.00096.

[36] Anthropic. Evaluating claude's bioinformatics research capabilities with BioMysteryBench, 2026. URL https://www.anthropic.com/research/Evaluating-Claude-For-Bioinformatics-With-BioMysteryBench. Anthropic research report, April 29, 2026.

[37] Loka Li, Duzhen Zhang, Xingbo Du, Leonard Song, Zixiao Wang, Assanali Aukenov, Noel Thomas, Shakhnazar Sailaukan, Yonghan Yang, Feilong Chen, Jiahua Dong, Kun Zhang, Bin Zhang, and Le Song. BioXArena: Benchmarking LLM agents on multi-modal biomedical machine learning tasks. *arXiv preprint arXiv:2605.15766*, 2026. URL https://arxiv.org/abs/2605.15766.

[38] Kevin Han, Renfei Zhang, Kathy Wei, Hamed Mahdavi, Niloofar Mireshghallah, and Amir Barati Farimani. SMDD-bench: Can LLMs solve real-world small molecule drug design tasks? *arXiv preprint arXiv:2605.21740*, 2026. URL https://arxiv.org/abs/2605.21740.

[39] Hannah Le, Ramesh Ramasamy, Alex Urrutia, Mahsa Yazdani, Tim Proctor, and Kenny Workman. TxBench-PP: Analyzing AI agent performance on small-molecule preclinical pharmacology. *arXiv preprint arXiv:2606.19245*, 2026. URL https://arxiv.org/abs/2606.19245.

[40] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2210.03629.

[41] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2302.04761.

[42] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024. URL https://arxiv.org/abs/2408.03314.

[43] OpenAI. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024. URL https://arxiv.org/abs/2412.16720.

A Additional Results

A.1 Full Results

Figure 7 shows evaluation results in DRUGDISCOVERYBENCH in all settings considered in this work.

A.2 Same harness different models

Figure 8 depicts model performances when we utilize the same mini-SWE-agent harness.

B Definitions of Capabilities

In Table 2 *Structural reasoning*, the largest group, reads a property off a real co-crystal structure: a bound ligand's net charge, a count of hydrogen bonds or salt bridges, a coordination number, an InChIKey or formula. *Database screening* chains queries across resources such as OpenTargets, ChEMBL, UniProt, and PubChem to land on a gene, target, compound, or short table. *Patent mining* pulls a specific patent or paper, extracts a compound table from long unstructured text, and applies multi-threshold filters with explicit tie-breaks. *Target identification and genetics* triages a disease to a gene or target through variant, expression, and association evidence. *Cheminformatics* identifies the right molecule and computes one well-defined descriptor (a Crippen LogP, a TPSA, a monoisotopic mass, a canonical SMILES, a structural-alert set). *SAR / affinity ranking* orders matched analogs by predicted affinity from their modeled protein contacts. *Molecular biology* works directly on sequence: primer and construct design, directional cloning, fusion-protein assembly, and tryptic-peptide and mass-spectrometry calculations. Structure- and chemistry-centric work dominates the set.

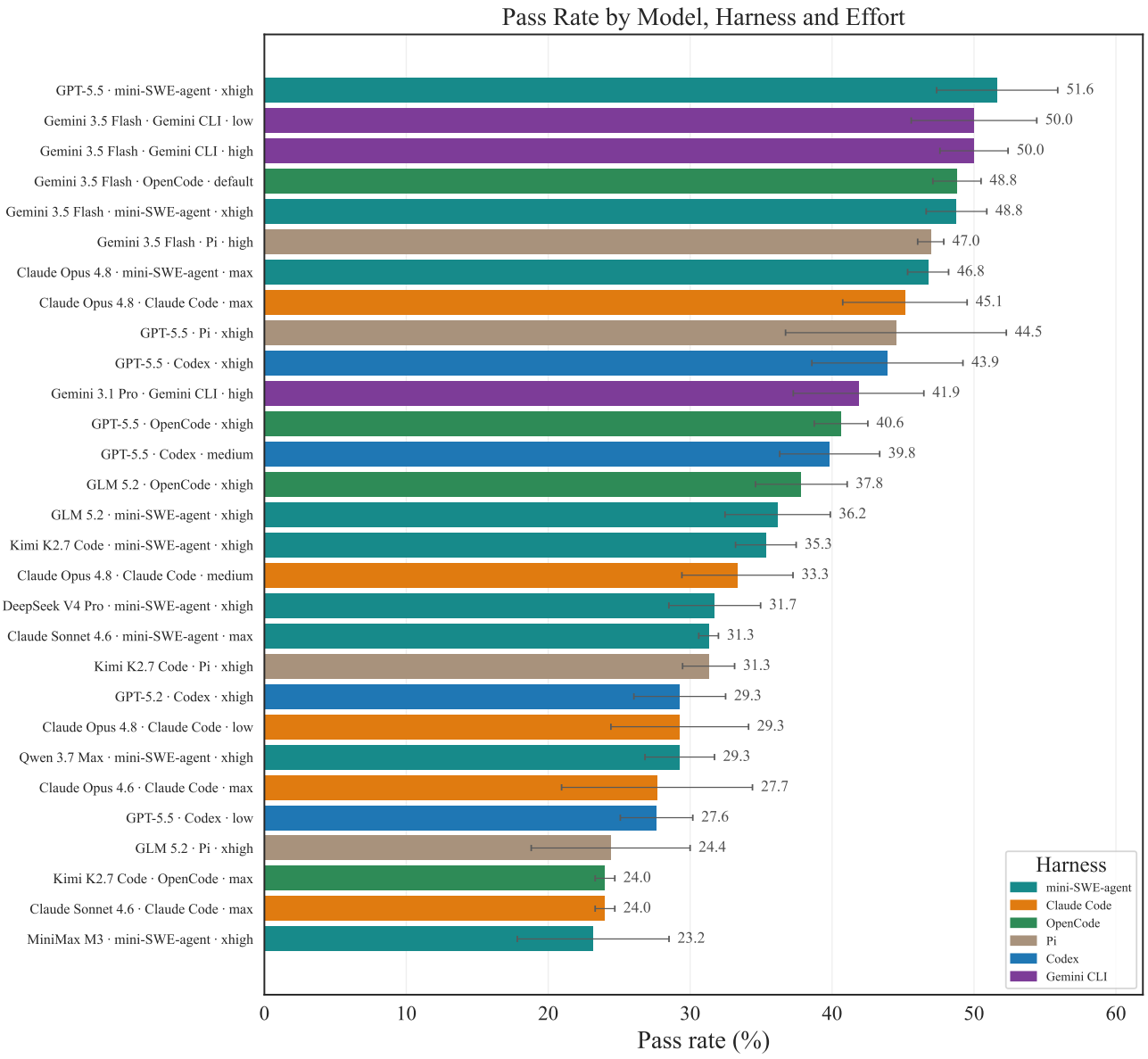


Figure 7. Pass rates across all settings, averages across 3 trials.

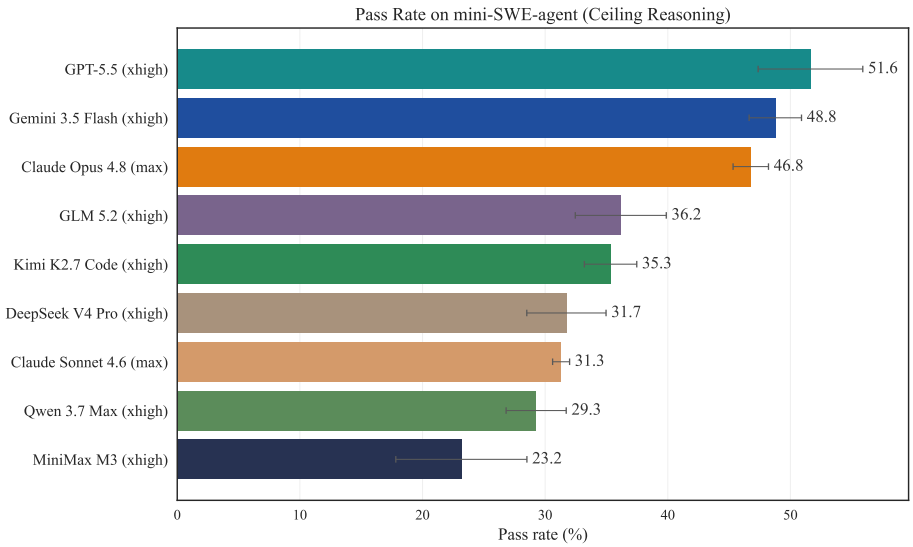


Figure 8. LLM performances using mini-SWE-agent as the harness. Maximum reasoning effort.

B.1 Claude Opus Refusals based on AUP

Some of the tasks Opus refused on usage-policy grounds are: competitor-patent analysis (`fa9338`, `3b0192`, `3b017a`), HIV-1 RT inhibitor ranking (`e3c5e2`, PDB 6UL5), retrieval of published OX-40L X-ray structures (`3b017f`), *M. tuberculosis* (H37Rv) target identification (`d5b551`), and a metal-ligand net-charge task (`e3c60e`, PDB 7KYF). All are routine analyses of already-published structural and patent data, so we treat the refusals as a capability gap rather than a safety boundary and score these runs as 0. Most of these refusals are reproducible across harnesses, and when using Claude Sonnet 4.6 instead of Opus 4.8. Meanwhile, GPT-5.5 and Gemini 3.5 Flash answer every task. These refusals are a Claude-family behavior tied to the request content.

This makes reliability a distinct axis from accuracy. On the tasks it does attempt, Opus is competitive with the leaders. Using mini-SWE-agent, its mean outcome score over answered tasks (59.6) is on par with GPT-5.5 (59.4) and Gemini 3.5 Flash (59.3). However, because each refusal is scored as a failure, its reported score understates that ability. This is an important consideration for a practictioner, who must choose whether to trust a model that is strong where it answers yet reliably declines a fraction of routine biomedical queries, or use a slightly less accurate model that always returns an answer.

C The environment in detail

Agents work inside BIOMNI, the Stanford SNAP-lab biomedical agent framework, in a version adapted at Scale for evaluation. BIOMNI exposes a few hundred biomedical functions as an ordinary Python library, grouped by domain; the agent imports and calls them the way it would any package (`from biomni.tool.<domain> import <function>`), organized into the domains in Table 8. The `database` domain holds the `query\_*` clients for the large reference databases (PubChem, UniProt, ChEMBL, Open Targets, ClinVar, ENCODE, and others), which auto-route to a local SQLite copy when one is present and fall back to the live API otherwise.

A curated data lake is mounted read-only at `/workspace/data/biomni_data/data_lake/`, holding 76 reference files including BindingDB, GTEx, DepMap, the GWAS catalog, GeneBass, gene-info, DDInter,

Table 7. Domain taxonomy of BIOMNI. The six areas partition all 22 domains and 226 functions; representative on-disk datasets and operations are listed alongside.

Area (domains, #functions)	Representative datasets	Representative operations
Knowledge bases & literature database 42, literature 7, support 5	Knowledge graph (`kg.csv`), DisGeNET, OMIM, TxGNN predictions	Identifier mapping; query formulation; cross-database joins
Genomics & genetics genomics 19, genetics 9	GWAS Catalog, variant table, GeneBass, `gene_info`	Coordinate conversion; variant interpretation; reference handling
Molecular & cell biology mol. biology 18, cell 5, biochemistry 6, biophysics 3	Protein–protein interaction maps (affinity-capture MS, two-hybrid)	Construct & assay design; parameter and unit correctness
Pharmacology & drug discovery pharmacology 25	BindingDB, Broad Repurposing Hub, DDInter, Enamine	SMILES / structure handling; affinity and interaction reasoning
Physiology, immunology & disease physiology 11, microbiology 12, immunology 10, pathology 7, cancer 6	DepMap CRISPR effects, Human Protein Atlas, McPAS-TCR	Cell-type annotation; phenotype reasoning; statistical testing
Imaging, systems & bioengineering bioimaging 9, synthetic 8, systems 7, bioeng. 7, protocols 4, glyco 3, lab-automation 3	MSigDB & MouseMine gene sets, CZI single-cell census	Image segmentation; pathway / enrichment analysis; protocol synthesis

Table 8. Tool domains available under `biomni.tool` on the trial container.

biochemistry	bioengineering	bioimaging	biophysics
cancer_biology	cell_biology	database	genetics
genomics	glycoengineering	immunology	lab_automation
literature	microbiology	molecular_biology	pathology
pharmacology	physiology	protocols	support_tools
synthetic_biology	systems_biology		

MSigDB, miRTarBase, and GO. An on-container inventory (`/biomni/biomni/AVAILABLE_DATA.md`), per-domain tool source, data-resource descriptions (`env_desc.py`), and a handful of step-by-step workflow descriptions (`know_how/*.md`) are all available for the agent to read before it acts. Each trial runs in a fixed container image (`ghcr.io/scaleapi/drugdiscoverybench:1.0.0-full`) with 2 CPUs, 8 GB of memory, and internet access enabled, under a 7200-second agent timeout; the verifier runs separately with a 600-second timeout.

Task-presentation template. The agent receives the task-specific prompt followed by a fixed footer describing the tool suite, the on-container references, the local data lake, and the requirement to write only the final answer to `/workspace/answer.md`. The footer is as follows:

Shared task footer (identical across all 82 tasks)

```
You have access to Biomni's biomedical tool suite. Every Biomni tool is importable in Python as
`biomni.tool.<domain>.<name>` and runnable from Bash, e.g. `python -c "from biomni.tool.<domain>
```

```
import <name>; print(<name>(...))"`. The biomni package and its conda env
(`/opt/conda/envs/biomni_e1/`) are installed on this container.

Before writing code, consult the on-container reference so you call the right function with the
right arguments instead of guessing:

- `/biomni/biomni/tool/<domain>.py` -- implementations of every tool, grouped by domain
  (genomics, pharmacology, molecular_biology, literature, database, etc.); read the function you
  intend to use for its exact signature and behavior. `database.py` holds the `query_*` helpers
  for the large reference DBs.
- `/biomni/biomni/env_desc.py` -- descriptions of the local data-lake files and the available
  data resources, so you know what is already on disk before fetching anything.
- `/biomni/biomni/know_how/*.md` -- step-by-step playbooks for common workflows (e.g. gene/drug-
  target identification, sgRNA design, single-cell annotation); skim the relevant guide before
  starting a multi-step analysis.

## Local data already on the container

The Biomni data lake is mounted read-only at `/workspace/data/biomni_data/data_lake/` -- 76
curated biomedical files including BindingDB, GeneBass, GTEx, DepMap, GWAS catalog, gene-info,
DDInter, MSigDB, miRTarBase, GO, etc. **Check `/biomni/biomni/AVAILABLE_DATA.md` for the full
inventory with one-line descriptions before downloading anything from a public API**; reading a
local parquet/CSV is much faster than refetching from the internet.

For larger reference databases (PubChem, UniProt, PubMed, ChEMBL, OpenTargets, ClinVar, ENCODE,
etc.), prefer the corresponding `biomni.tool.database.query_*` functions -- they auto-route to
the local SQLite copy when available and fall back to the live API otherwise.

## Final answer

When you have your answer, write it -- and only it, no prose explanation -- to
`/workspace/answer.md`. Keep it terse by default; but if the task specifies an output format
(e.g. JSON, a table, a SMILES string, a comma-separated list), produce exactly that format as
the answer.
```

C.1 Docker Image sizes

Table 9 provides details into three docker images we release with different levels of database coverage.

D Failure Modes Example Trajectories

We provide two representative examples for each failure mode, spanning the Claude, Gemini, and GPT model families. Table 10 summarizes each case, describing what the agent did and the single step it missed. We verified each case against the agent run, the gold trajectory, and the task's grading rubric. These examples illustrate a recurring pattern in which the agents may perform most of the task correctly, and still fail critically at a single identifiable step.

Table 9. DRUGDISCOVERYBENCH image tiers. All 82 Harbor tasks pin the `lightweight` image, which bakes only the initial BIOMNI data-lake files that have *no* public-API equivalent (∼2.3 GB) and falls back to live public APIs for all database `query_*` tools; the `core` and `full` tiers additionally bake biomedical databases into the image for offline, deterministic execution. "Offline DB coverage" is the fraction of the benchmark's data-lake `query_*` tool calls (measured from our evals) served from local databases rather than live APIs. Experiments in this work use the `full` tier.

		Size (GB)		Offline DB
Tier (tag)	Baked Data (Beyond default)	Pull	On-disk	Coverage
`1.0.0-lightweight`	Only the initial BIOMNI data-lake files with no public-API equivalent (∼2.3 GB): `gene_info.parquet`, `gtex_tissue_gene_tpm.parquet`, `variant_table.parquet`, `proteinatlas.tsv`, `kg.csv`, `txgnn_name_mapping.pkl`, `broad_repurposing_hub_molecule_with_smiles.parquet`, `broad_repurposing_hub_phase_moa_target_info.parquet`, `DepMap_CRISPRGeneEffect.csv`, `DepMap_CRISPRGeneDependency.csv`, `DepMap_Model.csv`, `DepMap_OmicsExpressionProteinCodingGenesTPMLogp1.csv`, `affinity_capture-ms.parquet`, `co-fractionation.parquet`, `two-hybrid.parquet`, `proximity_label-ms.parquet`. All API-backed database queries hit live public APIs	6	∼22	live
`1.0.0-core`	∼21 local databases (cBioPortal, UniProt, ChEMBL, ENCODE, Ensembl, ClinicalTrials/AACT, STRING, openFDA, Reactome, GEO, GWAS, PDB, …) + all bundled data-lake files	37	∼210	∼74%
`1.0.0-full`	`core` + PubChem, PubMed, AlphaFold, Open Targets	136	∼570	∼99.7%

Mode	Model	Task	What the agent did, and the step it missed
Domain reasoning	GPT-5.2	`e3c5f1`	Computed the tryptic-peptide fragment masses with exact arithmetic but added only one phosphate group, failing to recognize that CK2α phosphorylates two residues in the peptide. It reported m/z 419.89 and 629.34 rather than the correct 446.55 and 669.32.
	Gemini 3.5 Flash	`bf9c76`	Computed the hydrogen-bond-donor count correctly, but on the PDB-deposited ligand (Z4I, an in-situ hydrolysis product) instead of the intended inhibitor. It reported 3 donors, whereas the correct value is 1.
Derivation error	GPT-5.5	`bf9c6d`	Correctly identified the single Asp1088 carboxylate-to-amine salt bridge, and its own code printed the correct group-level count of 1, but the final tally used the atom-level count of 2 and double-counted the bridge. It reported 8 interactions instead of 7.
	Gemini 3.1 Pro	`3b017a`	Selected the correct three inhibitors and ran the maximum-common-substructure computation, but the result over-included an aromatic carbonyl ring shared by only two of the three molecules, yielding a common scaffold larger than the true one.
Retrieval	Claude Opus 4.6	`c4222c`	Searched the wrong structural database (RCSB rather than the NAKB nucleic-acid database) and never queried the correct source, so the qualifying EM structures were never retrieved; it answered "None" where the gold answer is three structures.
	GPT-5.2	`e3c63e`	Queried PubChem through the assay, compound, and cross-reference endpoints but never the literature-collection endpoint holding the required record, and consequently reported a molecular weight of 394.4 instead of 527.8.
Constraint	Claude Opus 4.6	`c4222f`	Counted pathogenic/likely-pathogenic ClinVar variants over all conditions, ignoring the prompt's explicit restriction to melanoma, and therefore ranked the wrong gene (PTEN rather than CDKN2A).
	GPT-5.5	`bf9c78`	Retrieved all six required compounds but added two extra rows, returning eight entries despite the explicit instruction to report exactly one entry per compound.
Final-answer slip	Claude Opus 4.8	`e3c5e9`	Completed the full pipeline and correctly identified Haloperidol, but omitted the required SMILES string from the final answer.
	Gemini 3.5 Flash	`fa9335`	Recognized that the deposited ligand was the post-reaction form and computed both the free-inhibitor and bound-adduct values, then reported the bound-adduct numbers instead of the free-inhibitor values it had already derived.

Table 10. Representative failure trajectories. For each we summarize the agent's (correct) work and the single load-bearing step it missed. Task identifiers are internal benchmark IDs.