Archimedes finds 57% more rare cancer vulnerabilities than a frontier language model in the same experiments

Pol Castellano-Escuder[1,*]

November 23, 2026

[1]Heureka Labs, Raleigh, NC, USA

*Correspondence: pol@heurekalabs.co

Abstract

The expensive part of research is choosing which experiment to run. We asked whether an AI agent finds a cancer cell line's rare gene dependencies in fewer experiments when it works with Archimedes, Heureka's model of measured biology, than when the same agent works without it. The task, the arms, the metric and the criterion for success were pre-registered. The test set was every cell line first released in DepMap 26Q1: 31 lines, mostly rare and pediatric cancers, none of them in Archimedes' training or reference data. For each line Claude Opus 5.5 ran campaigns of 30 knockout experiments, three with Archimedes and three without, each experiment answered from the line's real held-out measurement. With Archimedes the agent found 57% more rare dependencies in the same 30 experiments: 2.56 more per line (95% CI 1.59 to 3.54; $p = 2 \times 10^{-4}$). It matched the agent's 30-experiment total after 17 experiments, a saving of 43%. A third arm set Archimedes against the strongest alternative we could build: the same agent handed the public data Archimedes learned from, and more, already selected and prepared, with a code sandbox and a longer time limit. Archimedes delivered the yield of that agent (difference 0.27 per line, 95% CI −0.46 to 0.97) in less than half the time, 2.8 minutes per campaign against 6.3. It was far more reproducible: repeat campaigns shared 71% of their experiments with Archimedes against 38%. And its worst campaign found about one discovery more. Archimedes returns in one call, the same way every time, what the data-equipped agent has to rebuild in every campaign. An audit re-scored all 8,370 experiments from the raw data without a single mismatch.

Introduction

A discovery campaign is a sequence of choices. Each experiment costs money and time, and most of the candidates a lab could test will not pay off. A lab that chooses well reaches the same result with fewer experiments, or finds more with the same budget.

Language models are now capable scientific agents, and they carry a great deal of published biology. But the answer to "which gene does *this* cell line depend on" is rarely in the literature. It is in measurements. Archimedes is built on measured biology: for cancer dependencies, it learns from the gene-effect profiles of about 1,100 cell lines and predicts, for a line it has never seen, which genes that line is likely to need.

This preprint tests the claim that matters for an autonomous lab: that an agent choosing its experiments with Archimedes gets more out of every experiment than the same agent choosing alone. We test it on the

hardest version of the question, the rare dependencies. A dependency shared by most cancer lines is well known and easy to guess. A dependency found in fewer than one line in ten is a candidate selective target, and it is the kind that an agent cannot recall from general knowledge.

A second comparison measures what Archimedes delivers against an agent that is given the raw data and a place to analyze it. That arm was built to be as strong as possible: the data were chosen and prepared for the agent, and it had more time.

Methods

Pre-registration

The design was fixed on October 11, 2026, about an hour before the first experiment, as a dated commit in version control: the cell lines, the task, the two arms, the number of runs, the primary metric, the criterion for success, the secondary measures, and the handling of failed runs. A third arm, in which the agent receives public data and a code sandbox in place of Archimedes, was added to the pre-registration the same day, before any campaign of that arm, with its own comparison and criterion. Its campaigns ran on November 23, 2026, with the same task, prompt and model.

Cell lines

The test set is every cell line whose CRISPR gene-effect data first appeared in DepMap Public 26Q1 (1,208 lines) and is absent from DepMap 24Q4 (1,178 lines) and from Project Score 2021. There are 31. They were not selected: they are all the lines that meet the rule. They are 13 bone, 5 soft-tissue, 6 blood, 3 peripheral nervous system, 2 central nervous system, 1 liver and 1 pancreas line, and most are rare or pediatric cancers (Table 3). Archimedes was trained on 24Q4 data only, and none of the 31 lines is in its training labels or in the reference data it consults.

Task

The agent was told it was studying an uncharacterized cancer cell line, "line X". It could read the line's lineage, disease, subtype and hotspot mutations, query the line's gene expression and copy number, and look up how often any gene is a dependency across public cancer lines. These tools were free. An *experiment* was one gene knockout, which returned the line's measured CRISPR gene effect for that gene from the held-out 26Q1 data. Each campaign had a budget of 30 experiments.

A *discovery* was a knocked-out gene with a gene effect below -0.5 in that line that is a dependency in fewer than 10% of cancer lines. Discoveries were counted from the environment's own log of knockouts, never from what the agent reported.

Arms

The same language model, Claude Opus 5.5 running in Claude Code, chose every experiment in every arm. Each campaign started in an empty environment in which only the study's tools could be called: no files, shell, web access, memory or other sessions. The prompts are given in the appendix.

Language model alone. The task tools only. The agent could knock out any gene in the genome.

Language model + Archimedes. The same tools, plus Archimedes' ranked hypotheses for the line, updated after every result. Each knockout had to be one of Archimedes' current top five hypotheses, and the agent chose among them with its own judgement.

Language model + data + code. The task tools, plus a data workspace and a Python environment (pandas, NumPy, SciPy and scikit-learn, up to 15 minutes per call). The workspace held everything Archimedes' dependency model was built from, and more: DepMap 24Q4 CRISPR gene effects for all 17,916

genes in the 1,109 training lines (Archimedes ranks 2,416 of them), RNAi gene effects, expression, copy number, mutations, genomic signatures and annotations for those lines, and the same data types for line X. We chose the data, cleaned it and removed the 31 test lines, so the agent started with the right data already in place. The sandbox had no network access and could read nothing outside the workspace. The time limit was 60 minutes per campaign, against 40 in the other arms. The agent could knock out any gene in the genome. Each of these choices favors this arm.

Each arm ran three times on every line, for $31 \times 3 \times 3 = 279$ campaigns and 8,370 experiments. The first comparison isolates what Archimedes adds to an agent: the two arms share the model, the prompt, the line's own data and the budget, and differ only in Archimedes. The second measures Archimedes against an agent that does the analysis itself.

Reference arms. Four arms with no language model ran on the same lines: Archimedes taking its own top-ranked gene each time; a lookup that ranks genes by how often they are dependencies in the line's cancer type; Archimedes given another line's molecular profile, as a control; and random choice among eligible genes.

Analysis

Primary. For each line, the mean number of discoveries within 30 experiments over the three runs, with Archimedes minus without, paired by line, with a 95% confidence interval by bootstrap over lines. Success was defined in advance: the interval excludes zero.

Secondary. The speed multiplier (for each line, 30 divided by the number of experiments the Archimedes arm needed to reach the model-alone 30-experiment mean; median over lines), and the number of experiments to the first and third discovery.

Two analyses. In 16 of the 93 model-alone campaigns the language model declined a turn on a biology safety filter, and Claude Code continued the campaign on the previous model version. No Archimedes campaign was affected. Before the final analysis we fixed two analyses and a reporting rule. *As run* keeps every campaign. *Per protocol* excludes the 16 affected campaigns, and with them one line that had no unaffected model-alone campaign. The claim is made only if both analyses meet the criterion, and the smaller effect is the one quoted.

Third arm. The pre-registered comparison is the per-line difference in discoveries, Archimedes arm minus data-and-code arm, with the same bootstrap interval. The secondary measures are the data-and-code arm minus the model alone, and wall time per campaign. No campaign in this arm was affected by the fallback, so its two analyses coincide. We also describe how reproducible each arm is across its three campaigns on a line: the spread in discoveries, the share of experiments two repeat campaigns have in common, and the worst of the three campaigns.

Audit and independent re-analysis

An audit script, written separately from the study code, re-read the raw DepMap files and re-scored every knockout in every campaign. It also checked that the 31 lines are absent from Archimedes' training labels, from its reference data and from Project Score 2021; that every campaign completed with exactly 30 distinct knockouts; that neither comparison arm ever called Archimedes; that the first two arms' tool sets differ only by the Archimedes tools; and that none of the 1,491 code executions in the data-and-code arm reached outside its workspace. For this preprint the full analysis was then repeated from the raw campaign logs with newly written code.

Results

With Archimedes, the agent found 57% more in the same 30 experiments

The pre-registered criterion was met in both analyses (Table 1). We quote the smaller. Over 30 lines, the agent with Archimedes found 211.0 rare dependencies against 134.2 without it: 57% more, or 2.56 more per line (95% CI 1.59 to 3.54; Wilcoxon signed-rank $p = 2 \times 10^{-4}$). Keeping every campaign, the difference is larger: 2.73 per line (95% CI 1.77 to 3.68; $p = 6 \times 10^{-5}$), 61% more over 31 lines.

	As run	Per protocol
Lines; campaigns (alone, with Archimedes)	31; 93, 93	30; 77, 90
Discoveries in 30 experiments, summed over lines: alone → with Archimedes	138.3 → 223.0	134.2 → 211.0
Primary: difference per line (95% CI)	**+2.73 (1.77 to 3.68)**	**+2.56 (1.59 to 3.54)**
Wilcoxon signed-rank over lines	$p = 6 \times 10^{-5}$	$p = 2 \times 10^{-4}$
Lines won / tied / lost by Archimedes	24 / 1 / 6	23 / 3 / 4
Speed multiplier, median over lines	1.94	1.88
Experiments to the third discovery, difference (95% CI)	−5.56 (−7.91 to −3.25)	−5.24 (−7.72 to −2.82)
Experiments to the first discovery, difference (95% CI)	−1.76 (−4.58 to +0.73)	−1.80 (−4.58 to +0.74)

Table 1. Pre-registered outcomes in both planned analyses. Differences are the Archimedes arm minus the model-alone arm, paired by line; intervals are bootstrap over lines. The quoted result is the per-protocol column, the smaller of the two.

The same yield in 17 experiments instead of 30

Summed over the 30 lines, the agent with Archimedes had found as many rare dependencies after 17 experiments as the agent alone found in all 30 (Figure 1): 13 of 30 experiments saved, or 43%. The curves separate within the first five experiments and the gap keeps widening to the end of the campaign. The pre-registered per-line measure says the same thing a little more strongly: a median speed multiplier of 1.88, or the model-alone yield in about 16 experiments. The third discovery arrived 5.2 experiments sooner (95% CI 2.8 to 7.7).

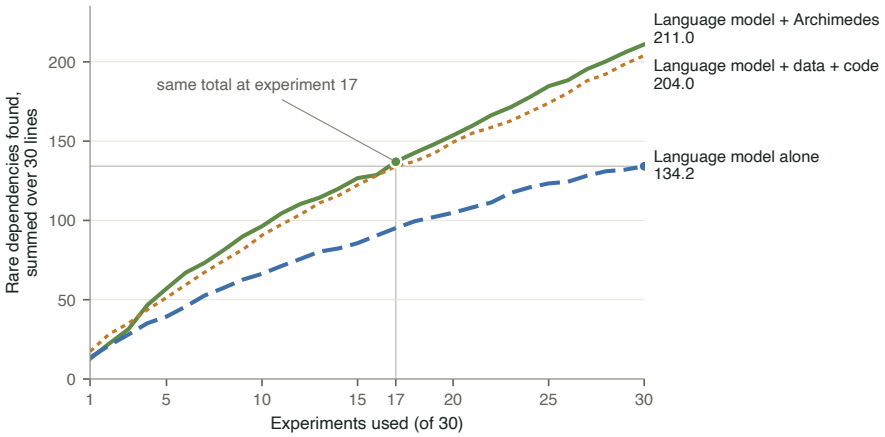

Figure 1. The agent with Archimedes reaches the model-alone total at experiment 17. Rare dependencies found as experiments accumulate, summed over the 30 per-protocol lines (each line is the mean of its campaigns). Solid green: language model + Archimedes. Dashed blue: language model alone. Dotted orange: language model + data + code.

The gain is consistent across lines

Archimedes won on 23 of 30 lines, tied on 3 and lost on 4 (Figure 2). The median per-line gain is 2.67. The result does not rest on a few lines: dropping the three lines where Archimedes helped most leaves a gain of 2.06 per line, and dropping any single line leaves at least 2.34. It holds on the 21 lines that played no part in piloting the study (+2.38 per line, 95% CI 1.30 to 3.43, $p = 0.0015$). It holds under a sign-flip permutation test ($p = 4 \times 10^{-5}$) and under a bootstrap that resamples campaigns as well as lines (95% CI 1.69 to 3.74, as run). Campaigns with Archimedes were also more repeatable: the run-to-run standard deviation on the same line was 0.60 discoveries against 0.84.

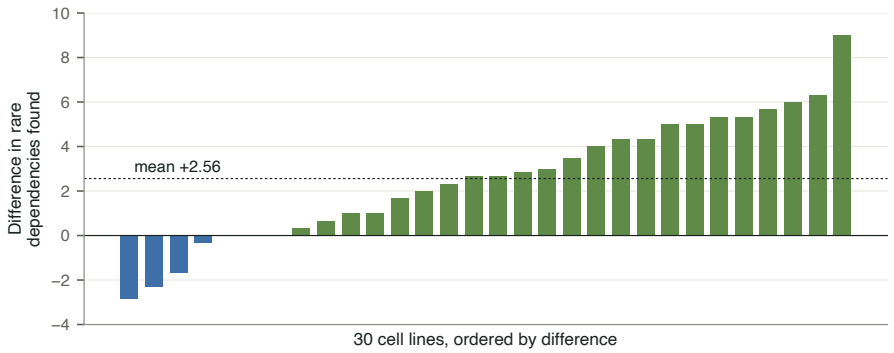

Figure 2. Archimedes wins on 23 of 30 lines. Per-line difference in discoveries within 30 experiments, language model + Archimedes minus language model alone (per protocol). Green: Archimedes ahead. Blue: model alone ahead. Dashed line: the mean.

The advantage comes from measured biology

The reference arms show where the gain comes from (Figure 3). Over all 31 lines, random choice found 56 rare dependencies and the agent alone found 138. A simple lookup of how often each gene is a dependency in the line's cancer type found 172, more than the agent alone (+1.09 per line, 95% CI 0.16 to 2.05): on this question, general knowledge does not substitute for measurements. Archimedes on its own found 214, and the agent choosing among Archimedes' hypotheses found 223, four times random and 1.65 more per line than the lookup (95% CI 0.88 to 2.51).

The control confirms that Archimedes is reading the line in front of it. Given another line's molecular profile, it found 113, below the lookup. Its advantage is specific to each line's own biology.

The agent alone was not held back by the rules. It was free to knock out any gene in the genome, and 119 of its 415 discoveries were genes outside the set Archimedes ranks, which the Archimedes arm could not choose. The Archimedes arm won from inside a smaller search space.

Archimedes delivers the yield of a data-equipped agent in less than half the time, and far more reproducibly

Over the 31 lines the agent with Archimedes found 223.0 rare dependencies. The same agent with the prepared workspace and the code sandbox found 214.7, and the agent alone 138.3 (Table 2). The pre-registered comparison for this arm is the difference in yield: 0.27 per line in favor of Archimedes (95% CI −0.46 to 0.97). The interval includes zero, so the two arms are equal in yield. Archimedes on its own, with no language model at all, finds as many (214).

Archimedes holds in a single call what the data-equipped agent rebuilds in every campaign. That agent wrote a median of 15 analysis scripts per campaign, and in 89 of its 93 campaigns it fitted its own regression model of dependency on the reference lines. Both routes confirm where the advantage over the agent alone comes from: measured data, read for the line in front of it.

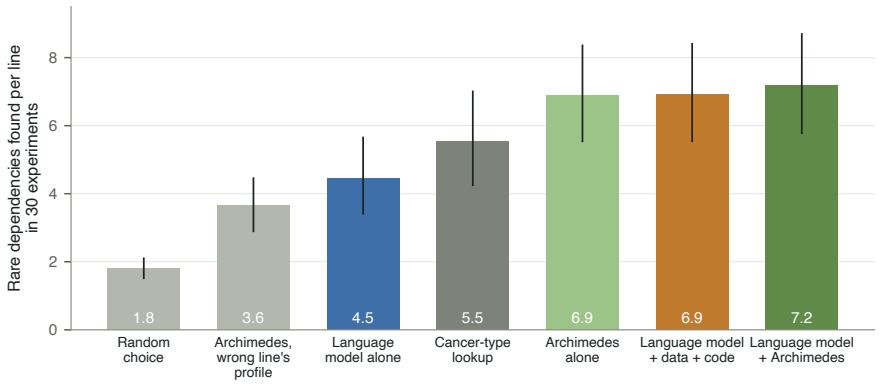

Figure 3. Every arm on the same 31 lines. Mean rare dependencies found per line within 30 experiments. Whiskers are 95% bootstrap intervals over lines and reflect how much lines differ from one another; the comparisons in the text are paired by line. Gray: reference arms with no language model. Blue: the language model alone. Orange: the language model with data and code. Green: arms that use Archimedes.

Archimedes delivers that yield faster, more reproducibly and with a higher floor (Figure 4).

Time. A campaign took 2.8 minutes with Archimedes and 6.3 minutes with data and code, 2.2 times as long (208 seconds more per campaign, 95% CI 139 to 289; Archimedes faster on 29 of 31 lines). The longest campaign ran 7 minutes with Archimedes and 38 minutes with data and code. Per hour of agent time, that is 152 discoveries against 66.

Reproducibility. With Archimedes, two campaigns on the same line shared 71% of their experiments. With data and code they shared 38% (difference 33 points, 95% CI 28 to 37), no more than the agent alone (39%). Asked the same question twice, the agent with Archimedes proposes largely the same campaign; the data-equipped agent writes a different analysis and proposes a different one. The run-to-run standard deviation in discoveries was 0.60 with Archimedes, against 1.22.

A higher floor. The worst of three campaigns found 6.7 discoveries with Archimedes against 5.7 with data and code, a difference of 0.94 per line (95% CI 0.26 to 1.65). The best of three was the same in both arms (7.8 and 8.0).

No preparation. The comparison also began after its hardest step. The agent was handed the right data, chosen and cleaned, with the test lines removed. A lab asking a new question does not start there. It has to know which data bear on the question, obtain them, and keep what it is testing out of what it learns from. With Archimedes that step is already inside the model.

	Model alone	Model + data + code	Model + Archimedes
Discoveries, summed over 31 lines	138.3	214.7	223.0
Difference from model + Archimedes, per line	−2.73 (−3.68 to −1.77)	−0.27 (−0.97 to +0.46)	–
Minutes per campaign	3.2	6.3	2.8
Discoveries per hour of agent time	83	66	152
Experiments shared by repeat campaigns	39%	38%	71%
Run-to-run standard deviation, discoveries	0.84	1.22	0.60
Worst of three campaigns, discoveries	3.8	5.7	6.7
Best of three campaigns, discoveries	5.3	8.0	7.8

Table 2. The three arms on all 31 lines, as run. Per-line quantities are means over lines. The data-and-code arm exceeds the model alone by 2.46 discoveries per line (95% CI 1.63 to 3.32).

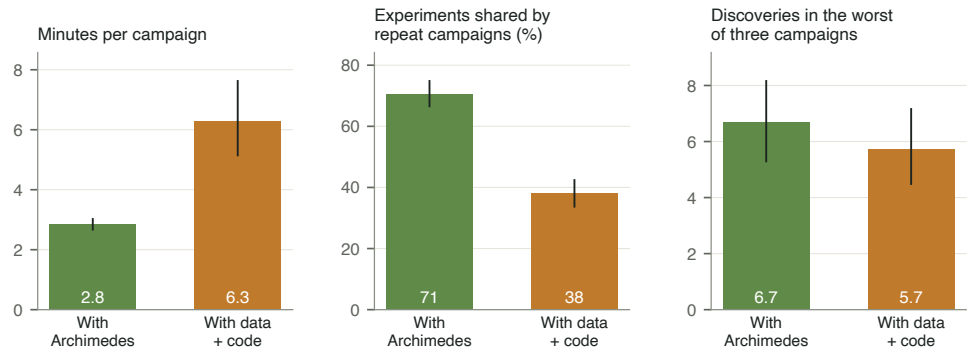

Figure 4. Archimedes delivers the same yield faster, more reproducibly and with a higher floor. Mean over the 31 lines, with 95% bootstrap intervals over lines. Green: language model + Archimedes. Orange: language model + data + code.

Every number re-derived from the raw data

The audit re-scored all 8,370 knockouts in the 279 campaigns against the raw DepMap 26Q1 file and found no mismatches. All of its leakage and integrity checks passed; the one thing it flagged was the model fallback handled by the two analyses above. Every data-and-code campaign completed with 30 distinct knockouts on Claude Opus 5.5, with no fallback, no call to Archimedes and no access outside its workspace. The independent re-analysis reproduced the totals, the per-line differences, the win counts, the significance tests and the experiment-17 crossing exactly, and the confidence intervals to within bootstrap precision. Recomputing the 10% rarity threshold from every line in DepMap 24Q4 leaves the ratio between the arms unchanged.

Discussion

An agent that chooses its experiments with Archimedes finds 57% more of what it is looking for in the same campaign, or the same result in a little over half the experiments. The effect was specified in advance, it is large, it is consistent across lines, and it survives every check we applied to it.

The lines in this study are the ones where prior knowledge is thinnest: newly released, mostly rare and pediatric cancers. That is where a model of measured biology should matter most, and it is where a lab has the least to go on. The frontier language model, a strong scientific agent, did worse on its own than a table of dependency frequencies. With Archimedes, the same agent outperformed both.

Against an agent equipped with the same data, Archimedes delivers the yield in less than half the time and with nearly twice the agreement between repeat campaigns. Matching it took the right data, already chosen and prepared; a sandbox; more time; and an analysis rebuilt in every campaign, with a different plan each time. A scientist planning a campaign needs the same answer on Tuesday as on Monday, needs it in minutes, and should not have to know in advance which data hold it. Archimedes is that analysis done once, tested on lines it never saw, and returned in one call.

For a lab, the unit that matters is the experiment. Thirteen fewer experiments in every thirty is budget and months returned on each target campaign, and it compounds across a portfolio. This is the role Archimedes plays inside an autonomous lab: it proposes what to test next, updates on every result, and leaves the agent, and the scientist approving the work, to decide.

Scope. The study is a simulated campaign scored against real held-out measurements: cell lines and single-gene knockouts from DepMap. Archimedes draws here on one family of public data, and the data-equipped agent was handed that same data ready to use. The natural extensions are campaigns run at the bench, experiment types beyond gene knockouts, and questions that draw on many data sources at once, where

assembling the data inside each campaign stops being practical.

Data and competing interests

The study uses DepMap Public 24Q4 and 26Q1 (Broad Institute, CC BY 4.0) and Project Score 2021. Per-line results are in Table 3 and the prompts are in the appendix. Archimedes is developed by Heureka Labs, and the author is a co-founder of Heureka Labs.

Appendix

Line	Disease	Rare deps.	Model alone	Model alone (per protocol)	Model + data + code	Model + Archimedes	Archimedes alone	Lookup	Random
ACH-000197	T-Lymphoblastic Leukemia/Lymphoma	85	6.3	8.0	8.0	5.7	5	3	1.8
ACH-000299	Acute Myeloid Leukemia	135	10.3	10.5	10.0	13.3	13	12	2.9
ACH-000981	T-Lymphoblastic Leukemia/Lymphoma	32	8.7	8.0	8.0	8.0	8	8	0.7
ACH-001021	Hepatoblastoma	51	6.0	6.5	12.7	10.0	9	4	1.3
ACH-001349	Embryonal Tumor	125	5.3	–	10.7	12.0	10	6	2.9
ACH-001376	Pancreatic Adenocarcinoma	148	2.7	2.7	7.3	7.7	9	6	3.8
ACH-001428	Ewing Sarcoma	74	3.7	3.7	11.7	12.7	13	9	1.4
ACH-001465	Neuroblastoma	98	3.0	3.0	7.0	8.3	7	10	2.5
ACH-002060	Acute Leukemia, Ambiguous Lineage	118	3.7	4.0	5.7	5.0	8	4	2.6
ACH-002142	Osteosarcoma	17	1.0	1.0	1.0	2.7	2	1	0.5
ACH-002433	Osteosarcoma	135	9.3	9.3	9.0	11.7	10	8	3.2
ACH-002688	Osteosarcoma	104	5.7	5.7	6.3	7.7	7	8	2.6
ACH-002708	Liposarcoma	87	0.7	0.7	4.3	6.3	5	6	1.9
ACH-002801	Nerve Sheath Tumor	75	0.7	0.7	2.0	3.3	3	4	2.0
ACH-002923	Neuroblastoma	74	6.7	6.7	11.3	6.3	6	7	2.1
ACH-003064	Other	20	0.3	0.0	1.0	0.0	0	0	0.6
ACH-003065	Embryonal Tumor	97	7.0	7.0	9.3	9.7	11	8	2.2
ACH-003171	Osteosarcoma	31	0.0	0.0	5.7	4.0	5	3	1.3
ACH-003176	Osteosarcoma	90	6.0	6.0	11.3	12.3	11	3	2.1
ACH-003178	Osteosarcoma	63	2.0	2.0	6.3	7.3	5	8	0.9
ACH-003184	Leiomyosarcoma	29	2.7	2.7	4.0	2.7	4	2	0.7
ACH-003185	Osteosarcoma	65	4.0	4.0	4.3	5.0	5	3	1.7
ACH-003186	Undifferentiated Pleomorphic Sarcoma	39	2.0	2.0	1.3	2.3	2	1	0.8
ACH-003189	Liposarcoma	55	3.3	3.3	7.0	8.3	9	5	1.4
ACH-003190	Undifferentiated Pleomorphic Sarcoma	42	3.3	3.3	6.3	4.0	5	3	1.1
ACH-003191	Giant Cell Tumor of Bone	50	2.7	2.7	2.7	7.0	3	3	1.8
ACH-003273	Acute Myeloid Leukemia	121	5.7	5.0	7.7	11.0	10	9	3.1
ACH-003410	Chordoma	59	5.3	5.5	6.3	2.7	3	3	1.1
ACH-003573	Osteosarcoma	50	2.0	2.0	0.7	0.3	1	0	0.9
ACH-003587	Osteosarcoma	26	3.0	3.0	5.0	6.0	6	5	0.6
ACH-003590	B-Cell Acute Lymphoblastic Leukemia	145	15.3	15.3	20.7	19.7	19	20	3.3

Table 3. Per-line results. Discoveries within 30 experiments. "Rare deps." is the number of rare dependencies the line has among the genes Archimedes ranks. Language-model arms are means of three campaigns (per protocol: of the unaffected campaigns).

Prompts

All arms. *You are running an experimental campaign on an uncharacterised cancer cell line, "line X". Goal: find genes that line X depends on — knocking them out reduces its fitness (CRISPR gene effect < -0.5) — and that are RARE dependencies across cancer cell lines (a dependency in fewer than 10% of lines). Each such gene you knock out counts as one discovery. Common dependencies do not count. You have a budget of 30 knockout experiments (run_knockout). Each reveals the measured gene effect in line X. Maximise the number of discoveries within the budget. Learn about line X with describe_line, get_expression, get_copy_number and population_dependency_rate (these are free). Use all 30 experiments, then call submit_findings. Work autonomously; do not ask questions.*

Added in the Archimedes arm. *You also have Archimedes, a model trained on measured dependency data from about 1,100 cancer cell lines. Every knockout must be one of Archimedes' current top 5 hypotheses: call archimedes_rank before each experiment and choose among its first 5 genes, using your own judgement and the free omics tools. The list updates after every result; a knockout outside it is refused without spending budget. archimedes_explain gives its evidence for one gene.*

Added in the data-and-code arm. *You also have the public DepMap data and a Python environment (run_python; pandas, numpy, scipy, scikit-learn). The workspace holds measured CRISPR gene effects for about 1,100 previously screened cancer cell lines, RNAi data, their omics and annotations, and the same omics for line X; read README.txt first. Analyse it however you like to choose your experiments. You decide which experiments to run.*