Towards Evaluating and Training an AI Scientist

Mehdi Soleimanifar
mehdi@caltech.edu

June 4, 2025

Abstract

In this preliminary write-up, we propose a framework for evaluating and training large language models in scientific reasoning. We introduce lightweight, interactive environments that simulate core aspects of scientific inquiry across diverse domains. Each such environment, which we refer to as a Tiny Lab, defines objects, measurable properties, and operators that generate complex, emergent behaviors. For example, a genomics Tiny Lab might model DNA sequences, mutations, and motif statistics. Similar to video games in reinforcement learning, Tiny Labs offer controlled, modular settings for systematic experimentation. Within these environments, AI systems can propose experiments, observe outcomes, revise hypotheses, and iterate—mirroring the scientific process. We show that Tiny Labs can distinguish shallow heuristics from deeper, multi-step reasoning in frontier models, with and without chain-of-thought prompting.

Contents

1 Going from Chat to Lab

The scientific method is one of humanity's greatest intellectual achievements, shaping centuries of discovery by guiding how we ask questions and seek answers about the universe. We now face an extraordinary possibility as we begin to create AI systems that can engage in scientific thinking, extending our ability to explore, experiment, and learn about the world in ways not possible before.

Modern large language models (LLMs) exhibit striking capabilities. Trained on vast collections of human knowledge, they excel at factual recall and prediction, particularly within their training distribution. Recent research also shows that with the right prompting or fine-tuning, such as chain-of-thought techniques, these models can also perform surprisingly sophisticated forms of reasoning.

When it comes to *scientific* reasoning, two views emerge. One holds that scientific thinking is a natural by-product of general cognitive ability: as models scale and develop deeper abstractions, stronger pattern recognition, and better long-horizon context management, scientific reasoning will "come along for the ride," much as language fluency and commonsense inference have already emerged.

The other view holds that scientific reasoning requires more than passive exposure to text. Models must engage actively by forming hypotheses, running experiments, discovering causal relationships, and constructing conceptual frameworks. Observing science is not enough; doing science is what builds the underlying competencies.

Human development points to a hybrid model where innate general intelligence provides the foundation, but true scientific ability *emerges* only through active engagement with scientific practice. By analogy, we argue that LLMs will need not only sufficient capacity, but also interactive, experiment-driven training to develop genuine scientific reasoning skills.

From this perspective, most current reasoning benchmarks fall short of evaluating genuine scientific capabilities. Their static, one-shot format freezes inquiry at a single moment, overlooking the dynamic, iterative process that

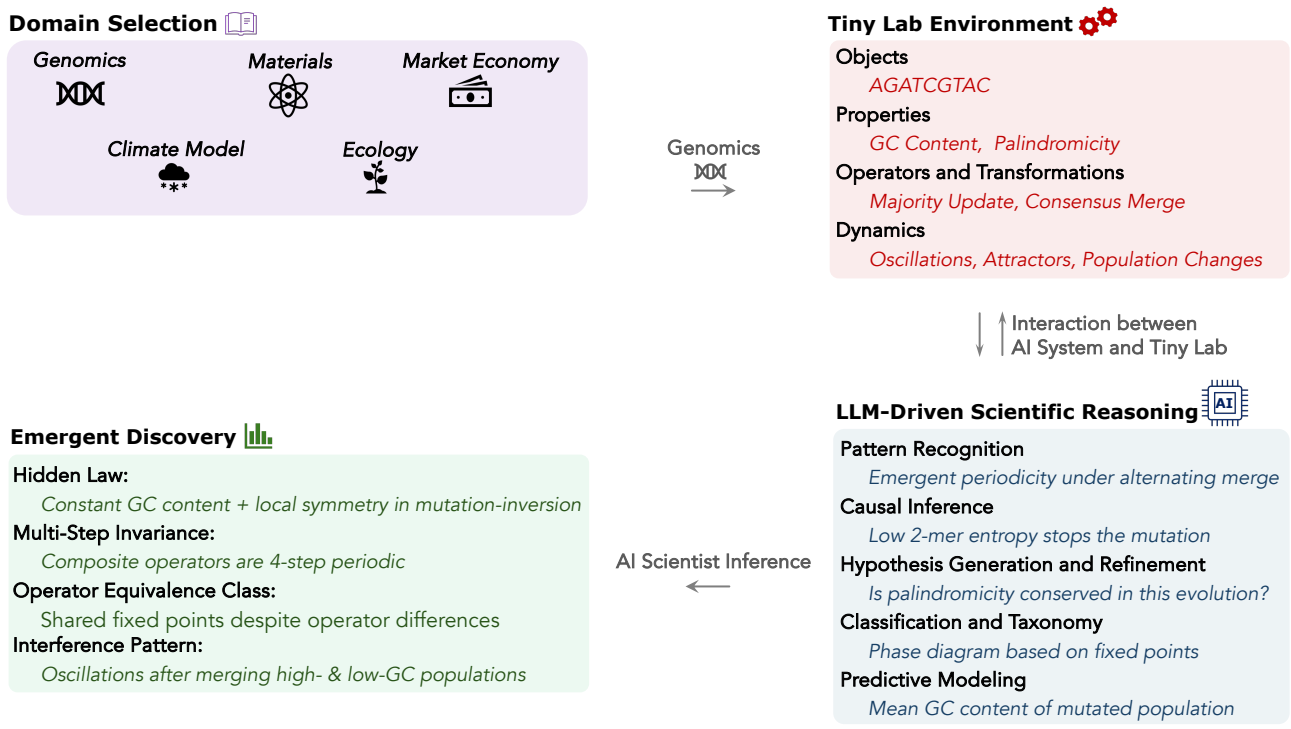

Figure 1: **Tiny Lab framework for probing scientific reasoning.** A domain-specific goal (e.g., in genomics, climate, or economics) is instantiated into a modular Tiny Lab defined by objects, properties, and a set of operators inducing dynamics. An AI system interacts with the lab by designing experiments, analyzing outcomes, and iteratively refining hypotheses. This setup supports a range of scientific reasoning capabilities, such as pattern recognition, causal inference, and predictive modeling, and can reveal unexpected structural features of the environment.

defines real research. These benchmarks also bypass complementary cognitive skills essential to scientific work such as system-level reasoning, multi-scale abstraction, taxonomic classification, uncertainty management, and strategic allocation of experimental effort.

Beyond evaluation, interactive benchmarks offer a path to better *training*. Static tests function primarily as exams rather than classrooms. A useful analogy comes from reinforcement learning breakthroughs in domains like Atari, where progress emerged from training in simplified, yet expressive, video-game environments that captured core features of the real world—sparse rewards, long-horizon planning, and the integration of multiple skills. In much the same way, large language models are unlikely to develop deep scientific reasoning through quizzing alone. They need training grounds that foster the habits of inquiry, experimentation, and reflection that science demands.

2 Tiny Labs: Scientific Sandboxes

To accelerate progress, we advocate for the development of carefully constructed toy problems: small, interactive environments rich enough to demand original scientific thinking and flexible enough to serve as experimental sandboxes. We will refer to these environments to as *Tiny Labs*.

Well-designed Tiny Labs offer several core advantages:

- *Simplicity and Speed*: They are based on simple underlying rules or programs that can be simulated rapidly, enabling extensive interaction within feasible training times.
- *Rich Emergent Behavior*: Despite simple rules, these environments can exhibit complex, non-obvious dynamics and emergent properties, creating genuine opportunities for discovery.
- *Clean Evaluation*: Because the underlying ground truth about the system's rules and behavior is known, we can rigorously and objectively evaluate an AI's scientific performance.

To accelerate development, we aim to facilitate (semi-)automated generation of scientific sandboxes. Automation would enable a vast supply of environments for targeted training and testing, and systematic variation of complexity to study scaling behavior.

To make this possible, Tiny Labs must satisfy two additional criteria:

- *Tunable Complexity*: Tiny labs should have clear parameters and well-defined measures of complexity, allowing difficulty to be adjusted systematically.
- *Composable Framework*: Each environment should be formally specified by a modular set of generation rules, enabling environments to be composed, instantiated, and verified automatically.

3 How to Design Tiny Labs

We now illustrate how these criteria can be satisfied through a simple and general framework. We focus on domains that naturally lend themselves to mathematical structure, including examples inspired by DNA sequence modeling, simple market economies, and climate models. However, the underlying principles apply much more broadly.

In this framework, a Tiny Lab consists of four ingredients:

- A space of *objects* to explore,
- *Properties* that can be measured or inferred,
- *Operators* that transform objects,
- And *dynamics* that emerge from applying operators over time.

We now describe each component more precisely.

Objects and Properties

We begin with a set of *objects* S. Each object $s \in S$ exhibits a collection of measurable *properties*. Each property $f : S \to [0, 1]$ maps objects to real values, typically normalized to $[0, 1]$:

Multidimensional properties $f = (f_1, f_2, \ldots, f_d)$ can be represented as tuples of one-dimensional properties.

Properties should be simple to compute, requiring only a few lines of code.

In addition, we require that properties are *easy to sample*: given a target property value p and a small tolerance ε, we should be able to sample objects s such that $|f(s) - p| \leq \varepsilon$. This can be naturally generalized to sampling based on a subset of properties whose preimages overlap substantially.

To sidestep computational bottlenecks, we allow approximate sampling with a nonzero ε and relaxed (potentially non-uniform) distributions. Beyond computational convenience, this relaxation also mirrors the noise and uncertainty inherent in real scientific experiments.

Property Space

Properties define a *property space*, where each object s corresponds to a point $(f_1(s), f_2(s), \ldots, f_d(s))$.

A natural notion of *distance* arises by choosing a metric, such as Euclidean distance or Hamming distance. This allows us to define *neighborhoods*: given an object s, its neighbors are objects within a small distance according to the chosen metric.

Locality becomes important when we define how objects change: many transformations act *locally*, making small moves within property space rather than jumping arbitrarily.

In principle, we can set up the property space so that a subset of properties uniquely specifies each object. When this holds, locality acquires a particularly intuitive meaning—as we will see in concrete examples.

Operators and Transformations

Operators govern how objects evolve or interact. Several types of operators are useful:

Evolution operators: Map an object to another $T : S \to S$, often through small, local moves. Evolution may push objects toward stable configurations (attractors or fixed points) through a sequence of local steps.

Composition operators: Combine multiple objects to produce a new one $C : S^k \to S$. Composition can model recombination (e.g., in genetics) or history-dependent rules, where the next state depends on previous states.

Operators may also exhibit additional structure:

Property-dependent behavior: Operators can behave differently depending on an object's properties, introducing dynamic regime switching and phase-like transitions.

Stochasticity: Operators may introduce randomness, sampling among possible outcomes rather than applying a fixed rule deterministically.

Together, these mechanisms support a wide range of emergent behaviors and enable systems to explore complex, adaptive dynamics.

Dynamics

Applying operators iteratively generates *dynamics* in property space. Even at the level of a single object, a wide range of behaviors can emerge:

Attractors: Objects converge to fixed points through local updates, with the specific fixed point depending on the object's properties.

Oscillations: Operators can induce oscillatory behavior in an object's property values by adaptively shifting them toward complementary targets.

Bifurcations: Small changes in initial conditions or parameters cause sudden shifts in behavior, creating chaos or phase transitions between dynamic regimes.

When composition operators and population-level interactions enter the picture, additional patterns appear:

Branching structures: Repeated composition can lead to tree-like growth of objects or expansion of populations.

Population dynamics: In environments with many interacting objects, recombination and composition create evolving distributions rather than single trajectories.

Finally, *stochasticity* can layer on further complexity, producing noisy exploration and uncertainty.

Jointly, these simple ingredients generate the diverse behaviors that make Tiny Labs scientifically interesting.

Developing Core Scientific Capabilities

By interacting with these controlled environments, AI systems can be trained and evaluated on a wide range of emergent scientific competencies, including:

1. **Pattern Recognition and Trend Analysis**: Detecting regularities, recurring structures, and changes in system behavior across time or parameters, spotting both steady trends and subtle anomalies.
2. **Causal Inference**: Determining the cause-and-effect relationships between interventions (e.g., changing a parameter) and outcomes.
3. **Hypothesis Generation and Testing**: Systematically formulate testable explanations for observed phenomena, then design and run experiments to validate, refine, or refute those hypotheses.
4. **System Dynamics Understanding**: Grasping how interacting components lead to complex, system-level behaviors like stability, oscillation, chaos, or phase transitions.
5. **Classification and Taxonomy**: Organizing observations or system states into meaningful categories based on shared properties.
6. **Predictive Modeling**: Build internal or external models that forecast future system states or outcomes from current conditions and known dynamics, and continually update those models as new data arrive.

We next present concrete instantiations of the general framework described above for creating Tiny Labs. Each example highlights how a small set of objects, properties, operators, and dynamics can create an environment for scientific exploration, including training and testing the core capabilities stated in this section.

4 Example: Sequence Modeling Lab

We begin with a miniature scientific world where the basic building blocks are sequences of letters. Inspired by molecular biology, this Tiny Lab captures important features of biological systems, including sequence structure, mutation dynamics, and combinatorial richness, while remaining simple and tractable.

Objects. In this example, the set of objects $s \in \{\mathsf{A}, \mathsf{G}, \mathsf{C}, \mathsf{T}\}^N$ consists of all sequences of length N made from the four-letter alphabet or "bases" $\{\mathsf{A}, \mathsf{G}, \mathsf{C}, \mathsf{T}\}$. For instance, when $N = 5$, a possible sequence could be $s =$ "AGTCA" suggestively chosen as a short fragment of a DNA molecule.

Properties. Each sequence admits a collection of properties which assign real values, normalized in $[0, 1]$, to each object. These properties are selected to exhibit complex but controllable structure, and include:

Property	Description
GC Content	Fraction of bases that are G or C.
Weighted Count	Weighted average of the number of single-base contributions.
k-Masked Alternation	Normalized number of changes between successive bases in the subsequence formed by positions $0, k, 2k, \ldots$
Palindromicity	Similarity between a sequence and its reverse complement.
k-mer Entropy	Diversity of short subsequences (k-mers) within the sequence.
Hamming Distance	Number of differences compared to a fixed reference sequence.
Interaction Energy	Energy based on pairwise interactions between neighboring bases.

Table 1: Descriptions of sequence properties used in the benchmark

Each property can be computed easily, typically requiring only a few lines of code. They are also amenable to approximate sampling for modest sequence lengths ($N \leq 1000$) and tolerances $\varepsilon \geq 0.1$.

Together, these properties embed the space of sequences into a low-dimensional property space with interpretable structure, tunable complexity, and rich opportunities for dynamical exploration.

Operators. The behavior of sequences is shaped by operators, which act either on individual sequences or pairs of sequences. These fall into two broad categories: evolution dynamics and composition rules.

A summary of the available operators is shown below.

Operator	Type	Description
Majority Update	Evolution	Each base updates toward the majority of its neighbors.
GC-Biased Mutation	Evolution	Random mutations favor increasing GC Content.
Palindrome Pull	Evolution	Sequence evolves toward greater palindromic symmetry.
Directed Evolution	Evolution	Adaptive mutations guide sequence toward attractors in property space, with stochastic exploration to avoid traps.
Alternating Merge	Composition	New sequence formed by alternating bases from two parent sequences.
Consensus Merge	Composition	At each position, if parents agree, keep base; otherwise, choose randomly.
Palindrome Sandwich	Composition	Concatenation of Parent 1, Parent 2, and the reverse complement of Parent 1.

Table 2: Types and descriptions of sequence transformation operators

Evolution operators transform sequences over time, inducing trajectories through property space. For instance, in Majority Update, each base updates to match the majority of its local neighborhood, smoothing out local variations and promoting uniformity. Other evolution operators, such as GC-Biased Mutation and Palindrome Pull, follow similar principles, as outlined in the table.

Another type of dynamic is the *Directed Evolution* process, where sequences adaptively mutate toward attractor points in property space. The strength of attraction can be tuned to allow either smooth convergence or sharp transitions between competing targets. To avoid becoming trapped in local minima, occasional random mutations introduce stochastic exploration. The basic flow is:

```
while not converged:
    compute current properties of sequence
    find nearest attractor in space of properties
    propose mutation that moves closer to attractor
    if mutation accepted:
        update sequence
    else if stuck:
        apply random mutation (stochastic exploration)
```

Finally, *Composition* operators combine two sequences into a new one. For example, Alternating Merge constructs a new sequence by alternating bases from each parent. Other composition rules, such as Consensus Merge and Palindrome Sandwich, allow hierarchical growth and the emergence of structural patterns from simpler building blocks.

5 Example: Macroeconomic Simulation Lab

Our second Tiny Lab represents a closed economy that unfolds in discrete *quarters*. Although deliberately stylised, the model captures a recognisable macroeconomic logic: firms hire workers to meet expected demand; households decide how much of their income and savings to spend (subject to shocks and interest-rate effects); the government conducts counter-cyclical fiscal policy; and a (currently passive) central bank sets the policy rate. Through this web of feedback loops, the economy generates business-cycle-like fluctuations in GDP, unemployment, inflation, and public debt—providing a test-bed for evaluating the performance of current AI systems.

Objects. The simulator contains four classes of agents:

1. **Households** $h_i,\ i = 1, \ldots, N_H$: Each household holds a cash balance m_i, supplies labor, receives a wage w_i if employed, and may receive lump-sum transfers z_i from the government. Households pay taxes $T_i = \tau \cdot w_i$, where τ is a fixed income-tax rate, and choose their consumption level C_i each quarter. Their consumption decision follows a rule that depends on disposable income and prior savings, and is modulated by an idiosyncratic uncertainty shock and a real interest rate adjustment.
2. **Firms** $f_j,\ j = 1, \ldots, N_F$: Each firm manages an inventory I_j, sets a price p_j, and hires workers L_j based on expected demand. Wages are fixed across firms. Firms update their markups μ_j through a noisy adjustment process. Production is proportional to labor employed and scaled by a time-dependent **innovation factor** A_t, which captures economy-wide productivity improvements: $A_{t+1} = A_t(1 + g), \quad A_0 = 1$. Thus, output per firm is: $A_t \cdot \varphi \cdot L_j$. This introduces steady, innovation-driven growth into the economy.
3. A single **Commercial Bank** tracks its equity and assets and adjusts credit conditions based on the unemployment rate. Lending and credit allocation mechanisms are placeholders for future extensions.
4. The **Government** collects tax revenue from households, issues lump-sum transfers, and tracks its outstanding debt B_t. Transfers are computed automatically each quarter as a fraction of the previous quarter's GDP, scaled up in periods of high unemployment. The income-tax rate τ is fixed. Debt increases by the total amount of transfer spending each quarter. The **Central Bank** maintains a constant policy interest rate r_t, which affects household consumption behavior, but does not currently influence firms or credit markets.

Properties. Each agent maintains internal state variables: households store cash, employment status, income, and consumption; firms track inventory, prices, markups, and workforce; the bank records equity and credit tightness; and the government maintains debt and spending ratios. These microstates aggregate into economy-wide indicators such as:

$$\text{GDP}_t = \sum_j Y_{j,t}, \qquad \text{Unemployment}_t = 1 - \frac{1}{N_H} \sum_i \mathbf{1}[\text{employed}_i], \qquad \pi_t = \frac{\bar{p}_t - \bar{p}_{t-1}}{\bar{p}_{t-1}},$$

where $Y_{j,t}$ is the output of firm j and $\bar{p}_t$ is the average price across firms. A set of sixteen such macro-indicators is collected over the past four quarters and flattened into a 64-dimensional observation vector provided to the learning agent.

Operators: Quarterly Cycles. Each quarter proceeds in four phases:

1. **Policy Update.** The central bank sets the policy interest rate r_t, held constant over time. The government computes its total transfer spending as:

$$G_t = g \cdot (1 + 2 \cdot u_{t-1}) \cdot \text{GDP}_{t-1},$$

where g is a spending ratio and u_{t-1} is the previous quarter's unemployment rate. The income-tax rate τ remains fixed throughout the simulation.

2. **Firms Update.** Each firm forecasts next-quarter demand:

$$\hat{D}_{j,t} = \alpha \cdot \text{sales}_{j,t-1} + (1 - \alpha) \cdot \left(\frac{1}{2} I^{\max} - I_{j,t} \right),$$

where α is a smoothing parameter and $I^{\max}$ is the inventory capacity. The firm adjusts its workforce to match the target labor demand:

$$L_j^\star = \left\lceil \frac{\hat{D}_{j,t}}{A_t \cdot \phi} \right\rceil,$$

where A_t is the innovation factor and ϕ is baseline productivity per worker. Output is then:

$$Y_{j,t} = A_t \cdot \phi \cdot L_j,$$

and unsold goods are added to inventory. The firm's markup evolves via:

$$\mu_{j,t} = \text{clip}\left(\mu_{j,t-1} + \gamma(\mu^\star - \mu_{j,t-1}) + \varepsilon_t,\ 0.10,\ 0.50\right), \qquad p_{j,t} = \frac{w^\star}{A_t \cdot \phi}(1 + \mu_{j,t}),$$

where ε_t is Gaussian noise and $w^\star$ is the fixed wage.

3. **Households Update.** Households compute their disposable income $y_i = w_i + z_i - T_i$, and choose consumption using:

$$C_i = c_1 \cdot \left(y_i + c_2 \cdot m_i^{\text{prev}}\right) \cdot \varepsilon_i^{\text{unc}} \cdot (1 - 0.5 \cdot r_t),$$

where c_1 and c_2 are consumption parameters, $\varepsilon_i^{\text{unc}}$ is an idiosyncratic shock, and m_i^{prev} is prior cash. Remaining cash is updated as $m_i = m_i^{\text{prev}} + y_i - C_i$. Households allocate their consumption budget across firms via a softmax function favoring lower-priced goods.

4. **Government Budget.** Taxes are collected from households. Transfer payments G_t are disbursed across the population (with greater amounts to the unemployed), and total government debt increases as:

$$B_t = B_{t-1} + G_t.$$

(Tax revenue is tracked but does not reduce the debt in this version.)

Emergent dynamics. A negative sales shock lowers firms' demand forecasts, prompting lay-offs that raise unemployment and reduce household income. Lower income suppresses consumption, which validates firms' pessimistic expectations. That is, a recessionary feedback that continues until larger government transfers and falling prices rekindle demand. Because prices adjust only gradually through the noisy markup process, short-run inflation can turn negative in slumps and positive in booms. These interacting mechanisms are sufficient to generate rich, endogenous cycles, which are complex enough to challenge an LLM-based agent, yet lightweight enough to run on a laptop CPU. A typical example of the simulated dynamics is given in the following plot:

6 Preliminary Evaluations of Sequence Modeling Lab

In this section, we present preliminary small-scale results from evaluating AI systems within our Tiny Sequence Modeling Lab. The aim is to illustrate how these models engage with a unified environment across a range of interdependent scientific scenarios. The framework's modular and compositional structure proves especially valuable in supporting this evaluation while maintaining coherence.

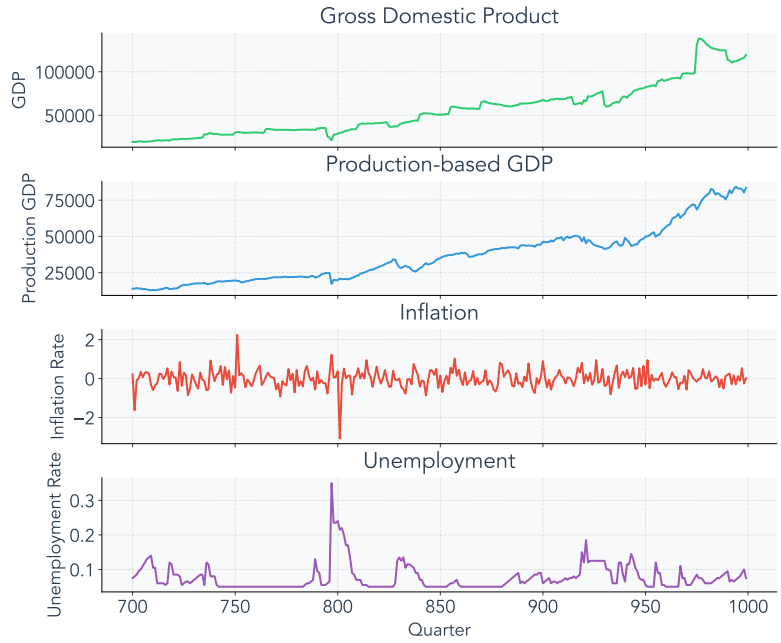

Figure 2: A typical dynamics of the properties in Macroeconomic Simulation Lab

6.1 Inferring Hidden Properties with Planned Measurements

In our first experiment, we introduce an interactive setup where the AI model acts as a scientist attempting to reverse-engineer an unknown property defined over DNA-like sequences $S = \{\mathsf{A}, \mathsf{G}, \mathsf{C}, \mathsf{T}\}^N$. The environment serves as a "black-box" system: the model can propose sequences and receive evaluations of an unknown function $f : S \to [0, 1]$, but is not directly told what f computes. The goal is to formulate and refine hypotheses about the nature of f using iterative experimentation.

Each round of interaction consists of:

1. *Prompt Context*: The model is reminded of the setting—fixed sequence length (e.g., 10), valid bases (A, C, G, T), and the availability of a function f(s) whose output is accessible through sampling or querying.
2. *Sampling Request*: The model proposes a target score (e.g., "`SAMPLE 0.8`"), and the environment returns a batch of sequences $\{s_i\}$ that achieve that score ($f^{-1}(s_i) \approx 0.8$) under the hidden function.
3. *Query Request*: The model proposes a batch of sequences $\{s_i\}$, and the environment returns the property value $f(s_i)$ for each sequence. Alternatively, the model can ask for a uniformly random batch of sequences and their evaluated properties.
4. *Evaluation Phase*: The model observes the sampled or queried sequences and their scores.
5. *Hypothesis Update*: Based on observed patterns, the model is prompted to state its best-guess functional description of f.

Example: Palindromicity Discovery. Below is an example from the Open AI o4-mini model discovering the hidden function Palindromicity $f(s) = \text{Palindromicity}(s)$, defined as the fraction of positions where the sequence matches its reverse complement. The reverse complement is obtained by reversing the sequence and swapping each base with its pair: A $\leftrightarrow$ T, C $\leftrightarrow$ G). The initial prompt is similar to:

Prompt for inferring unknown properties

```
You are a scientist inferring an unknown property f(s) in [0,1] of DNA sequences s with
length 15 over alphabet {A,G,C,T}.  You will have exactly 5 rounds of interaction.  In
each round, you must choose ONE of these two options:
1.  Say 'SAMPLE value' to receive 10 sequences with f(s) = value (up to some error),
where value is a number between 0 and 1.
2.  Provide exactly 10 sequences (one per line) to learn their f(s) values.
```

```
After seeing all results from 5 rounds, you will state your final hypothesis.
```

An example round where the hidden property is the Palindromicity looks like:

Interaction between Tiny Sequence Lab and AI model

```
LLM Response:
SAMPLE 0.5

Sequence Model Response:  Sampling sequences with value around 0.5.
Evaluations:
AGGGGTCCCC → 0.6
GGCCTACGAC → 0.6
ATTGGCCAGG → 0.6
ATGAGCGCGT → 0.6
CGGGATTACG → 0.6
```

The final hypothesis of the AI model is presented at the end of the specified rounds, as illustrated in the example below.

AI model's final response

```
Final hypothesis:
f(s) is computed as the "reverse-complement match fraction" of the 10-mer.  In other
words, if you form the reverse complement r of s (using A<->T and C<->G), then f(s)
equals the fraction of positions i for which s[i] exactly equals r[i].  For example,
for s = GCTAGCTAGC one finds that every position matches its reverse-complement partner
(f(s) = 10/10 = 1.0), while for s = CAGGGTCGGT only 2 out of 10 positions agree (f(s) =
0.2).  This interpretation exactly reproduces the provided evaluations:
- GCTAGCTAGC: every position matches its reverse{complement → f(s) = 1.0
- CAGGGTCGGT: 2 matching positions → f(s) = 0.2
- CAAGGCGTCG: 6 positions match → f(s) = 0.6
```

Distinguishing shallow heuristics from deeper, multi-step reasoning. We evaluated *Tiny Sequence Modeling Lab* on a suite of hidden sequence properties designed to distinguish shallow heuristics from deeper scientific reasoning. These properties range from local statistics (GC Content) to global structure (Palindromicity) and implicit comparisons (Hamming distance). For alternating patterns, we introduced periodic masking (e.g., every kth base), and varied the sequence length to test how complexity affects reasoning.

Table 3 shows that both models reliably identified GC Content. GPT-4.1 (with `temperature = 0.3`) found a qualitatively correct description of the weighted count property but otherwise consistently failed on more structured tasks, often proposing plausible but incorrect rules based on local or surface-level patterns. In contrast, o4-mini successfully inferred Palindromicity and Hamming distance—tasks requiring non-local reasoning or comparison to an implicit reference. It also solved the 1-masked alternation task—but only for short sequences (length 10). Performance degraded at longer lengths or higher masking periods, where the alternation signal becomes harder to detect, suggesting limitations in multi-step abstraction.

Hidden property	**GPT-4.1**	**o4-mini**
GC Content	✓	✓
Weighted count	(partially)	✓
1-masked alternation (length = 10)	✗	✓
1-masked alternation (length = 30)	✗	✗
2-masked alternation	✗	✗
Palindromicity	✗	✓
Hamming distance to reference	✗	✓

Table 3: Pass@3 results (✓= at least one of the top-3 hypotheses matches the hidden rule).

These results illustrate two things. First, the benchmark effectively distinguishes shallow, local heuristics from

more global, multi-step reasoning. Second, tuning the masking period k and sequence length provides a natural way to scale task difficulty.

6.2 Autonomous Exploration of Unknown Dynamics

We now examine a more complex setting in which the AI model must infer the governing dynamics of a system undergoing *Directed Evolution*, an iterative process where adaptive greedy mutations guide sequences toward convergence points, while stochastic perturbations help avoid local optima.

The system evolves in a two-dimensional property space defined by *GC Content*—the fraction of G and C bases—and *Palindromicity*—a measure of similarity between a sequence and its reverse complement. Sequences reliably converge to one of two dynamic attractors: a high-GC, high-Palindromicity region near $(0.8, 0.8)$, or a low-GC, low-Palindromicity region near $(0.4, 0.2)$. Due to the greedy, bit-flip nature of the updates, convergence typically results in tight two-cycles oscillating near these attractor points.

Moving beyond the previous task, where the model inferred a static hidden property from simple measurements, we now shift to a more open-ended, dynamic scientific setting. This expanded setup grants the model greater experimental agency: it may choose any 30-base initial sequence, select up to four measurable properties (including two new ones: `hamming_distance` and `entropy_score`), and strategically specify the trajectory length, up to 20 steps. No prior information about the nature of the dynamics is provided. The model's goal is framed as a scientific investigation—characterizing system behavior, identifying attractors, and inferring the underlying rules driving evolution.

We evaluated three language models—GPT-4.1-mini, o3-mini, and o4-mini—on their ability to perform iterative scientific inquiry in a synthetic dynamical system.

The prompt used in the evaluations is as follows.

Prompt for inferring unknown dynamics

```
You are an experimental scientist studying an unknown dynamical process that acts on
DNA-like sequences with sequence length 30 and alphabet A,G,C,T.

You may run experiments.  One experiment consists of:
1) Propose an initial sequence (30 chars).
2) Select a subset of MEASURABLES (listed below) to analyze.
3) Choose number_of_steps in {1,...,20}.
The lab will return the full trajectory with the requested measurables at each step of
the dynamical process up to number_of_steps.

MEASURABLES (random order)
- gc_content:  fraction of G/C bases
- palindrome_score:  fraction of matching bases when the sequence is compared to its
reverse complement (formed by swapping A<->T and C<->G and reversing).
- hamming_distance:  number of differences to a hidden reference
- entropy_score:  diversity of 2-mers within the sequence.

Goal:  Infer and explain how the dynamical process behaves, including:
(i) qualitative description of long-term behaviors or attractors,
(ii) the rule or objective it optimizes,
(iii) supporting evidence from your experiments.

You may conduct up to 6 rounds of experiments, but aim to minimize the number of rounds
by reasoning strategically.  In each round, provide a concise hypothesis, rationale, or
plan explaining your chosen experiment.  If this is your final experiment round, please
reveal your final answer based on the goal.  Wait for the lab to respond before the next
message.
```

While all models produced plausible experiments and exhibited some scientific reasoning, none succeeded in inferring the full rule governing the dynamics. Each observed two-cycle behavior and sampled trajectories reaching the low-GC/low-Pal attractor, but none entered or stabilized in the high-GC/high-Pal region. Consequently, while the low basin was well-supported by evidence, the high basin remained effectively invisible to all models—limiting their hypotheses and accuracy.

Aspect	GPT-4.1-mini	o3-mini	o4-mini
Experimentation	6 runs with varied inputs; broad but shallow coverage	1 run trajectory; limited but targeted	4 diverse long runs; best overall coverage
Trajectory Results	Reached low-GC/Pal basin; clear 2-cycle	Reached low basin; 2-cycle noted	Reached low basin; multiple 2-cycles observed
Hypothesis Quality	Incorrect: entropy-only explanation; ignored GC/Pal trends	Partial: entropy + GC/Pal; attractor misaligned	Qualitative match to low attractor; missed high basin
Use of Evidence	Overfit to entropy; neglected contradictory signals	Integrated some trends; incomplete synthesis	Strong evidence use; some inconsistency across runs
Strengths	Consistent probing; detected cycling	Noted multi-factor dynamics	Best structural reasoning; greedy updates described
Limitations	Failed to generalize; wrong objective	Weak coverage; shallow analysis	Missed high basin; conflicting hypotheses

Table 4: Model-by-model summary of experimental design, reasoning quality, and hypothesis accuracy in the Directed Evolution task.

An example response from `o4-mini` can be seen as follows.

Example response from o4-mini

```
LLM Response:
----- Experiment Response (run 1)-----
Type: experiment
Sequence: ATGCGTACGATTAGCCATCGATGCAATTGGC
Measurables:
- gc_content
- palindrome_score
- hamming_distance
- entropy_score
Steps: 20
Reasoning: Survey all four metrics over time from a random-start sequence to identify
which measurable(s) change under the process and reveal its optimization target and
dynamics.
------------------------------
RESULTS:
Step 0: ATGCGTACGATTAGCCATCGATGCAATTGGC,
Properties: 'gc_content': 0.5, 'palindrome_score': 0.2, 'hamming_distance': 0.733,
'entropy_score': 0.922

---Some steps are removed for presentation---
Step 20: AGGAGTACGATTAGCCATTGGGGCAATTGAG,
Properties: 'gc_content': 0.5, 'palindrome_score': 0.2, 'hamming_distance': 0.667,
'entropy_score': 0.913
----- Experiment Response (round 2 - 4) removed for presentation-----
Type: report
Sequence: None
Measurables:
Steps: None
Reasoning: None
------------------------------
Final Report from LLM:
Hypothesis: The unknown process is a greedy, two-objective hill{climber that alternates
every step between maximizing GC content and maximizing palindrome_score. It ignores
hamming_distance and 2-mer entropy. Because each step greedily improves only one metric
at a time, the system settles into a 2-cycle: one sequence is locally optimal for GC
content (higher GC, lower palindrome), the other for palindrome_score (higher palindrome,
lower GC). Long-term it oscillates between these two 'compromise' states rather than
converging to a single fixed sequence.
```

```
Evidence:
1) In all four experiments both GC content and palindrome_score trace out anti-phase,
sawtooth patterns (e.g.  Expt 1 toggles GC between 0.5/0.4 and palindrome between
0.2/0.4; Expt 3 from 0->0.5 GC while palindrome falls 1.0->0.067, then oscillates).
2) Hamming_distance and entropy_score quickly plateau or fluctuate within noise, showing
no consistent upward drift, so they are not being optimized.
3) No fixed point is reached; instead every run ends in a 2-cycle that alternates
between the sequence that is best for GC and the one best for palindrome_score.
```

In conclusion, the evaluated task in this section required more than data collection—it tested the ability to weigh conflicting trends, generalize across experiments, and form mechanistic explanations. These results highlight both the promise and current limitations of LLMs in performing full-loop scientific reasoning. A score-based evaluation of these models is summarized in the following table.

Evaluation Aspect	**GPT-4.1**	**o3-mini**	**o4-mini**
Experimental coverage (breadth & probe design)	4	2	5
Attractor identification (qualitative + numeric)	2	3	4
Objective inference accuracy	1	3	4
Use of evidence / justification	2	3	4
Overall scientific reasoning	2.5	3	4

Table 5: Scores (0–5) for the Directed-Evolution diagnostic–higher is better.

Simpler variants with hints. To better isolate sources of failure and assess the impact of guidance, we also tested simplified variants of the setup in which the prompt offered additional hints or examples.

The information revealed in the prompt plays a crucial role in shaping the model's behavior. Without explicit examples such as

```
Examples of sequence with palindrome_score = 1.0:  AAAAAAATTTTTTT, GGGGGGGCCCCCCC,
                                                   AGAGAGTACTCTCT
```

the models—both reasoning and non-reasoning—struggled to discover high-Palindromicity sequences on their own. Their explorations remained confined to low-palindrome regions and yielded hypotheses centered solely on GC Content. However, once provided with such examples, the models expanded their exploration meaningfully, covering a broader swath of the two-dimensional property space $[0, 1] \times [0, 1]$ and improving their ability to locate the convergence basins.

The model's performance also improved markedly when provided with a high-level hint about the structure of the dynamics—specifically, that the system evolves within a two-dimensional space defined by GC Content and Palindromicity:

```
The system roughly tends toward two different convergence points.
Your goal is to strategically design queries that help you effectively
span the ('gc_content', 'palindrome_score') space, so that you can
locate the location of these rough convergence points and the rules
governing the evolution.
```

With this additional guidance, GPT 4.1 was able—after five trials—to correctly identify both attractor regions and the oscillatory, two-cycle behavior near each. This success came despite occasional misinterpretations, including incorrect assumptions about the Palindromicity of certain sequences. Similarly, o4-mini deduced key features of the system: the greedy nature of updates, the approximate locations of both convergence points, and the emergence of two-cycle dynamics. It even approached a correct description of the update rule as being based on distance to the attractors.

Searching for relevant properties. In a variant of the experiment, we introduced 2-mer Entropy as an additional measurable, representing the diversity of 2-mers in the sequence. Although the prompt continued to

emphasize GC Content and Palindromicity, GPT-4.1 repeatedly prioritized Entropy Score, diluting its focus and failing to explore property regions near the attractors at $(0.8, 0.8)$ and $(0.4, 0.2)$. Crucially, Entropy did not reliably distinguish between the two basins, making it a poor guide for hypothesis formation.

By contrast, o4-mini focused on the relevant properties and successfully identified both attractors. Its final hypothesis closely matched the correct coordinates and captured the two-cycle dynamics. It also proposed a plausible mechanism of stabilization of a 13-base core with a single flipping base, and refined its strategy over time by seeding attractors and probing basin boundaries.

Example response from o4-mini

```
Hypothesis:
Based on the span of trajectories you've sampled, the system settles into one of two
small oscillatory attractors in (gc_content, palindrome_score)-space:
1.  A "high-GC" attractor oscillating between roughly
-(gc=0.857, pal=0.714)
-(gc=0.786, pal=0.857)
2.  A "low-GC" attractor oscillating between roughly
-(gc=0.286, pal=0.286)
-(gc=0.357, pal=0.143)
Key observations suggest a greedy, one-base-at-a-time hill-climbing on a composite
objective that rewards both GC-richness and self-complementarity.  At each step exactly
one nucleotide flips in order to locally improve (or at least not worsen) that composite
score.  Because the optimum lies between pure GC-maximization and perfect palindrome,
the system overshoots one metric on its way to improving the other, creating the small
two-state cycle rather than a fixed point.
Hypothesis of the underlying rule:  - Evaluate at each step all single-base mutations.
- Pick the mutation that gives the largest net gain in a weighted sum alpha x gc_content
+ beta x palindrome_score.
- Apply it, then repeat.
- Depending on the initial sequence, this converges to either the "high-GC" basin or the
"low-GC" basin, each manifesting as a 2-state oscillation due to the discrete nature of
single-base steps.
```

In summary, the models initially failed to uncover the correct dynamics when left to explore autonomously. Only with targeted prompts and explicit examples did o4-mini succeed in identifying both attractors and inferring the underlying mutation rule. This suggests that, while capable of scientific abstraction, current models still rely heavily on structured guidance and struggle with open-ended discovery without it.

7 Preliminary Evaluation of Macroeconomic Simulation Lab

Implementation details. As stated before, the environment simulates a closed economy over discrete time steps. It accepts a set of tunable macroeconomic parameters and outputs key economic indicators such as GDP, inflation, unemployment, and production. The system evolves deterministically based on these parameters.

Parameter	Value	Description
`r0`	0.02	Initial interest rate set by the central bank
`tau0`	0.2	Income tax rate set by the government
`g0`	0.25	Government spending multiplier
`kappa`	0.1	Sensitivity of inflation to demand pressure
`c1`	0.6	Consumption parameter (primary)
`c2`	0.4	Consumption parameter (secondary)
`phi`	1.0	Wage adjustment parameter
`markup`	1.2	Firm pricing markup
`initial_cash`	100	Initial cash endowment for the agent
`inventory_capacity`	150	Maximum goods that can be stored
`number_of_steps`	100	Simulation length in time steps

Table 6: Model parameters with typical values and descriptions.

The agent can propose a JSON experiment by modifying these parameters and choosing the number of time steps to run. The lab then returns the resulting macroeconomic trajectory.

Scientific objectives. The primary challenge is to assess whether a language model, given access only to a black-box interface for this simulation, can form testable hypotheses, perform targeted interventions, and build causal or structural understandings of the economy. Unlike standard reinforcement learning, no explicit reward is provided. The model must learn by observing how the system responds to parameter changes.

Key questions include:

1. Can the LLM discover which parameters most strongly influence inflation, unemployment, etc?
2. Can it deduce causal relationships (e.g., higher interest rates $\rightarrow$ lower inflation)?
3. Can it uncover non-obvious dynamics, such as inventory saturation or oscillatory cycles?

Prompt for causal discovery and scientific understanding of Macroecon Lab

```
You are an economist studying an unknown macroeconomic simulation.

Environment
You may run experiments.  One experiment consists of:
- Propose a set of macroeconomic parameters to change (see below).
- Choose number_of_steps ∈ {10, . . . , 200}.
The lab will return a summary of the simulation results.

Tunable Parameters
- r0:  initial policy interest rate (central bank)
- tau0:  income tax rate (government)
- g0:  base government spending as share of GDP
- kappa:  bank capital ratio
- c1, c2 { household consumption propensities
- initial_cash { household starting cash
- phi:  firm labor productivity
-markup:  firm markup over cost
- inventory_capacity:  firm inventory capacity

Workflow
- Hypothesis:  State a causal claim (e.g., "Lowering the policy rate will temporarily
boost GDP but raise inflation with a lag").
- Design experiment:  Choose parameter tweaks and simulation length that could falsify
your claim.
- Run experiment:  Send JSON with your experiment.
-Observe & update:  Explain whether the result supports or contradicts your hypothesis.
- Iterate:  Repeat until your graph is stable and explains the data.
```

Model comparison. We benchmarked five language models—`o3-mini`, `o4-mini`, `GPT-4.1`, `Claude Sonnet`, and `Claude Opus`—on the Tiny Macroecon Lab task suite. All models were evaluated using a structured rubric covering workflow discipline, experimental coverage, hypothesis depth, data analysis quality, and progress toward a stable explanatory graph. While all models adhered to the basic experimental loop, higher-performing models like `Claude Opus` and `Claude Sonnet` executed broader parameter sweeps, deeper falsification cycles, and built more coherent causal narratives. `GPT-4.1` showed strong consistency with controls and factorial tests, while the smaller `o3-mini` and `o4-mini` models often lacked follow-through in iteration and quantitative depth. Table 7 summarizes the average rubric scores across each model's best-performing run.

Claude Sonnet 4 example response

Hypothesis: The simulation exhibits a critical fiscal threshold; government spending above 15% of GDP leads to unstable debt dynamics and boom-bust cycles.
Design Experiment: Reduce government spending to test fiscal stabilization effects.

Model	Workflow	Coverage	Hypothesis	Data Analysis	Goal Progress	Net Score
o3-mini	5.0	2.5	3.0	2.0	2.0	**2.9**
o4-mini	5.0	3.0	4.0	2.5	2.5	**3.4**
GPT-4.1	5.0	3.0	4.0	3.0	3.0	**3.6**
Claude Sonnet	5.0	3.5	4.5	3.5	3.5	**4.0**
Claude Opus	5.0	4.0	4.5	3.5	4.0	**4.2**

Table 7: The detailed evaluation of various models (with and without reasoning) for understanding the causal connection between the properties.

```
{
  "g0": 0.10,
  "number_of_steps": 100
}

Observation & Analysis:
Government debt peaked at  35K versus 260K in the baseline.  GDP decline was smoother,
without a dramatic crash.  Inflation remained volatile, and unemployment was higher
on average.  This suggests high spending triggers fiscal crises, while lower spending
trades off economic capacity for debt sustainability.

Updated Hypothesis:  A fiscal cliff exists between 12-15% spending; crossing it causes
systemic instability.  Next, test if moderate spending (e.g., 0.15) balances capacity
and sustainability.
```

Common Shortcomings. Despite successfully following the hypothesis–experiment–update workflow, these models exhibit several recurring weaknesses. Their exploration of parameter space is often shallow, leaving many levers untouched or varied only in isolation. Analysis tends to be narrative-heavy and lacks quantitative rigor. The models consistently fail to construct or refine explicit causal graphs, even when causal relationships are central to the task. Hypotheses are revised incrementally but seldom elevated to structural theories. These gaps highlight the need for fine-tuning strategies that emphasize quantitative reasoning, causal modeling, systematic exploration, and long-horizon synthesis.