

DEAIMER RESEARCH

From Coding Agents to Scientific Agents: Why Domain Expertise Matters

Building AI that understands the data, assumptions, and tools behind scientific work

Shahzaib Awan
CEO, Deaimer
7 October 2026

Research perspective | Domain expertise and scientific AI

How data semantics, numerical judgment, and specialist environments can help AI agents perform useful scientific work.

Abstract

Scientific work increasingly takes place through software, but scientific competence cannot be reduced to producing executable code. An agent may implement an analysis correctly at the programming level while selecting an unsuitable method, misreading the meaning of an input, or overstating what the result establishes. The transition from coding assistant to scientific agent therefore depends on a richer combination of capabilities: representation reasoning, domain knowledge, tool orchestration, numerical judgment, reproducibility, and disciplined interpretation. This article examines that transition through six primary research sources covering scientific coding, data-driven discovery, computational reproduction, and multi-step scientific tool use. We distinguish the role of executable tasks from the broader ambitions of autonomous discovery, and explain how domain experts can make meaningful scientific decisions visible within an environment. We then develop a practical framework for custom datasets: specify the scientific capability, preserve the semantics of the assets, design relevant variations, and test transfer to separate workflow families. For model developers and dataset buyers, the central opportunity is to turn specialist knowledge into repeatable learning situations. The article presents a research perspective and illustrative environment designs, rather than a new performance experiment or a claim of scientific autonomy.

1. Scientific work begins where execution stops being enough

A program can finish without answering the intended question. It can read a file, fit a model, generate a plot, and write a report while silently carrying an incorrect assumption through every step. Scientific software makes this distinction especially visible because the meaning of the computation depends on the meaning of the data and the conditions under which a method is appropriate.

Consider an agent asked to analyze a set of measurements. At the coding level, the workflow appears straightforward: load the values, clean the table, choose a library, and produce an output. At the scientific level, each action contains a decision. What do the columns represent? Are observations independent? Does the instrument record a transformed quantity? Which values are missing because of the measurement process? Is the selected model consistent with the available evidence?

An agent that ignores those questions may generate a technically polished artifact with weak scientific value. A smooth curve does not establish a good model. A low numerical error on one input does not establish that a method is stable across the intended regime. A plausible summary does not establish that the supplied data support its conclusion.

The research target is therefore richer than generic coding performance. Useful scientific agents must connect observations, assumptions, computations, and

conclusions. They should use software to implement a defensible workflow, while preserving enough evidence for a researcher to understand what was done.

This does not require treating every scientific task as open-ended discovery. Many valuable research activities have concrete objectives: transform an instrument output, reproduce a computation, calibrate a model, run a simulation, or assemble a consistent analytical artifact. These bounded workflows offer a practical starting point for agent development.

2. What the research is actually measuring

The scientific-agent literature covers several related but different targets. Some benchmarks test code synthesis for scientific problems. Others test data-analysis decisions, execution of research workflows, reproduction of published results, or coordination across specialized tools. Those targets should remain distinct when interpreting results.

SciCode draws on scientific expertise across 16 natural-science subfields. Its original benchmark contains 80 main problems decomposed into 338 subproblems involving knowledge, reasoning, and implementation. Its contribution is a scientist-curated view of research coding, rather than a general claim about every form of scientific autonomy. [1]

ScienceAgentBench takes another route, extracting 102 tasks from 44 peer-reviewed publications across four disciplines. It emphasizes individual workflow tasks as a prerequisite to stronger claims about end-to-end automation. The paper's reported results concern the models and configurations tested in that study, not a current universal limit on AI performance. [2]

More recently, the official Terminal-Bench-Science 0.1 release presents 70 expert-curated workflows across scientific disciplines. Its release evaluation reports a highest resolution rate of 30% among the tested systems. That finding indicates substantial headroom on that task collection under those conditions; it does not mean agents can perform only 30% of science. [3]

These sources support a common research direction: scientific capability should be examined through specific activities with meaningful inputs, tools, and outputs. They also show why a single score requires context. Different collections sample different workflows and impose different interfaces, resources, and outcome definitions.

For dataset buyers, the important step is to name the capability they need. Reproducing a published analysis, selecting a numerical method, and integrating a scientific toolchain are all useful objectives, but they demand different environments and different evidence.

3. Three layers of scientific correctness

It is helpful to distinguish three layers of correctness. This is an analytical framework for the article, rather than a new scoring system or a proposed official benchmark metric.

The first layer is operational correctness. Does the program execute in the supplied environment? Can it locate its inputs, use its dependencies, and produce the required artifact? Failure here can prevent any scientific work from taking place.

The second layer is semantic correctness. Does the computation preserve the meaning of the quantities involved? Units, indexing conventions, coordinate systems, identities, and data relationships belong here. A program can execute perfectly while applying an operation to the wrong representation.

The third layer is inferential correctness. Does the result justify the conclusion being drawn? This includes the suitability of a statistical model, the scope of a simulation, uncertainty, and the distinction between an observed relationship and an explanatory claim.

Layer	Central question	Illustrative failure
Operational	Did the workflow run as required?	A dependency error prevents the analysis
Semantic	Did the computation preserve the data's meaning?	A unit mismatch changes the calculated scale
Inferential	Does the evidence support the conclusion?	Correlated measurements are treated as independent samples

The layers interact. A representation mistake can invalidate the assumptions of an otherwise reasonable method. An operational workaround can change the scientific question if it silently drops difficult inputs. A reporting mistake can conceal the distinction between an observed outcome and an unsupported interpretation.

A useful agent environment should expose the layers relevant to its objective. A task about tool integration may emphasize operational behavior. A task about numerical reasoning may make representation and method choice central. A task about analysis may require a result whose interpretation stays within the evidence supplied.

This separation helps developers diagnose failures. It also helps buyers avoid purchasing a collection that appears scientific but exercises only generic file handling and code generation.

4. Data semantics are part of the problem

Scientific data carry meaning that is often distributed across files, metadata, conventions, and documentation. The raw values alone may be insufficient to identify the right computation.

A time series can use different sampling intervals. A coordinate can refer to an image pixel, a physical location, or a transformed frame. A molecular representation can encode properties that are lost through a careless conversion. An observation can represent a subject, a repeated measurement of the same subject, or an aggregate over several subjects.

An agent must discover enough of that context before acting. This makes inspection a scientific skill rather than merely a preparatory shell operation. Reading a schema, checking metadata, and comparing dimensions can determine whether the subsequent method is meaningful.

Representation changes can also alter the structure of an analysis. Converting a spectral axis, for example, can change sample spacing. An algorithm that assumes uniform spacing should not be applied automatically after a nonlinear transformation. The required treatment depends on the quantity represented and the method's assumptions.

Custom environments can make these issues explicit without dictating every action. They can supply metadata in a realistic location, include inputs with different conventions, and define an output that requires reconciliation. The agent then has to connect the evidence to the computation.

The variation should be meaningful. Changing file names alone tests familiarity with paths. Changing the representation while preserving the scientific objective can test whether the agent understands what must remain invariant.

5. Numerical judgment goes beyond calling a library

A scientific library can supply a reliable implementation of a method while leaving the choice of method to the user. That boundary becomes important when an AI agent selects algorithms automatically.

An optimization routine may converge to a local solution that does not answer the intended question. A numerical solver may be unsuitable for a regime with rapidly changing behavior. A matrix computation may amplify small errors when the problem is poorly conditioned. A simulation may run with a resolution too coarse to support the requested conclusion.

The agent's task is to recognize the relevant conditions. It may need to inspect scale, constraints, boundary conditions, or the structure of the input before selecting a tool. A familiar function name is not enough to establish suitability.

A useful learning environment can present an admissible choice among methods rather than requiring one memorized implementation. The objective can specify an accuracy requirement and a resource envelope, while the data reveal which approaches are feasible. This tests the connection between mathematical reasoning and execution.

The output should also make relevant numerical assumptions inspectable. Parameters, convergence information, and the treatment of exceptional inputs can help a researcher understand how a result was obtained. The level of detail should follow the workflow's needs rather than become an arbitrary reporting burden.

For buyers, numerical tasks are most valuable when the challenge is scientifically consequential. Extra computation or obscure syntax can make a task difficult without exercising the method-selection capability the buyer wants to improve.

6. Analysis requires decisions, not just answers

Data-driven research often admits more than one reasonable analytical route. The important question is whether the selected route is justified by the data and the research objective.

BLADE studies this problem through research questions and datasets drawn from the literature. It emphasizes domain knowledge, statistical expertise, and data semantics in analytical decision making, while recognizing that valid analyses can have different representations. Its findings motivate attention to the choices within a workflow, not only to a final response. [4]

A custom environment can make an analytical decision visible. For example, the agent may need to identify the observational unit before summarizing outcomes. If repeated measurements belong to the same entity, a method that assumes independent records can be inappropriate. The environment can provide identifiers and documentation that allow the structure to be discovered.

Missingness is another useful mechanism. Some datasets omit values because of technical failure, others because of a selection process related to the quantity of interest. An agent should not automatically assume those situations are interchangeable. The task can specify enough context to support an admissible treatment without requiring an unsupported causal explanation.

Interpretation belongs in the output contract. The agent may be asked to produce a result and a short account of the assumptions needed to read it. That account should stay tied to the supplied evidence. Fluency should not compensate for a method that answers a different question.

These environments can support learning from successful strategies, comparison between attempts, and diagnosis of recurring mistakes. The best data preserve the relationship between the decision, its evidence, and the resulting artifact.

7. Scientific tool use is a sequential capability

Scientific workflows frequently require a chain of tools. An output from one step becomes the input to another, sometimes after a transformation or interpretation. The challenge lies in maintaining the contract across the sequence.

An agent might prepare an input, invoke a simulator, inspect its diagnostics, transform the generated data, and produce an analysis. Each stage can succeed locally while the overall workflow fails. A representation mismatch between stages may be more important than any individual command error.

SciAgentGym examines multi-step scientific tool use with 1,780 domain-specific tools across four disciplines. The authors report weaker performance as interaction horizons extend and investigate training trajectories that reflect tool dependencies. These results belong to their experimental setting; they provide research context for studying coordinated tool use in custom environments. [5]

The practical implication is that a dataset should represent dependencies, not just a list of tool names. A workflow requiring three unrelated commands elicits different behavior from one where each command depends on interpreting the previous result.

Useful difficulty can arise through these dependencies. An agent may need to revise an initial assumption after a diagnostic, choose a different configuration, or discover that a downstream tool expects a different format. The environment should make the relevant facts accessible and the consequence of each decision meaningful.

For a buyer, the toolchain is part of the capability specification. A task built around an unfamiliar interface can be useful for transfer research, while a task using the deployment toolchain can be useful for targeted development. The intended experiment determines which design is appropriate.

8. Reproducing a result is a meaningful research task

Computational reproduction offers a concrete bridge between coding and scientific practice. The agent receives code and data associated with a study and must recover the relevant result under specified conditions.

CORE-Bench formalizes this target through 270 tasks based on 90 scientific papers across three disciplines. Its focus is computational reproducibility, including the execution and interpretation needed to answer questions about a study's outputs. This is a distinct capability from inventing a new hypothesis or independently establishing a novel discovery. [6]

Reproduction can require substantial reasoning. The agent may need to identify an entry point, understand configuration options, find the relevant output, or determine why a runtime result differs from a documented expectation. Those decisions connect system understanding to the scientific artifact.

However, successful reproduction should be interpreted carefully. Recovering a result shows that a particular computational pathway can be executed and inspected. It does not independently establish that the study's assumptions are correct or that its conclusions generalize beyond the original setting.

A custom reproduction environment can make that scope explicit. It can ask for a defined artifact, documented execution conditions, and a bounded comparison with a reference. The agent's responsibility is then concrete and useful to a researcher.

Dataset rights also matter. Public access to a paper does not imply permission to redistribute every associated asset. Buyers and suppliers should identify the rights covering code, data, and any adapted materials before treating a reproduction task as a licensed product.

9. What domain experts contribute to the dataset

Domain expertise shapes which decisions matter. An expert can identify a workflow where a superficially correct output would still be scientifically misleading, or where a representation assumption changes the method that should be used.

This contribution begins before implementation. A useful task idea should identify the scientific objective, the essential reasoning, and the conditions that distinguish an admissible result from a plausible-looking one. The environment can then be built around that structure.

Experts also help choose realistic assets and constraints. A synthetic input may be useful when it preserves a scientifically meaningful mechanism. An apparently realistic dataset may be less useful if the task ignores the features that make the domain challenging. Authenticity comes from the decisions the environment requires, not solely from the source of its files.

Technical authors translate those decisions into executable packages. They prepare dependencies, inputs, tool interfaces, manifests, and outputs. Their work makes specialist knowledge available as repeatable experience rather than leaving it as an informal description.

The collaboration should preserve alternative valid approaches. A task should not reject an admissible method merely because it differs from the author's preferred implementation. Scientific objectives often allow several sound routes, and flexibility can reveal useful agent capability.

For buyers, the question is whether the team supplying the dataset can explain the scientific mechanism behind each family. Credentials and domain labels provide context, but a clear account of the reasoning target is more directly useful when assessing fit.

10. Designing a scientific learning environment

A scientific environment should begin with a capability target and a task contract. The target describes what the agent should learn or demonstrate. The contract describes the objective, constraints, inputs, and expected outputs for a specific situation.

The initial state should contain enough context to make the work executable. This can include datasets, metadata, configuration, partial code, or a prepared toolchain. Important scientific information should be discoverable rather than omitted as an accidental obstacle.

Dependencies should serve the workflow. A library, solver, or simulator can be central when its behavior determines the decisions the agent must make. Unnecessary installation complexity can increase cost while adding little scientific value.

The expected artifact should connect to the target skill. A numerical-method task might require an output whose accuracy and resource behavior reflect method choice. An analysis task might require a structured result with assumptions and provenance. A toolchain task might require a complete pipeline whose stages remain compatible.

Design component	Scientific purpose	Example
Objective	Define the bounded research activity	Produce a calibrated simulation output
Assets and metadata	Preserve the meaning of inputs	Units, sampling conventions, identifiers
Tools	Expose relevant scientific operations	Solver, analysis library, simulator
Constraints	Make the desired judgment necessary	Accuracy, runtime, admissible assumptions
Artifact	Make the work inspectable	Results, configuration, bounded interpretation
Family variation	Test adaptation across meaningful changes	New regimes, representations, or tool interfaces

A package should also fit the buyer's execution setup. Repeated training requires practical initialization and reset behavior. Evaluation requires controlled conditions and a defined relationship between what the agent sees and what is reserved for measurement.

Quality information can remain concise: identify the exact package version, execution assumptions, readiness status, and known limitations. Detailed internal validation procedures are outside this article's scope. The buyer needs enough evidence to assess usability without mistaking a package description for a completed model experiment.

11. Four illustrative scientific task families

The following scenarios are proposed design examples. They are not disclosures of private tasks, claims about an existing licensed inventory, or reports of completed training experiments.

11.1 Representation-aware signal processing

An agent receives measurements, metadata, and an objective involving a transformed signal. It must reconcile the axis convention and sampling assumptions before applying a method. The learning target is the connection between representation and analysis.

A family can vary units, spacing, and metadata organization while retaining a common scientific objective. A separate family can change the tool interface or the type of signal. That distinction allows a researcher to investigate whether the strategy transfers beyond familiar files.

11.2 Numerical computation under a resource envelope

An agent must solve a defined computational problem within accuracy and runtime constraints. Several methods are available, but their suitability depends on the input regime. The environment makes those properties visible through data and documentation.

The target is method selection and interpretation of numerical behavior. A useful trajectory shows what the agent inspected before choosing an approach and how it responded to evidence of instability or poor convergence.

11.3 A multi-stage simulation workflow

An agent prepares a model input, runs a simulator, interprets diagnostics, and produces an output artifact. The environment includes a representation mismatch between stages that must be resolved through inspection.

The target is coordination across tools and preservation of scientific meaning. Variations can change the mismatch, configuration, or constraints. Adding stages is valuable when the dependencies create a substantive planning challenge.

11.4 Analysis with a structured observational unit

An agent receives measurements in which several rows belong to the same underlying entity. It must identify that relationship and produce an analysis consistent with the defined objective.

The target is data semantics and justified analytical choice. A collection can vary the representation of the grouping structure and the permitted methods. The final artifact should make the chosen treatment and its assumptions clear.

These families illustrate how domain knowledge becomes executable experience. Each situation requires more than recognizing scientific vocabulary: the agent must discover and use a relationship that changes the computation.

12. Training for a capability, then testing transfer

A custom scientific dataset becomes more useful when its learning purpose is explicit. A team may want successful demonstrations, repeated interaction with resettable environments, or comparative examples of different strategies. Those are related but distinct data products.

Supervised learning can use trajectories that show a sound sequence of observations and actions. Reinforcement learning can use environments where the agent's interventions produce task-relevant feedback. Comparative behavior data can distinguish attempts that reach the same result through different costs or assumptions. The buyer chooses the learning method; the dataset supplies the situations and agreed records.

The mixture should match the current agent. Very difficult tasks may reveal limitations but yield little positive experience for a weak policy. Easier tasks can teach foundations, while more demanding ones combine the foundations across a longer workflow. The curriculum is a hypothesis about learning that requires an experiment.

Transfer needs a separate collection. A team can hold out asset families, toolchains, representations, or problem regimes. Each choice tests a different claim. Success on new files with the same mechanism supports a narrower conclusion than success on a new mechanism within the same domain.

Cross-domain transfer should be measured rather than assumed. Scientific fields share some strategies, such as inspecting units or tracing dependencies, while differing sharply in their tools and assumptions. A training gain in one area does not automatically establish competence elsewhere.

The experiment should preserve identifiable model and harness configurations. A useful comparison controls the execution conditions and records the training exposure. It should report the scope of the observed improvement alongside the conditions that remain untested.

13. A practical brief for dataset buyers

The starting point is the work the agent should perform. A buyer should describe the scientific objective, the current bottleneck, the tools it uses, and the resources available. That brief gives the supplier a concrete basis for proposing task families.

A pilot should test representative environments in the buyer's actual setup. Integration, reset behavior, output collection, and experience cost matter because the same

package may be executed many times. Small differences in setup effort can become significant at training scale.

The buyer should also inspect capability coverage. A collection with many tasks can still repeat one reasoning pattern. A clear family map helps show how the proposed data address the intended scientific mechanisms.

Buyer decision	What to specify
Scientific scope	Disciplines, workflows, and capability targets
Runtime	Tools, compute, dependencies, network assumptions
Learning material	Packages, demonstrations, trajectories, and metadata
Evaluation boundary	Held-out families and access conditions
Commercial use	Rights, adaptation, redistribution, and delivery scope
Pilot outcome	Integration evidence and the research hypothesis to test

The decision to expand should follow evidence from the pilot. A useful result may show that a family fits the intended learning goal, that a toolchain needs adaptation, or that the bottleneck was defined too broadly. Procurement becomes stronger when it improves the experiment as well as supplying data.

14. Deaimer's role in specialist agent development

Deaimer's Termetheus offering provides a framework for custom terminal-style task datasets and environments. For scientific-agent teams, the relevant opportunity is to commission executable situations around defined domain capabilities, tools, and operating constraints.

The exact scope should be agreed for each collection. Scientific disciplines differ in their asset requirements, dependencies, compute needs, and rights. A supported task format should not be interpreted as evidence that every possible domain collection is already available.

The value of customization is alignment. A team focused on numerical reasoning can prioritize method-selection problems. A team building research assistants can prioritize tool coordination and reproducibility. A team improving analytical agents can prioritize data semantics and complete artifacts.

Deaimer's role is to help turn that capability brief into a scoped dataset proposal. The buyer retains responsibility for its learning method, experimental design, and interpretation of results. Meaningful model-improvement claims require evidence from those experiments.

Teams seeking custom environments can begin with a short description of their target workflow and agent limitations at <https://www.deaimer.com/>. That conversation can lead to a representative sample and a pilot designed around the intended scientific capability.

15. Conclusion

Scientific agents need to connect code with the world the code represents. That connection depends on data semantics, domain assumptions, numerical judgment, tool coordination, and interpretation. Successful execution provides important evidence, but its scientific meaning comes from the workflow around it.

The research literature offers several concrete ways to examine these capabilities: scientist-curated coding problems, data-analysis tasks, computational reproduction, and multi-step tool environments. Together they support a practical agenda focused on bounded research work and inspectable artifacts.

For developers, the next step is to identify the scientific decisions their agents need to master. For dataset buyers, it is to commission environments that make those decisions necessary and measurable, then test whether experience transfers to separate workflows.

Domain expertise becomes especially valuable when it is translated into executable situations. The result is a supply of learning experience shaped by the scientific work itself, giving AI teams a clearer path from general coding competence toward useful specialist research assistance.

References

- [1] Tian, M., Gao, L., Zhang, S. D., et al. SciCode: A Research Coding Benchmark Curated by Scientists. 2024. arXiv:2407.13168. <https://arxiv.org/abs/2407.13168>
- [2] Chen, Z., Chen, S., Ning, Y., et al. ScienceAgentBench: Toward Rigorous Assessment of Language Agents for Data-Driven Scientific Discovery. ICLR 2025. arXiv:2410.05080, version 3. <https://arxiv.org/abs/2410.05080>
- [3] Terminal-Bench-Science Team. Terminal-Bench-Science 0.1: Evaluating AI agents on scientific research workflows. 27 August 2026. <https://www.tbench.ai/news/terminal-bench-science-0-1>
- [4] Gu, K., Shang, R., Jiang, R., et al. BLADE: Benchmarking Language Model Agents for Data-Driven Science. EMNLP 2024; arXiv version 3, 2025. <https://arxiv.org/abs/2408.09667>
- [5] Shen, Y., Yang, Y., Xi, Z., et al. SciAgentGym: Benchmarking Multi-Step Scientific Tool-use in LLM Agents. 2026. arXiv:2602.12984, version 2. <https://arxiv.org/abs/2602.12984>
- [6] Siegel, Z. S., Kapoor, S., Nadgir, N., Stroebl, B., and Narayanan, A. CORE-Bench: Fostering the Credibility of Published Research Through a Computational Reproducibility Agent Benchmark. arXiv:2409.11363, version 2, 2026. <https://arxiv.org/abs/2409.11363>

Research note: Sources were checked on 7 October 2026. Findings are attributed to the tasks and experimental conditions described by their authors. The analytical framework and task scenarios in this article are design proposals, not new empirical results.

Deaimer product references describe potential custom scope and do not imply affiliation with the cited benchmark projects.