

SKILLFOUNDRY: Building Self-Evolving Agent Skill Libraries from Heterogeneous Scientific Resources

Shuaike Shen^{1,*}, Wenduo Cheng^{1,*}, Mingqian Ma², Alistair Turcan¹,
Martin JinYE Zhang¹, Jian Ma^{1†}

¹Ray and Stephanie Lane Computational Biology Department,
School of Computer Science, Carnegie Mellon University

²Machine Learning Department,
School of Computer Science, Carnegie Mellon University

*Equal contribution, †Correspondence: jianma@cs.cmu.edu

Abstract

Modern scientific ecosystems are rich in procedural knowledge across repositories, APIs, scripts, notebooks, documentation, databases, and papers, yet much of this knowledge remains fragmented across heterogeneous artifacts that agents cannot readily operationalize. This gap between abundant scientific know-how and usable agent capabilities is a key bottleneck for building effective scientific agents. We present SKILLFOUNDRY, a self-evolving framework that converts such resources into validated agent skills, reusable packages that encode task scope, inputs and outputs, execution steps, environment assumptions, provenance, and tests. SKILLFOUNDRY organizes a target domain as a domain knowledge tree, mines resources from high-value branches, extracts operational contracts, compiles them into executable skill packages, and then iteratively expands, repairs, merges, or prunes the resulting library through a closed-loop validation process. The current library snapshot contains 286 skills across 27 domains and 254 subdomains mined from 394 resources. Under the current search based comparison against SkillHub and SkillsMP, 71.1% of the skills have no retrieved match that completes the same task. The library improves the same coding agent on five of the six MoSciBench datasets. On broader evaluations, SKILLFOUNDRY raises SciCode sampled test accuracy from 65.43% to 71.60% and ScienceAgentBench success rate from 30.95% to 50.00%, outperforming EvoSkill on the shared held out test set without task specific training. We further show that SKILLFOUNDRY can design task specific skills that substantially improve cell type annotation and the scDRS workflow. Together, these results show that automatically mined skills improve agent performance on benchmarks and domain specific tasks, expand coverage beyond hand crafted skill libraries, and provide a practical foundation for more capable scientific agents. Code and an interactive website dashboard for exploring the skill library, domain knowledge tree, and mined resources are available at <https://github.com/ma-compbio-lab/SkillFoundry> and <https://ma-compbio-lab.github.io/SkillFoundry/>.

1 Introduction

LLM agents have demonstrated strong capabilities in reasoning, planning, tool use, and interacting with external environments across a wide range of applications (Wang et al., 2024). However, they often lack the contextual knowledge and domain expertise needed to perform real-world tasks reliably (Xu & Yan, 2026), especially in scientific settings where reliable execution depends not only on access to tools, but also on procedural know-how: specialized software conventions, environment-specific constraints, implicit assumptions, and complex multi-step workflows that are rarely available in a form agents can directly use.

Agent skills provide a natural abstraction for bridging this gap. A skill packages reusable procedural knowledge into a structured, executable unit that specifies what a capability does, how it should be invoked, what inputs and outputs it expects, and what dependencies or constraints it requires. In modern agent frameworks, skills are represented as self-contained packages that agents can discover, load, and invoke when relevant (Anthropic, 2026).

Unlike raw tools, which mainly expose callable interfaces, skills capture richer procedural guidance for coordinating multi-step decisions and actions, making them a more reliable foundation for workflow execution (Xu & Yan, 2026). However, most agent skills today are still hand-crafted, narrowly scoped, and evaluated only in limited settings (Alzubi et al., 2026), making them difficult to scale across diverse scientific domains. Recent work has demonstrated the potential of LLM agents to accelerate scientific discovery and automate complex workflows (Huang et al., 2025; Wang et al., 2025c; Swanson et al., 2025; Qu et al., 2025; Wang et al., 2025a). Yet most existing systems are still built on custom wrappers and predefined tools, making them difficult to scale, adapt, and extend across domains (Gao et al., 2025). At the same time, modern scientific ecosystems already contain a wealth of procedural knowledge in repositories, software packages, APIs, databases, scripts, notebooks, documentation, papers, and web services. The challenge is therefore not a lack of scientific know-how, but that much of this knowledge is buried in heterogeneous, fragmented, and unstructured resources that agents cannot directly retrieve, execute, validate, or compose into reliable workflows.

In this work, we ask whether scientific procedural knowledge hidden in heterogeneous domain resources can be automatically converted into agent skills that support reliable and scalable scientific agents. The central motivation is that agents may already have access to relevant tools and data yet still lack the procedural knowledge needed to combine them reliably in multi step workflows. To this end, we propose SKILLFOUNDRY, a self evolving systems framework that transforms heterogeneous scientific resources into structured, executable skills. Its contribution is an integrated process for mining resources, extracting operational contracts, validating executability and utility, and iteratively refining the resulting library rather than a new standalone learning algorithm. Beyond building a reusable library, SKILLFOUNDRY can also synthesize and test task specific skills on demand for a concrete scientific objective.

We evaluate SKILLFOUNDRY from two complementary perspectives, the quality of the skill library it builds and the downstream utility of those skills on scientific tasks. At the library level, we study whether the mined skills are internally valid, executable, and distinct from skills retrieved from existing libraries under the current comparison protocol. At the task level, we evaluate the existing library on MoSciBench, ScienceAgentBench (Chen et al., 2025), and SciCode (Tian et al., 2024), including a comparison with EvoSkill (Alzubi et al., 2026), and study newly constructed skills on cell type annotation and scDRS (Zhang et al., 2022). Across these settings, SKILLFOUNDRY improves five of six MoSciBench datasets, raises SciCode sampled test accuracy from 65.43% to 71.60%, raises ScienceAgentBench success rate from 30.95% to 50.00%, outperforms EvoSkill on the shared held out test set, and substantially improves both genomics workflows.

This paper makes the following contributions:

- We introduce SKILLFOUNDRY, a systems framework that converts heterogeneous domain resources into a self evolving library of structured, executable, and validated agent skills.
- We establish an evaluation protocol that combines library-level validation, including executability and novelty, with task-level utility on benchmarks and realistic scientific workflows.
- We demonstrate that the skills produced by SKILLFOUNDRY show substantial nonoverlap under the current search protocol and improve agent performance across MoSciBench, ScienceAgentBench, and SciCode, including against EvoSkill.
- We show that SKILLFOUNDRY can also design task specific skills for real world scientific workflows.

2 Related Work

Agent Tools and MCPs. Prior work has improved LLM agents’ ability to access external capabilities through tool use learning, API calling, and benchmarked tool augmented agent systems (Schick et al., 2023; Patil et al., 2024; Li et al., 2023; Qin et al., 2023). However, these lines of work mostly assume that the needed capabilities are already available in usable form. MCP provides a standard way to expose tools, resources, and prompts to agents (Anthropic, 2024). Tools and MCP services primarily expose executable interfaces and external capabilities. A workflow is an instantiated sequence for a particular task. A skill is a reusable procedural package that can support multiple workflows and records task scope, inputs, outputs, dependencies, provenance, execution guidance, and tests. SKILLFOUNDRY therefore operates at a layer above tools and MCP services. Its stable contribution is the process that converts heterogeneous resources into validated procedural capabilities and maintains them through mining, testing, repair, novelty checking, and refinement. This process remains useful even if the community later adopts different terminology for the resulting abstraction.

Agent Skills. Prior work has shown that agents benefit from reusable skill libraries. Voyager (Wang et al., 2023) introduced an ever-growing library of executable skills for lifelong embodied learning, while Anthropic’s Claude framework formalized agent skills as modular packages that combine instructions, metadata, and optional scripts or resources (Anthropic, 2026). More recent work has explored different ways of building and improving such skills, including reinforcement-learning-based skill reuse (Wang et al., 2025b), large engineered skill bases for computer-use agents (Chen et al., 2026), composable reasoning modules (Jiao et al., 2026), and iterative skill discovery through failure analysis (Alzubi et al., 2026). Together, these studies highlight the value of reusable and evolving skills. Our setting differs in that we focus on mining skills from heterogeneous domain resources, rather than constructing them primarily from trajectories, failures, reinforcement learning, or manual engineering.

Tool Ecosystems and Automatic Tool Discovery. A closely related line of work focuses on building large tool ecosystems or automatically converting external artifacts into agent accessible tools. ToolUniverse (Gao et al., 2025) presents a large scientific ecosystem for AI scientists and standardizes access to more than 1,000 machine learning models, datasets, APIs, and scientific packages. Deploy Master (Wang et al., 2026) addresses large scale tool discovery by identifying more than 50,000 runnable tools from public repositories. Recent work also automates construction from specific artifacts. Paper2Agent (Miao et al., 2025) converts papers and associated codebases into interactive MCP based agents. Code2MCP (Ouyang et al., 2025) transforms code repositories into MCP services, while ToolRosetta (Di et al., 2026) translates repositories and APIs into MCP compatible tools with reliability and security checks. The MCP native hierarchical AI scientist perspective studies how such services can be organized into interoperable scientific ecosystems (Yue et al., 2026). These efforts expand the supply and organization of agent accessible tools and services. SKILLFOUNDRY is complementary. It can consume MCP generated tools as resources and contributes reusable procedural skills that specify how those capabilities should be applied and validated across workflows.

Scientific Agents and Tool Use. Recent work has explored both domain-specific and more general scientific agent systems. Domain-specific agents such as ChemCrow (Bran et al., 2023), SpatialAgent (Wang et al., 2025a), and CRISPR-GPT (Qu et al., 2025) demonstrate the value of integrating specialized tools and workflows tailored to chemistry, spatial biology, and CRISPR-based gene editing, respectively. More general systems such as Biomni (Huang et al., 2025) aim to support a broader range of biomedical tasks through larger tool ecosystems. These works collectively highlight both the promise and the challenges of scientific tool use, particularly the need for strong domain-specific understanding, since even small errors in tool selection, parameterization, or execution can lead to invalid results or misleading conclusions (Wei et al., 2025). Our work is complementary to this line of research: rather than focusing on a particular scientific domain or agent system, we study how reusable procedural skills can be mined, validated, and refined from heterogeneous domain resources to support more capable scientific agents.

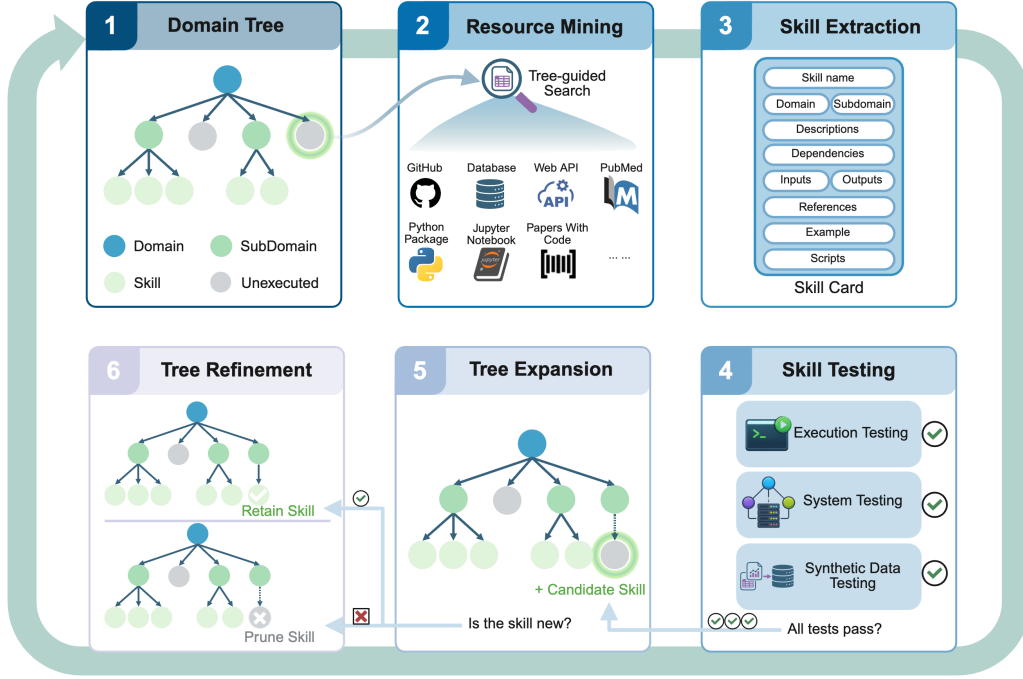


Figure 1: Overview of SKILLFOUNDRY. Starting from a domain knowledge tree, the framework mines branch-relevant resources, extracts structured candidate skills, validates them through multi-level testing, expands the tree with verified skills, and prunes redundant or low-value leaves.

3 Method

3.1 Method Overview

SKILLFOUNDRY is a tree guided systems framework for automatically building and maintaining a domain skill library from heterogeneous scientific resources. The domain knowledge tree serves both as a structured search prior and as the object being updated. Under covered branches trigger targeted resource mining, mined artifacts are compiled into executable skills, and evaluation signals determine whether the tree should be expanded, revised, or pruned. Searching bounded branches instead of the full resource space turns open ended skill collection into a scalable closed loop acquisition process.

Figure 1 illustrates this loop. Starting from a domain tree (Step 1), SKILLFOUNDRY selects a branch and mines relevant resources such as repositories, databases, APIs, papers, and notebooks (Step 2). The retrieved artifacts are compiled into structured skill cards with explicit scope, dependencies, inputs, outputs, sources, and examples (Step 3). Each skill then undergoes execution testing, system testing, and synthetic-data testing (Step 4). Skills that pass are added as new candidate leaves (Step 5), while redundant or low-value skills are consolidated or removed during tree refinement (Step 6). The updated tree then guides the next round of mining, yielding a self-evolving skill library.

3.2 Domain Knowledge Tree Construction

We represent the target field as a rooted tree $T = (V, E)$, where internal nodes denote domains and subdomains, and leaves denote actionable skill targets. Rather than serving as a static taxonomy, this tree acts as the control structure for adaptive mining. Each node stores lightweight state, including linked resources, existing skills, and validation status, so the system can distinguish well-covered regions from high-value frontiers. As a result, SKILLFOUNDRY prioritizes branches with abundant resources but weak verified skill coverage, rather than searching the resource space uniformly.

The tree is initialized only with a manually curated high level seed taxonomy. Concrete subdomains and skill leaves are not manually enumerated. They are expanded, split, merged, or pruned by SKILLFOUNDRY as resource mining and validation reveal stable structure. The tree thus serves as a structured prior over the domain and as a compact summary of current library coverage.

3.3 Tree-Guided Skill Mining

Given the current tree, SKILLFOUNDRY runs a staged mining loop that selects a focus branch, retrieves relevant resources, and compiles candidate skills. In the implementation, this loop is instantiated as `tree_check` $\rightarrow$ `resource_search` $\rightarrow$ `skill_build` $\rightarrow$ `skill_test` $\rightarrow$ `refresh`. Resource search prioritizes authoritative artifacts, such as official documentation, maintained repositories, package references, workflows, notebooks, and method papers, so that skill induction is grounded in reliable sources rather than generic web text.

From the retrieved artifacts, the system extracts an operational contract and compiles it into a reusable skill package. As reflected by the skill card in Fig. 1, each skill records its scope, dependencies, inputs, outputs, provenance, and example usage. In the repository, a compiled skill includes both human-readable instructions and machine-readable metadata, together with executable scripts or tests when available. The refresh stage then updates the registry so that subsequent iterations operate on an accurate view of coverage and gaps in the library.

3.4 Illustrative scDRS Skill Construction

An scDRS skill illustrates the full conversion process. SKILLFOUNDRY first consolidates the scDRS paper, documentation, command line references, file format requirements, MAGMA and GWAS resources, and relevant technical discussions. It then extracts an operational contract with environment checks, input schemas, command templates, expected outputs, intermediate checkpoints, failure handling rules, provenance, and validation gates. The skill also records domain assumptions that a general purpose agent can easily miss. These include HLA and MHC exclusion, SNP to gene mapping, gene set construction, consistency between AnnData objects and covariates, positive control checks, and FDR based interpretation. The resulting package is tested through the same execution and synthetic data interfaces used for mined library skills before it is exported to the downstream agent. This example shows how distributed scientific resources become one reusable and executable procedural capability rather than a collection of links or an API wrapper.

3.5 Skill Testing and Tree Refinement

A candidate skill is added to the library only after multi-stage validation. As shown in Step 4 of Fig. 1, SKILLFOUNDRY applies three complementary tests. Execution testing checks whether the skill runs under its declared contract. System testing validates skills whose realistic use depends on infrastructure such as SLURM. Synthetic-data testing is used when live resources are unavailable, expensive, unstable, or difficult to exercise repeatedly during development. Rather than approximating the full downstream task, it checks whether the skill correctly consumes controlled inputs, supports all required arguments and file interfaces specified by the skill contract, and produces stable outputs under fixed mock conditions. In this sense, it verifies both contract completeness and behavioral stability. Here, contract completeness refers to whether the skill specifies all inputs, outputs, and execution assumptions needed for reliable use, while behavioral stability refers to whether it produces consistent outputs under the same controlled inputs rather than varying with hidden state or incidental changes in the environment. Together, these tests establish a minimum standard of executability and robustness before a skill is accepted into the library.

Evaluation results directly update the tree. Skills that pass are inserted as candidate new leaves and expand coverage in the corresponding branch, as shown in Step 5. Skills that fail repeatedly, duplicate existing functionality, or contribute little marginal value are revised, merged, or pruned during refinement, as shown in Step 6. This final step closes the loop, since SKILLFOUNDRY does not simply accumulate skills, but continuously reorganizes the tree according to empirical signals of correctness, usefulness, and novelty.

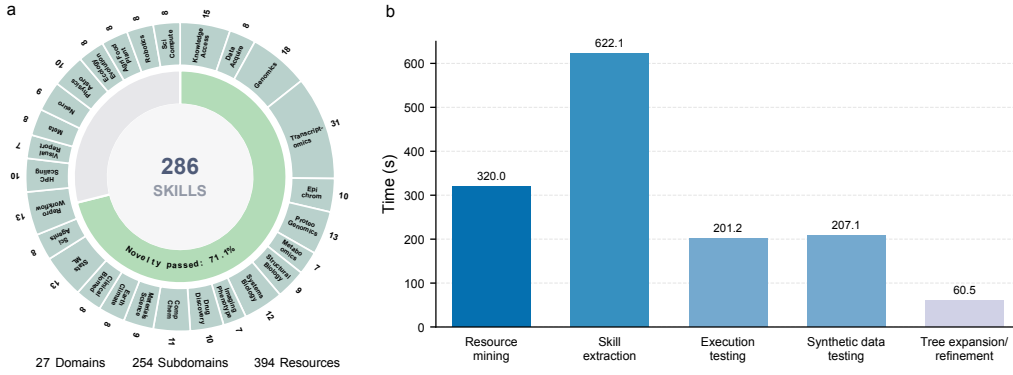


Figure 2: Composition of the mined skill library and runtime of the mining pipeline. (a) Distribution of the 286 mined skills across domains and subdomains. (b) Average runtime of the major pipeline stages under a fixed resource-mining budget.

4 Experiments

Having introduced how SKILLFOUNDRY builds a skill library from heterogeneous scientific resources, we next evaluate it in four complementary settings. First, we characterize the mined library through statistical analysis and internal validation. Second, we assess whether the existing library improves the same coding agent on MoSciBench (Liu et al., 2026). Third, we broaden downstream evaluation to ScienceAgentBench (Chen et al., 2025) and SciCode (Tian et al., 2024), including a comparison with EvoSkill (Alzubi et al., 2026). Fourth, we examine whether SKILLFOUNDRY can construct new skills for cell type annotation and the scDRS workflow.

4.1 Skill Library Overview

We first summarize the mined library before turning to downstream evaluation. Because SKILLFOUNDRY is continuously expanding and self evolving, the following statistics provide a snapshot at the time of writing. At this point, SKILLFOUNDRY contains 286 skills spanning 27 domains and 254 subdomains, mined from 394 resources (Fig. 2a). Every candidate skill is subjected to execution testing and synthetic data testing, and skills whose intended use depends on cluster infrastructure are additionally evaluated with system testing. When a candidate skill fails these checks, the framework enters a repair and optimization loop and retests the skill until it either passes or is removed from consideration. As a result, all retained skills in the current library pass the required validation tests. To estimate novelty, SKILLFOUNDRY derives search terms from each candidate description, queries SkillHub (SkillHub, 2026) and SkillsMP (SkillsMP, 2026), and judges whether any retrieved skill completes the same task with the same method scope. Under this protocol, 71.1% of the mined skills have no matching skill and are added as new leaves in the domain knowledge tree. The remaining 28.9% have related matches. When a retrieved skill has the same task description and method scope, the mined skill is discarded as redundant. When the retrieved and mined skills are complementary, the framework merges them and keeps the merged result as a leaf during tree refinement. Keyword search may miss functionally similar skills with different names or descriptions. The 71.1% value is therefore a protocol dependent estimate of nonoverlap rather than an absolute uniqueness claim.

We next evaluate the runtime of skill discovery. Fig. 2b reports the average running time of the major stages under a fixed resource-mining budget. Skill extraction is the dominant cost, followed by resource mining, whereas testing and tree updates contribute relatively little overhead. We cap the search budget during resource mining to avoid unbounded exploration. The reported cost therefore reflects runtime rather than API token usage, and it measures controlled retrieval under a fixed search budget rather than open-ended search.

Taken together, these results indicate that SKILLFOUNDRY can build a large skill library with substantial nonoverlap under the current search protocol and moderate time cost. The

Setting	health spa			massspecgym			pop genetics		
	Repo-Acc (%)	Paper-Acc (%)	Exec (%)	Repo-Acc (%)	Paper-Acc (%)	Exec (%)	Repo-Acc (%)	Paper-Acc (%)	Exec (%)
Codex	64.71	47.06	100.00	40.00	40.00	100.00	53.85	30.77	100.00
Codex + SkillFoundry	76.47	58.82	100.00	46.67	46.67	100.00	61.54	46.15	100.00

Setting	nurse stress			cyclone			terra		
	Repo-Acc (%)	Paper-Acc (%)	Exec (%)	Repo-Acc (%)	Paper-Acc (%)	Exec (%)	Repo-Acc (%)	Paper-Acc (%)	Exec (%)
Codex	80.00	66.67	100.00	64.29	42.86	100.00	64.29	35.71	100.00
Codex + SkillFoundry	80.00	66.67	100.00	71.43	57.14	100.00	64.29	42.86	100.00

Table 1: Results on all six datasets in MoSciBench for the same coding agent with and without SKILLFOUNDRY skills. Repo-Acc uses the scoring rule implemented in the official MoSciBench repository, whereas Paper-Acc uses the scoring rule described in the MoSciBench paper. Exec denotes code execution success rate. SKILLFOUNDRY improves performance on five of the six datasets and leaves one unchanged, while maintaining perfect execution success throughout.

snapshot of 286 skills from 394 resources provides initial evidence that the tree guided loop scales beyond a few manually constructed examples, although larger scaling studies remain future work. However, internal validation serves only as a screening step, not as final evidence of downstream usefulness. We therefore turn next to benchmark and workflow evaluations to assess whether the mined skills translate into practical gains on scientific tasks.

4.2 Benchmarking Results

We evaluate whether the skills mined by SKILLFOUNDRY improve agent performance on open-ended scientific tasks across three benchmarks. We first consider MoSciBench (Liu et al., 2026) and SciCode (Tian et al., 2024), where, within each benchmark, we directly compare a fixed coding agent with and without SKILLFOUNDRY skills. Detailed benchmark implementation settings, including the Codex model and reasoning configuration used for each benchmark, are provided in Appendix A.1.6. MoSciBench evaluates end-to-end scientific discovery across six datasets spanning climate science, biomedical engineering, cheminformatics, health psychology, population genomics, and earth science. Its tasks require agents to integrate scientific data, perform intermediate analyses, and carry out multi-step reasoning rather than solve isolated prediction problems. SciCode provides a complementary evaluation of scientific code generation across multiple scientific domains. Together, these benchmarks test whether success depends not only on access to tools, but also on reusable procedural knowledge for coordinating evidence across modalities, conducting intermediate analyses, and sustaining multi-step scientific reasoning. We therefore compare the performance of the same coding agent with and without SKILLFOUNDRY skills across the full MoSciBench benchmark.

Table 1 reports the MoSciBench results. Adding SKILLFOUNDRY skills improves performance on five of the six datasets and leaves one unchanged. Averaged over the six datasets, Repo-Acc increases from 61.19% to 66.73% and Paper-Acc increases from 43.85% to 53.05%. The improvement is most evident on *health spa*, *pop genetics*, and *cyclone*, and the skill-augmented agent also improves on *massspecgym* and *terra*. Only *nurse stress* remains unchanged, likely because the baseline is already strong on this dataset. The code execution success rate stays at 100.00% on all six datasets in both settings, which indicates that the gains come from better task completion and scientific reasoning rather than from easier execution alone.

The same trend holds on SciCode. As shown in Table 2, adding SKILLFOUNDRY skills increases test accuracy from 65.43% to 71.60%. Thus, the benefits of the mined skills are not limited to the multimodal discovery workflows in MoSciBench, but also transfer to scientific coding tasks across domains.

We next ask a different question: how does SKILLFOUNDRY compare with EvoSkill (Alzubi et al., 2026), a method that automatically discovers and refines reusable agent skills through

Method	Test accuracy (%)
Codex	65.43
Codex + SKILLFOUNDRY	71.60

Table 2: Results on SciCode, a scientist-curated benchmark for scientific code generation across multiple scientific domains. Test accuracy measures the fraction of evaluated tasks solved correctly. Bold indicates the best result.

Method	Valid output (%)	Success (%)	Mean CodeBERT
Codex	78.57	30.95	0.8348
Codex + EvoSkill	83.33	38.10	0.8475
Codex + SKILLFOUNDRY	92.86	50.00	0.8683

Table 3: Results on ScienceAgentBench, which evaluates scientific workflow coding on tasks derived from peer-reviewed studies. Valid output denotes the fraction of tasks for which the generated program executes successfully and produces the required output, Success denotes the fraction satisfying the task-specific success criteria, and Mean CodeBERT measures similarity between the generated and reference code. EvoSkill uses a task feedback training split, while SKILLFOUNDRY uses no task-specific training. Bold indicates the best result.

iterative failure analysis, using execution feedback to propose skill updates and held-out validation performance to select improvements? For this evaluation, we use ScienceAgentBench (Chen et al., 2025), which measures scientific workflow coding on tasks derived from peer-reviewed studies. In addition to the Codex baseline and Codex augmented with SKILLFOUNDRY skills, we compare against EvoSkill. Because EvoSkill requires task-level feedback for skill acquisition, we randomly allocate 20 of the selected tasks to its training set and evaluate all methods on the same held-out test set. In contrast, SKILLFOUNDRY uses no ScienceAgentBench-specific feedback or training.

Table 3 shows that both skill-augmented methods improve over the Codex baseline, but SKILLFOUNDRY achieves the strongest performance across all three metrics. It increases the valid-output rate from 78.57% to 92.86%, the success rate from 30.95% to 50.00%, and the mean CodeBERT score from 0.8348 to 0.8683. EvoSkill improves the success rate to 38.10% and the mean CodeBERT score to 0.8475, but remains below SKILLFOUNDRY despite using task-specific feedback.

Taken together, these benchmarking results show that augmenting the same underlying agent with SKILLFOUNDRY skills consistently improves performance across both end-to-end scientific discovery and scientific code generation. On ScienceAgentBench, we further show that skills mined from external scientific resources outperform a feedback-based skill-learning baseline, despite requiring no task-specific training. These results suggest that SKILLFOUNDRY captures reusable procedural knowledge that generalizes across scientific domains and task formats.

4.3 Cell Type Annotation

After the benchmark evaluations, we conduct an in depth analysis on two challenging genomics tasks, cell type annotation and scDRS. We show that SKILLFOUNDRY can construct new skills for concrete user specified scientific workflows and improve the performance of a coding agent on these domain specific tasks.

The first task is cell type annotation for spatial transcriptomics data. The goal is to assign each cell to a biologically meaningful identity using molecular profiles and spatial context, which is foundational for many downstream analyses (Shen et al., 2022). This task requires multi-step procedural execution including preprocessing, representation learning, marker or reference reasoning, and result validation, making it a suitable setting to test whether agents can reliably execute workflows rather than just call a single tool.

We use a developing human heart MERFISH dataset (Farah et al., 2024) containing 228,635 cells, 238 genes, and 3 samples. For this task, rather than selecting from a fixed skill library, we provide SKILLFOUNDRY with a task prompt and allow it to retrieve relevant resources, construct candidate skills, and test them for the task. We then evaluate Codex with and without the resulting skill, as well as SpatialAgent (Wang et al., 2025a), a domain specific agent for spatial transcriptomics analysis. The two Codex settings use the same model, dataset, task prompt, evaluation protocol, and search and tool permissions. The baseline is allowed to reason about the analysis strategy, search for supporting information, and construct its own plan. It is not instructed to use PCA or k means. We harmonize both ground truth labels and method outputs into eight major cell types following the SpatialAgent paper (Wang et al., 2025a).

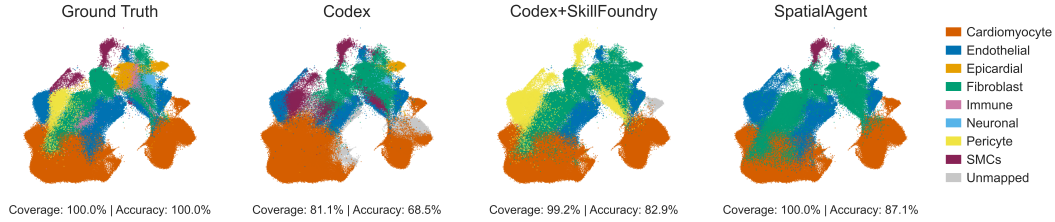


Figure 3: UMAP of cell-type annotations from ground truth, Codex (without skill), Codex+SKILLFOUNDRY, and SpatialAgent. SMCs denote smooth muscle cells. Labels are harmonized into eight major cell types, following the SpatialAgent (Wang et al., 2025a) paper. Coverage is the fraction of cells assigned to one of the eight major cell types, and accuracy is the fraction whose predicted label matches the harmonized ground truth.

Under this shared protocol, different methods adopt distinct workflows. Vanilla Codex selects PCA followed by k means clustering, with marker set scoring and ambiguity handling for cell type assignment. Codex with the SKILLFOUNDRY-generated skill instead applies a marker-signature-based pipeline that integrates PCA, neighborhood graph construction with Leiden clustering, centroid-similarity-based provisional labeling, cluster-level rule refinement, and local neighborhood validation. SpatialAgent uses a domain-specific workflow with expert-curated tools: it first retrieves an external single-cell RNA-seq reference dataset from the CZI database, transfers cell type labels using Harmony, and then annotates clusters based on the transferred labels and marker genes.

Quantitatively, Codex+SKILLFOUNDRY markedly outperforms vanilla Codex in both coverage and accuracy, reaching 99.2% coverage and 82.9% accuracy, compared with 81.1% and 68.5%, respectively (Figure 3). In this setting, higher coverage indicates that the agent is able to assign labels to a larger proportion of cells rather than leaving them unresolved, whereas higher accuracy indicates that those assigned labels better match the ground truth. SpatialAgent performs best overall, achieving 100.0% coverage and 87.1% accuracy. This stronger performance is expected, as SpatialAgent leverages an external single-cell reference dataset, whereas Codex does not rely on external reference data or services. Overall, these results show that SKILLFOUNDRY can successfully construct new skills for user-specified scientific workflows, and that adding the resulting skill substantially improves Codex on this challenging cell type annotation task.

4.4 scDRS Workflow

Besides the cell type annotation task, we evaluate task-specific skill design on the scDRS (Zhang et al., 2022) workflow. scDRS links GWAS summary statistics with single-cell RNA-seq data to identify disease-relevant cells or cell types, and therefore requires the agent to coordinate heterogeneous inputs, manage dependencies and formats, follow a multi-step statistical procedure, and produce interpretable outputs. Compared with benchmark-style tasks, it is a stronger test of workflow competence. In this setting, we ask whether SKILLFOUNDRY can design a useful skill for the scDRS workflow from a user request, and whether the resulting skill can improve the performance of an external scientific agent such as Biomni (Huang et al., 2025). Biomni is a general-purpose biomedical agent with a large tool and database ecosystem, but scDRS is not part of its existing knowledge base or predefined workflows. We therefore compare Biomni with and without the SKILLFOUNDRY skill, packaging the generated skill directly as an external resource to keep the comparison minimally invasive.

We evaluate transfer to an external scientific agent by comparing Biomni with and without these skills on the scDRS workflow using the TMS FACS (The Tabula Muris Consortium, 2020) scRNA-seq dataset together with height GWAS. We run each setting for three replicates and compare the outputs against expert reference analyses in a blinded evaluation by statistical genetics experts. The evaluation includes both qualitative and quantitative criteria. The qualitative score measures whether a run returns the key expected analyses and artifacts, including individual cell, cell-type, and gene-level results, proper FDR correction, identification of chondrocytes as the top associated cell type (Baron et al., 2015), within-cell-

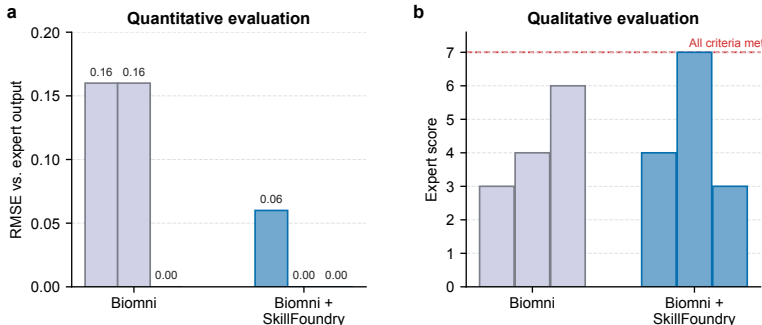


Figure 4: Evaluation of Biomni on the scDRS workflow with and without SKILL-FOUNDRY skills. Panel (a) reports RMSE against expert outputs across three replicates. Panel (b) reports expert scores, where 7 indicates that all criteria are met. SKILLFOUNDRY produces two exact numerical matches and the only run that meets every criterion.

type heterogeneity analysis, and all relevant output files. The quantitative score is the RMSE between the cell-level disease scores produced by each run and the expert reference results.

Results are shown in Fig. 4. One Biomni+SKILLFOUNDRY run is the only setting that satisfies all evaluation criteria. Two of the three skill augmented runs exactly match the expert scores, and the mean RMSE decreases from 0.11 to 0.02. Biomni without skills often omits the filter-data parameter, which causes scDRS to score unfiltered noisy genes and leads to incomplete or inaccurate outputs. The appendix compares representative figures from both settings (Fig. 5, Fig. 6, and Section A.2.2). Without the synthesized skills, Biomni produces only a ranking based on nominal significance. With SKILLFOUNDRY skills, it also reports FDR based support and within cell type heterogeneity. These results show that SKILLFOUNDRY improves workflow accuracy, completeness, and interpretability.

Overall, the experiments show that SKILLFOUNDRY can build a valid scientific skill library with substantial nonoverlap under the current search protocol and can design task specific skills for concrete objectives. Gains on MoSciBench, ScienceAgentBench, SciCode, cell type annotation, and scDRS demonstrate utility beyond the internal validation loop.

4.5 Limitations and Future Work

We acknowledge several limitations. First, the current snapshot covers only a subset of scientific domains and task families. Second, although downstream evidence spans MoSciBench, ScienceAgentBench, SciCode, and two genomics workflows, most individual skills are screened primarily through internal tests. These tests establish executability and contract compliance but cannot capture every practical failure mode. Third, search based novelty assessment may miss functionally similar skills described with different terminology, so the reported 71.1% is protocol dependent evidence of nonoverlap. Fourth, the downstream evaluation remains small relative to the intended scientific scope. Future work should broaden coverage, conduct larger scaling studies, strengthen expert informed evaluation, and improve long term maintenance and novelty assessment.

5 Conclusion

Modern scientific ecosystems contain abundant procedural knowledge, but much of it remains fragmented across resources that agents cannot directly use. SKILLFOUNDRY addresses this gap by converting heterogeneous resources into structured agent skills through a tree guided closed loop for mining, validation, and refinement. Empirically, the resulting skill library passes the required internal tests and shows substantial nonoverlap under the current search protocol. We demonstrate improvements on MoSciBench, ScienceAgentBench, SciCode, and two real world scientific workflows, including a comparison with EvoSkill. More broadly, SKILLFOUNDRY provides a path for transforming fragmented scientific know how into reusable capabilities for more capable, scalable, and reliable scientific agents.

References

- Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026.
- Anthropic. Introducing the model context protocol. <https://www.anthropic.com/news/model-context-protocol>, November 2024. Anthropic announcement, accessed 2026-03-24.
- Anthropic. Agent skills - claude api docs. <https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview>, 2026. Accessed: 2026-03-24.
- Jeffrey Baron, Lars Sävendahl, Francesco De Luca, Andrew Dauber, Moshe Phillip, Jan M Wit, and Ola Nilsson. Short and tall stature: a new paradigm emerges. *Nature Reviews Endocrinology*, 11(12):735–746, 2015.
- Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023.
- Tianyi Chen, Yinheng Li, Michael Solodko, Sen Wang, Nan Jiang, Tingyuan Cui, Junheng Hao, Jongwoo Ko, Sara Abdali, Leon Xu, et al. Cua-skill: Develop skills for computer using agent. *arXiv preprint arXiv:2601.21123*, 2026.
- Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=6z4YKr0GK6>.
- Shimin Di, Xujie Yuan, Hanghui Guo, Chaoqian Ouyang, Zhangze Chen, Ling Yue, Libin Zheng, Jia Zhu, Shaowu Pan, Jian Yin, et al. Toolrosetta: Bridging open-source repositories and large language model agents through automated tool standardization. *arXiv preprint arXiv:2603.09290*, 2026.
- Elie N Farah, Robert K Hu, Colin Kern, Qingquan Zhang, Ting-Yu Lu, Qixuan Ma, Shaina Tran, Bo Zhang, Daniel Carlin, Alexander Monell, et al. Spatially organized cellular communities form the developing human heart. *Nature*, 627(8005):854–864, 2024.
- Shanghua Gao, Richard Zhu, Pengwei Sui, Zhenglun Kong, Sufian Aldogom, Yepeng Huang, Ayush Noori, Reza Shamji, Krishna Parvataneni, Theodoros Tsiligkaridis, et al. Democratizing ai scientists using tooluniverse. *arXiv preprint arXiv:2509.23426*, 2025.
- Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Gavin Li, Junze Zhang, et al. Biomni: A general-purpose biomedical ai agent. *biorxiv*, 2025.
- Zhengbo Jiao, Shaobo Wang, Zifan Zhang, Xuan Ren, Wei Wang, Bing Zhao, Hu Wei, and Linfeng Zhang. Agentic proposing: Enhancing large language model reasoning via compositional skill synthesis. *arXiv preprint arXiv:2602.03279*, 2026.
- Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 3102–3116, 2023.
- Fan Liu, Xiaozhao Zeng, and Hao Liu. Towards multimodal data-driven scientific discovery powered by LLM agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL <https://openreview.net/forum?id=kZHSvETWdi>.
- Jiacheng Miao, Joe R Davis, Yaohui Zhang, Jonathan K Pritchard, and James Zou. Paper2agent: Reimagining research papers as interactive and reliable ai agents. *arXiv preprint arXiv:2509.06917*, 2025.

- Chaoqian Ouyang, Ling Yue, Shimin Di, Libin Zheng, Linan Yue, Shaowu Pan, Jian Yin, and Min-Ling Zhang. Code2MCP: Transforming code repositories into MCP services. *arXiv preprint arXiv:2509.05941*, 2025. doi: 10.48550/arXiv.2509.05941.
- Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37: 126544–126565, 2024.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.
- Yuanhao Qu, Kaixuan Huang, Ming Yin, Kanghong Zhan, Dyllan Liu, Di Yin, Henry C Cousins, William A Johnson, Xiaotong Wang, Mihir Shah, et al. Crispr-gpt for agentic automation of gene-editing experiments. *Nature Biomedical Engineering*, pp. 1–14, 2025.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36: 68539–68551, 2023.
- Rongbo Shen, Lin Liu, Zihan Wu, Ying Zhang, Zhiyuan Yuan, Junfu Guo, Fan Yang, Chao Zhang, Bichao Chen, Wanwan Feng, et al. Spatial-id: a cell typing method for spatially resolved transcriptomics via transfer learning and spatial embedding. *Nature communications*, 13(1):7640, 2022.
- SkillHub. Skillhub: Claude skills & agent skills marketplace, 2026. URL <https://www.skillhub.club/>.
- SkillsMP. Skillsmp: Agent skills marketplace for the open skill.md ecosystem, 2026. URL <https://skillsmp.com/>.
- Kyle Swanson, Wesley Wu, Nash L Bulaong, John E Pak, and James Zou. The virtual lab of ai agents designs new sars-cov-2 nanobodies. *Nature*, 646(8085):716–723, 2025.
- The Tabula Muris Consortium. A single-cell transcriptomic atlas characterizes ageing tissues in the mouse. *Nature*, 583(7817):590–595, 2020. doi: 10.1038/s41586-020-2496-1.
- Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, Shengyan Liu, Di Luo, Yutao Ma, Hao Tong, Kha Trinh, Chenyu Tian, Zihan Wang, Bohao Wu, Yanyu Xiong, Shengzhu Yin, Minhui Zhu, Kilian Lieret, Yanxin Lu, Genglin Liu, Yufeng Du, Tianhua Tao, Ofir Press, Jamie Callan, Eliu Huerta, and Hao Peng. SciCode: A research coding benchmark curated by scientists. In *Advances in Neural Information Processing Systems*, volume 37, pp. 30624–30650, 2024.
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.
- Hanchen Wang, Yichun He, Paula P Coelho, Matthew Bucci, Abbas Nazir, Bob Chen, Linh Trinh, Serena Zhang, Kexin Huang, Vineethkrishna Chandrasekar, et al. Spatialagent: An autonomous ai agent for spatial biology. *bioRxiv*, pp. 2025–04, 2025a.
- Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. *arXiv preprint arXiv:2512.17102*, 2025b.
- Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

- Yi Wang, Zhenting Huang, Zhaohan Ding, Ruoxue Liao, Yuan Huang, Xinzijian Liu, Jiajun Xie, Siheng Chen, and Linfeng Zhang. Deploy-master: Automating the deployment of 50,000+ agent-ready scientific tools in one day. *arXiv preprint arXiv:2601.03513*, 2026.
- Zhizheng Wang, Qiao Jin, Chih-Hsuan Wei, Shubo Tian, Po-Ting Lai, Qingqing Zhu, Chi-Ping Day, Christina Ross, Robert Leaman, and Zhiyong Lu. Geneagent: self-verification language agent for gene-set analysis using domain databases. *Nature Methods*, 22(8): 1677–1685, 2025c.
- Jiaqi Wei, Yuejin Yang, Xiang Zhang, Yuhan Chen, Xiang Zhuang, Zhangyang Gao, Dongzhan Zhou, Guangshuai Wang, Zhiqiang Gao, Juntai Cao, et al. From ai for science to agentic science: A survey on autonomous scientific discovery. *arXiv preprint arXiv:2508.14111*, 2025.
- Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430*, 2026.
- Ling Yue, Ching-Yun Ko, Pin-Yu Chen, Shimin Di, and Shaowu Pan. Building MCP-native hierarchical AI scientist ecosystems: a perspective on scaling multi-agent scientific discovery. *Frontiers in Artificial Intelligence*, 9:1820375, 2026. doi: 10.3389/frai.2026.1820375.
- Martin JinYE Zhang, Kangcheng Hou, Kushal K Dey, Saori Sakaue, Karthik A Jagadeesh, Kathryn Weinand, Aris Taychameekiatchai, Poorvi Rao, Angela Oliveira Pisco, James Zou, et al. Polygenic enrichment distinguishes disease associations of individual cells in single-cell rna-seq data. *Nature genetics*, 54(10):1572–1580, 2022.

A Appendix

A.1 Method

A.1.1 *Tree State and Branch Prioritization*

Beyond its role as a topical taxonomy, the domain knowledge tree serves as a stateful controller for skill acquisition. Each node stores lightweight repository state, including linked resources, linked skills, recent verification outcomes, and a coarse coverage flag. These signals are maintained through machine readable registries for taxonomy structure, provenance, and skill metadata. As a result, the tree functions not only as an ontology over the target domain, but also as an index of the current library state.

This state is used to prioritize where the next mining cycle should focus. In particular, SKILL-FOUNDRY does not choose resources independently. It first identifies branches or leaves whose expected marginal value is high under the current repository state. Examples include leaves that already have useful resources but no executable skills, leaves supported only by low confidence starter skills, and leaves whose current skills repeatedly fail validation. This branch first prioritization reduces redundant search and turns a large heterogeneous resource space into a sequence of bounded skill acquisition problems.

Only the high level seed taxonomy is manually curated, and it is intentionally broad. It covers computational biology together with adjacent scientific software areas such as chemistry, visualization, workflows, spatial transcriptomic analysis, and scientific agents. Concrete subdomains and skill leaves are not manually enumerated. The tree is revised over time as broad branches are split when mining reveals stable subareas and stale or weakly supported branches are merged, collapsed, or marked for lower priority. In this sense, the tree is neither fixed nor purely data driven. It is a structured prior that is updated according to observed resource coverage and skill outcomes.

A.1.2 *Skill Packaging and Repository Integration*

After a branch is selected, SKILLFOUNDRY retrieves a branch-conditioned pool of candidate resources rather than performing an unrestricted crawl. From these resources, the system extracts an operational contract for each candidate skill. This contract specifies the task scope of the capability, expected inputs and outputs, environment and dependency assumptions, execution steps, provenance links, follow up guidance, and, when available, example invocations or test commands.

The extracted contract is then compiled into a concrete skill package. In the implementation, a compiled skill is more than a text description. It contains a human readable skill specification, machine readable metadata, provenance records, executable scripts, example assets, and repository level tests whenever such artifacts can be resolved. This packaging choice is important because it turns induced skills into reusable software objects that remain interpretable to agents while still being maintainable at the repository level.

Whenever possible, each candidate skill is paired with a smoke target or test command before integration. The resulting status is written to the shared skill registry so that later cycles can distinguish between untested, repaired, and verified entries. The same contract format is also reused in prompt driven skill design for user specified tasks. This means that task specific skill synthesis does not introduce a separate pathway for ad hoc tool creation, but instead reuses the same packaging and validation interface as the main mining loop.

A.1.3 *Hierarchical Validation, Repair, and Optimization*

The self improving behavior of SKILLFOUNDRY comes from a hierarchical validation process that goes beyond simple execution checks. The first layer is correctness oriented execution testing. The system runs a skill through its resolved smoke target or registered test command and checks whether the declared contract is actually executable. When this layer fails, the framework enters a repair loop that attempts to diagnose the failure, revise the imple-

mentation or test surface, and rerun the check. This establishes a minimum executability requirement before a skill can be considered for promotion.

The second layer asks whether the skill is useful as an agent abstraction. A skill may run successfully while still being too incomplete, too brittle, or too poorly structured to help an agent. To address this, SKILLFOUNDRY constructs skill versus no skill comparisons on task specific cases and checks whether the maintained wrapper provides a meaningful advantage over an ad hoc baseline. If the advantage is weak or inconsistent, the framework enters an optimization loop that revises the skill and benchmarks it again.

In addition, the framework uses two complementary validation surfaces that are especially important in scientific settings. System testing is used for skills whose realistic deployment depends on infrastructure such as SLURM, environment modules, or cluster-specific execution conventions. Synthetic data testing is used when live resources are unavailable, unstable, rate limited, or too expensive for routine regression testing. Rather than approximating the full downstream task, these tests verify whether the skill handles controlled inputs correctly, respects expected file and argument contracts, and produces stable outputs under deterministic mock conditions. This makes them useful for catching interface errors, broken assumptions, and incomplete packaging even when real world execution is not practical.

A.1.4 Novelty Review and Tree Update

Passing validation is still not sufficient for acceptance into the library. A candidate skill may be correct and useful while contributing little marginal value if it largely duplicates an existing abstraction. SKILLFOUNDRY therefore performs a final novelty and redundancy review against both the local library and external skill ecosystems. For the external review, the system derives search terms from the candidate description, retrieves matches from SkillHub and SkillsMP, and judges whether any retrieved skill completes the same task with the same method scope. Retrieved matches are also compared by provenance and intended coverage when deciding whether to discard or merge a candidate. This search based procedure can miss functionally similar skills with different names or descriptions, so the resulting novelty estimate is protocol dependent rather than an absolute uniqueness guarantee.

Depending on the outcome, a candidate skill may be accepted as a new leaf, merged into an existing entry, marked for review, or deprioritized in future updates. These decisions are then written back to the domain tree. Skills that pass become new or improved leaves. Skills that repeatedly fail leave a branch marked as only partially covered. Skills that are found to be redundant or low value may trigger consolidation or pruning of the corresponding branch. In this way, validation outcomes do not merely filter skills locally, but actively reshape the search structure used in subsequent mining rounds.

A.1.5 Parallel Acquisition and Controlled State Mutation

Although branch level exploration can be parallelized, SKILLFOUNDRY is conservative in how it mutates shared repository state. Resource search and skill construction can run in isolated worker workspaces, but only leaf owned artifacts are merged automatically. Shared global files, including registries, reports, and site summaries, remain refresh owned. This separation prevents concurrent workers from writing inconsistent global state and ensures that each mining cycle begins from a coherent repository snapshot.

At a larger timescale, the framework supports checkpointable campaigns that alternate branch focused mining with batched evaluation over pending skills. This allows the library to grow incrementally while preserving auditability. More broadly, it reflects a central design principle of SKILLFOUNDRY. Exploration may be parallel, but integration should remain explicit, deterministic, and easy to inspect.

A.1.6 Implementation Details

We instantiate SKILLFOUNDRY as a staged pipeline whose internal modules align with the phases in Fig. 1. Specifically, `tree_check` corresponds to *Domain Tree*, `resource_search` to *Resource Mining*, `skill_build` together with task-specific `design_skill` to *Skill Extraction*, `skill_test`, `layer1_fix`, `layer2_benchmark`, and `layer2_optimize` to *Skill Testing*, and `refresh` plus `novelty_check` to the post-validation tree update supporting *Tree Expansion* and *Tree Refinement*.

For model allocation, we use GPT-5.4 with medium reasoning effort for `tree_check`, `skill_build`, `refresh`, and `design_skill`; GPT-5.4 with high reasoning effort for `resource_search`; and GPT-5.4-mini with medium reasoning effort for `skill_test`, `layer1_fix`, `layer2_benchmark`, `layer2_optimize`, and `novelty_check`. This allocation makes `resource_search` the only default high-effort stage in the main loop, reflecting that searching and triaging heterogeneous scientific resources is the most reasoning-intensive step. By contrast, skill construction remains on the larger model at medium effort, while validation, repair, benchmarking, and novelty review are delegated to the smaller model for efficiency.

For downstream evaluation, we use Codex with GPT-5.1-Codex-mini and the default medium reasoning effort on MoSciBench. For ScienceAgentBench and SciCode, we instead use GPT-5.4-mini with medium reasoning effort. For cell type annotation, we use GPT-5.3-Codex with medium reasoning effort. In the cell type annotation comparison, both Codex settings use the same model, dataset, prompt, search and tool permissions, and evaluation protocol. The baseline may search, reason, and select its own workflow. Its PCA and k means strategy is therefore selected by the agent rather than prescribed by the prompt. For the scDRS transfer experiment, SKILLFOUNDRY first synthesizes task specific skills and then exports them to Biomni as external skill resources. Within each benchmark or task, we keep the underlying agent configuration fixed and vary only the availability of the relevant SKILLFOUNDRY skills.

A.1.7 Prompt Templates

We use stage-conditioned prompt templates rather than a single monolithic instruction. A cycle controller governs iterative repository mining, specialized prompts handle isolated leaf workers and user-specified skill design, and a hierarchical evaluation suite repairs failures, benchmarks skill utility, and checks novelty.

We show abbreviated prompt templates rather than all instantiated prompts. Repeated fields such as repository summaries, artifact directories, and strict JSON response schemas are compressed for readability.

System Prompt: You are operating inside the SkillFoundry repository. Follow the workflow contract in `experiments.md` and reuse existing repository scripts instead of creating parallel registries.

Prompt Inputs: STAGE={stage}, MODE=cycle, repository summary, focus leaves, and a stage-specific objective.

Key Constraints: Work directly in the current checkout when appropriate; reuse repository scripts and registries; keep verification labels honest.

Response Contract: Return only a JSON object matching the stage schema.

System Prompt: You are operating inside an isolated workspace copy of the SkillFoundry repository. This is a parallel worker stage for a single taxonomy leaf.

Prompt Inputs: STAGE={stage}, MODE=parallel_leaf_stage, repository summary, target leaf, stage objective, and artifact directory.

Key Constraints: Stay scoped to the assigned leaf; prefer edits under skill-local files and leaf-specific tests; do not modify shared global files such as registries, site files, `README.md`, or planning documents.

Response Contract: Return only a JSON object, and record any shared-file follow-up in structured fields such as `repo_changes`, `blockers`, or `next_steps`.

System Prompt: You are operating inside the SkillFoundry repository. This mode is for a user-specified task: search resources first when the request needs additional materials, then implement and test the resulting skill.

Prompt Inputs: STAGE=design_skill, MODE=design_skill, user task prompt, repository summary, and focus leaves.

Key Constraints: Search for canonical papers, repositories, notebooks, package references, or workflows when needed; build or refine one concrete skill path; add or update repository-level tests and, when relevant, a Slurm execution path.

Response Contract: Return only a JSON object matching the design-skill schema.

System Prompt: Hierarchical evaluation operates in multiple modes: failure repair, task-specific benchmarking, benchmark-driven optimization, and novelty checking.

Prompt Inputs: Skill metadata, repository summary, failure records or benchmark results, local similarity candidates, and an artifact directory.

Key Constraints: Diagnose failures before editing; apply targeted fixes rather than broad refactors; benchmark with-skill versus no-skill baselines conservatively; optimize only when benchmark deficits are clear; and assess overlap against both local skills and external ecosystems.

Response Contract: Return structured JSON containing repair findings, benchmark scores, optimization actions, or novelty scores, depending on the evaluation sub-stage.

System Prompt: Design or refine an experiment-only portable skill for a specific task.

Prompt Inputs: Task title, task goal, target directory, preferred methods, family-specific starter guidance, and required deliverables.

Key Constraints: Modify only files under the task-specific target directory; do not edit repository-level registries, shared skill libraries, site files, README.md, experiments.md, or planning files.

Response Contract: Produce only the required experiment-local deliverables for the designated task.

A.2 Experiments

A.2.1 ScienceAgentBench and SciCode

For ScienceAgentBench, we evaluate the Codex baseline, Codex with EvoSkill, and Codex with SKILLFOUNDRY on the same held out test set. We randomly assign 20% of the selected tasks to the EvoSkill training set because EvoSkill acquires skills from task feedback. The remaining tasks are used for all reported comparisons. SKILLFOUNDRY does not use the training split or task specific feedback. We report valid output rate, success rate, and mean CodeBERT score using the benchmark evaluation protocol. For SciCode, we compare the same Codex agent with and without SKILLFOUNDRY skills and report sampled test accuracy. In both benchmarks, the base agent configuration is held fixed when SKILLFOUNDRY skills are added.

A.2.2 scDRS

Here we provide the evaluation criteria, the replicate-level outcomes, and representative analysis figures produced by Biomni with and without the task-specific skills synthesized by SKILLFOUNDRY.

We compare Biomni with and without SKILLFOUNDRY skills on the scDRS workflow using the TMS FACS ([The Tabula Muris Consortium, 2020](#)) scRNA-seq dataset together with height GWAS. Each setting is run three times and compared against an expert reference analysis. We evaluate the outputs from two complementary perspectives. The qualitative score ranges from 0 to 7, with one point assigned for each of the following criteria: returning individual cell-level, cell-type-level, and gene-level analyses, applying proper FDR correction, identifying chondrocytes as the top associated cell type ([Baron et al., 2015](#)), performing within-cell-type heterogeneity analysis, and returning useful output files. The quantitative metric is the RMSE between the cell-level disease scores produced by each run and the expert reference results. We report both metrics because numerical agreement alone does not guarantee that the resulting scientific artifact is complete and interpretable.

Replicate	Setting	Qualitative score	RMSE	Notes
1	Biomni	3	0.16	Filtering option not enabled
2	Biomni	4	0.16	Minor deviation from expert pipeline
3	Biomni	6	0.00	Exact scores but incomplete analysis
4	Biomni + SKILLFOUNDRY	4	0.06	Nearly complete analysis
5	Biomni + SKILLFOUNDRY	7	0.00	All criteria met, exact numerical match
6	Biomni + SKILLFOUNDRY	3	0.00	Exact scores but incomplete reporting

Table 4: Replicate-level scDRS evaluation. Replicates 1 to 3 correspond to Biomni without SKILLFOUNDRY skills, and replicates 4 to 6 correspond to Biomni with SKILLFOUNDRY skills.

Table 4 shows that SKILLFOUNDRY improves both accuracy and robustness. Biomni+SKILLFOUNDRY is the only setting that satisfies all seven qualitative criteria, and it also produces exact agreement with the expert scores in two of three runs, compared with one of three runs for Biomni alone. Its mean RMSE is substantially lower, decreasing from 0.11 to 0.02. The replicate-level breakdown also shows why both metrics are needed. Some runs recover the correct cell-level scores but still fail to return the full set of analyses or outputs needed for scientific interpretation.

Fig. 5 and Fig. 6 compare representative analysis figures produced in the two settings. Without the synthesized skills, Biomni generates only a basic ranking plot based on nominal significance. This is enough to show a rough ordering of top cell types, but it does not summarize FDR-supported signal or within-cell-type heterogeneity. With SKILLFOUNDRY skills, Biomni produces a richer and more interpretable figure that combines ranked cell-type associations with an additional panel showing the proportion of FDR-significant cells in each cell type. As a result, the with-skill figure provides a more complete summary of both statistical support and cell-type heterogeneity, which helps explain the higher qualitative scores.

A.2.3 Dashboard

We also provide a web dashboard for browsing, monitoring, and managing the mined skill library. The dashboard presents each skill together with its verification status, domain assignment, compute profile, and other metadata, and supports interactive search and filtering across the library. A taxonomy explorer further situates each skill within the domain tree and summarizes coverage at different levels of granularity. In addition, the interface exposes provenance information, allowing users to inspect the resources from which a skill was mined and to access associated artifacts such as the skill specification, metadata, references, and tests. Fig. 7 and Fig. 8 show example views for a cell type annotation skill and scDRS-related skills, respectively.

A.3 Large Language Model Usage Disclosure

Large language models were used as components of SKILLFOUNDRY for resource search, skill construction, repair, benchmarking, novelty review, and downstream agent evaluation. The authors are responsible for all generated outputs and reported claims. Exact model identifiers, model snapshots, prompts, run dates, tool permissions, and execution logs are provided in the appendix and accompanying repository. Bibliographic records, quantitative aggregates, plots, and manuscript claims were checked against archived source materials and experiment artifacts.

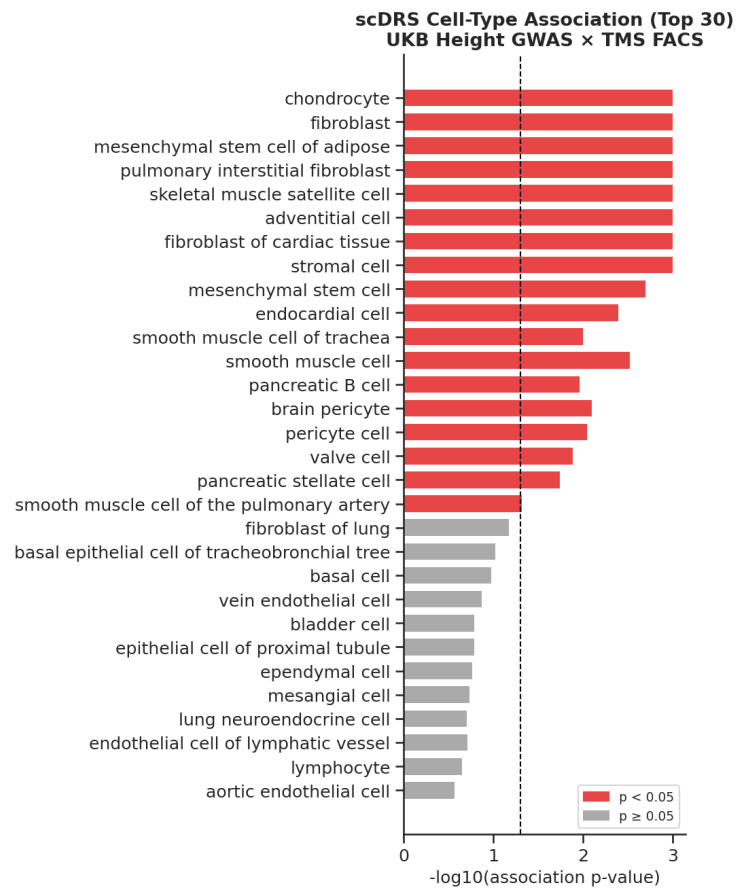


Figure 5: Representative scDRS output generated by Biomni without SKILLFOUNDRY skills. The figure ranks cell types by association strength and marks nominal significance, but does not summarize FDR-supported signal or within-cell-type heterogeneity.

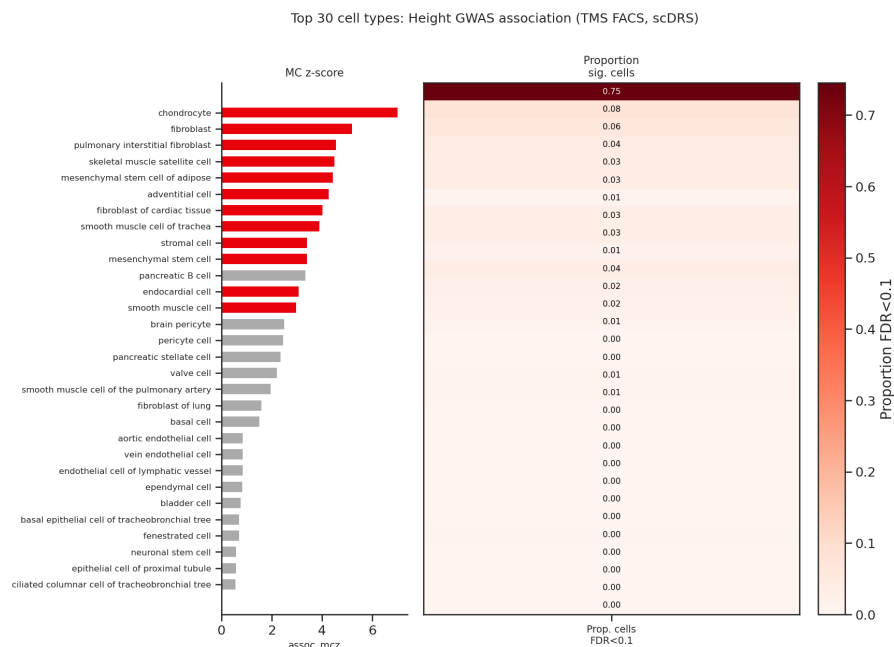


Figure 6: Representative scDRS output generated by Biomni with SKILLFOUNDRY skills. In addition to ranking cell types by association strength, the figure reports the proportion of FDR-significant cells within each cell type, providing a richer summary of statistical support and cell-type heterogeneity.

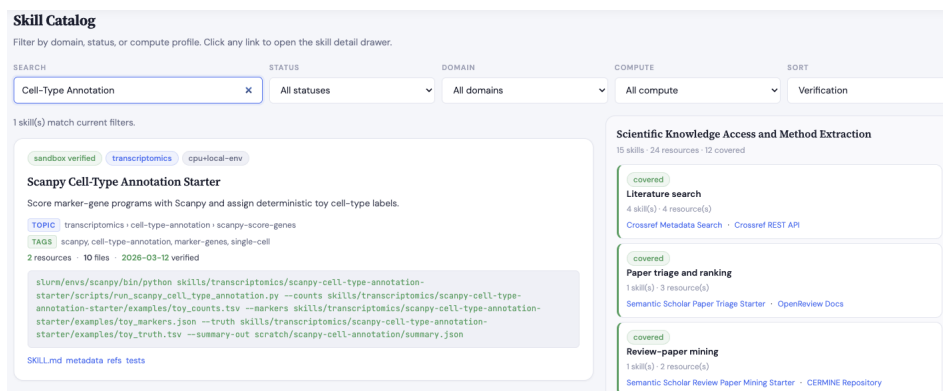


Figure 7: Web dashboard view of a cell type annotation skill. The interface shows searchable skill metadata, taxonomy placement, and links to associated artifacts and source resources.

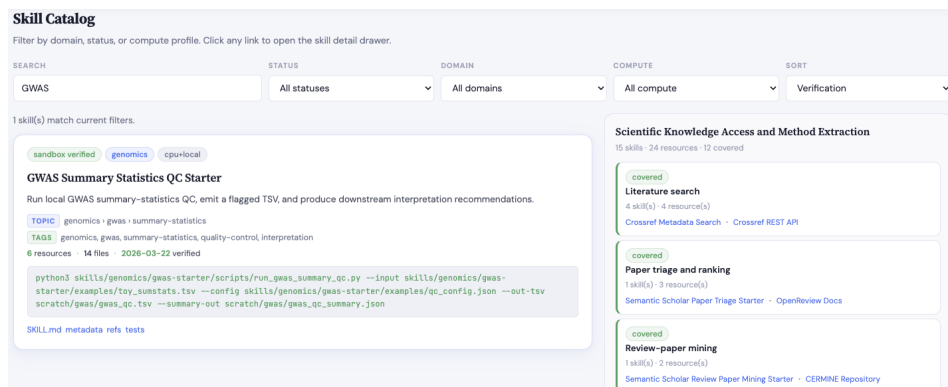


Figure 8: Web dashboard view of scDRS-related skills. The interface supports search and filtering over runnable skills while exposing their domain assignments, verification status, and provenance information.