nature portfolio

https://doi.org/10.1038/s41586-026-11044-y

Supplementary information

Reimagining research papers as interactive and reliable AI agents

In the format provided by the authors and unedited

Supplementary Information

Reimagining research papers as interactive and reliable AI agents

Jiacheng Miao[1,2,*], Joe R. Davis[1], Yaohui Zhang[3], Jonathan K. Pritchard[1,4], James Zou[2,3,5,*]

[1] Department of Genetics, Stanford University, Stanford, California, USA
[2] Department of Biomedical Data Science, Stanford University, Stanford, California, USA
[3] Department of Electrical Engineering, Stanford University, Stanford, California, USA
[4] Department of Biology, Stanford University, Stanford, California, USA
[5] Department of Computer Science, Stanford University, Stanford, California, USA

* Correspondence: jcmiao@stanford.edu; jamesz@stanford.edu

Contents

Supplementary Note to "Reimagining Research Papers as Interactive and Reliable AI Agents"

Contents

1 Extended related work

AI agents for scientific discovery

Recent advances highlight the promise of agents for accelerating discovery. For example, the Virtual Lab framework organizes teams of AI scientist agents that collaboratively design and execute research projects across biology and chemistry [1]. Similarly, Google's AI co-scientist serves as a virtual collaborator, assisting with hypothesis generation and research proposal development [2]. Sakana AI's co-scientist aims to automate the research lifecycle—from ideation to publication [3]. FutureHouse provides an AI scientist platform designed for diverse scientific tasks [4]. Alongside these general-purpose platforms, specialized agents are also emerging for specific domains [5]. For example, CellVoyager introduces an agentic system for autonomous analysis of single-cell omics data [6]. Biomni is an AI agent for diverse biological tasks [7]. These systems demonstrate that agents can not only execute code, but also generate hypotheses, evaluate uncertainty, and adapt methods to new datasets. Recent work has also explored automatic code generation from scientific text [8, 9]. Paper2Agent complements this emerging paradigm by generalizing the concept: any research paper can be converted into an agent that embodies the knowledge and methods described in the publication.

Executable papers and reproducibility platforms

Efforts to make research outputs more executable and accessible have been ongoing for years. Executable papers—such as those proposed in Elsevier's Executable Paper Grand Challenge [10] and more recent Jupyter Notebook–backed publications [11]—sought to merge narrative text with runnable code. These approaches increased reproducibility but still required substantial technical familiarity to engage with fully. The Papers with Code initiative [12] similarly aimed to bridge papers and implementations by linking publications to open-source repositories. More recently, executable artifact platforms such as Binder [13] and CodeOcean [14] provide containerized environments for running paper-associated code, while paper-to-code systems like Paper2Code [8] automatically generate functional code repositories from research papers. While these efforts improved reproducibility, there were still substantial barriers to understanding, customizing, and applying code to new projects.

2 Comparison of Paper2Agent with related systems

Here we present a comparison table of Paper2Agent and related systems.

3 Details for Paper2Agent

Below we attach the pseudocode for Paper2Agent that turns a paper's code repository into an MCP server.

Step 0 Prompt: Language and GPU Detection

```
# Language and GPU Detection
```

Table 1: Comparison of Paper2Agent with related systems for computational reproducibility and paper operationalization.

Feature	**Binder**	**CodeOcean**	**Paper2Code**	**Biomni**	**CellVoyager**	**Paper2Agent**
Input	Code repository	Code repository	Paper only	User query	Dataset + paper summary	Paper + code for agent generation; user queries for agent execution
Primary output	Executable environment	Executable environment	Generated code repo	Code-based analysis	Jupyter notebook	MCP server (tools + resources + prompts)
Requires existing codebase	Yes	Yes	No	No	No	Yes (when available)
Automated tool extraction	No	No	No	No	No	Yes
Structured tool interface (API schema)	No	No	No	No	No	Yes (MCP)
Natural language interaction	No	No	No	Yes	Yes	Yes
Multi-paper agent composition	No	No	No	No	No	Yes
Cross-domain generalization	N/A	N/A	ML papers	Biomedicine	scRNA-seq	Any domain
Reusable by other agents/LLMs	No	No	No	No	No	Yes (via MCP protocol)
User needs to understand code	Yes	Yes	Yes	No	No	No
Automated env setup & repair	Manual (Dockerfile)	Managed	No	Pre-installed env	Manual	Yes
Paper knowledge as resource	No	No	No	Partial (retrieval)	Paper summary	Yes (full text + suppl.)
Workflow templates (prompts)	No	No	No	No	No	Yes

Algorithm 1: Paper2Agent: Automated MCP Agent Generation from Research Repositories

Input: GitHub repository URL, language hint (`auto/python/r/cli`)
Output: Deployable MCP server exposing repository tools as AI-callable functions

`// Phase 1: Repository Analysis`
Clone the GitHub repository (with submodules);
if *language* = `auto` **then**
    language ← LLM-DETECT(repo README, file structure);
    if *detection fails* **then** language ← heuristic fallback;
Load language-specific prompt templates;

`// Phase 2: Environment Setup & Tutorial Discovery`
Create an isolated environment and install dependencies;
tutorials ← scan repository for notebook / script / example files;

`// Phase 3: Tutorial Execution`
executed ← {};
foreach tutorial *t* in *tutorials* **do**
    Execute *t* in the isolated environment;
    Record gold-standard outputs (figures, tables, metrics);
    Add result to executed;

`// Phase 4: Tool Extraction & MCP Integration`
tools ← {};
foreach executed tutorial *n* in *executed* **do**
    Extract reusable functions from *n* as standalone tool modules;
    Generate unit tests for each extracted tool;
    Add tool modules to tools;
Integrate all tools into a unified MCP server;

`// Phase 5: Validation & Deployment`
Generate and validate the dependency manifest;
Install the MCP server for AI assistant access;
return MCP server;

```
Analyze this GitHub repository to determine its primary usage type and GPU
requirements.

## README Content

```
{{README_CONTENT}}
```

Tutorial/Example Files

{{TUTORIAL_FILES}}
```

```
## Task 1: Language Classification

Classify this repository into ONE of three categories:

1. **cli** - Command-line tool that users run from terminal
   - README shows usage like: `python script.py args`, `Rscript script.R args`,
   `./tool.py`
   - Users interact via command line arguments, not by writing code
   - Examples: bioinformatics pipelines, data processing scripts

2. **python** - Python library/package that users import and use in their Python
code
   - README shows usage like: `import package`, `from package import ...`
   - Users write Python code that calls the library functions
   - Has tutorials as Jupyter notebooks (.ipynb) or Python code examples

3. **r** - R package/library that users load and use in their R code
   - README shows usage like: `library(package)`, R code with `<-` assignment
   - Users write R code that calls the package functions
   - Has tutorials as R scripts (.R) or R Markdown (.Rmd)

## Task 2: GPU Requirement Detection

Determine if this repository requires a GPU to run.

**First, read the requirements files** in the repository to check for GPU-specific
packages:
- Look for `requirements.txt`, `setup.py`, `pyproject.toml`, `environment.yml`, or
`DESCRIPTION` (R)
- Use the Read tool to examine these files

**Strong GPU indicators (requires_gpu = true):**
- CUDA-specific packages: `torch.*+cu*`, `tensorflow-gpu`, `cupy`, `cudf`, `cuml`,
`rapids`
- PyTorch/TensorFlow with CUDA builds in requirements (e.g., `torch==2.0.0+cu118`)
- Code using `torch.cuda`, `tf.device('/GPU')`, CUDA kernels
- README mentions GPU requirements, CUDA version, or NVIDIA dependencies
- Command-line args like `--gpu`, `--device cuda`, `--cuda`
- Environment setup mentioning CUDA toolkit or cuDNN

**Weak/No GPU indicators (requires_gpu = false):**
- CPU-only deep learning (e.g., `torch` without CUDA suffix, `tensorflow-cpu`)
- No deep learning frameworks
- Pure R packages (unless using GPU-accelerated libraries like `gputools`)
- Standard data processing libraries (pandas, numpy, scikit-learn)
- README explicitly states CPU-only or no GPU mentioned

## Important Rules

- **Self-description hint**: If the README explicitly says "this is a command line
tool", "this is an R package", or "this is a Python package/library", this is a
strong hint! However, verify by checking how users actually interact with it.
```

```
- **Command-line invocations**: If the README/wiki shows usage like `Rscript
something.R [args]` or `python something.py [args]`, this is very likely a
command-line tool even though it's written in R or Python. Classify as **cli**, NOT
r or python.
- Focus on HOW users are expected to USE the tool, not what language it's written
in.
- For GPU detection, be conservative: only mark `requires_gpu: true` if there's
clear evidence the code needs GPU acceleration to function properly.

## Response Format

Respond with ONLY a JSON object (no markdown code blocks):

{"language": "cli", "requires_gpu": false, "language_reason": "brief explanation",
"gpu_reason": "brief explanation"}

or

{"language": "python", "requires_gpu": true, "language_reason": "brief explanation",
"gpu_reason": "e.g., torch+cu126 in requirements, --gpu flag in README"}

or

{"language": "r", "requires_gpu": false, "language_reason": "brief explanation",
"gpu_reason": "brief explanation"}
```

Step 1 Prompt: Environment Setup & Tutorial Discovery Coordinator

```
# Environment Setup & Tutorial Discovery Coordinator

## Role
Orchestrator agent that coordinates parallel environment setup and tutorial
discovery for scientific research codebases. You manage subagent execution, handle
errors, validate outputs, and ensure successful completion of both tasks.

## Core Mission
Transform scientific research codebases into reusable tools by coordinating two
specialized agents working in parallel to prepare the codebase for tool extraction.

## Subagent Capabilities
- **environment-python-manager**: Comprehensive Python environment setup with uv,
pytest configuration, and dependency management
- **tutorial-scanner**: Systematic tutorial identification, classification, and
quality assessment for tool extraction

## Input Parameters
- `repo/${github_repo_name}`: Repository codebase directory
- `github_repo_name`: Project name (exact capitalization from context)
- `PROJECT_ROOT`: Absolute path to project directory
- `UV_PYTHON_ENV`: Target uv python environment name
```

````
- `tutorial_filter`: Tutorial filter for file path or title matching. Value:
"${tutorial_filter}" (if "(none)", no filter is applied)

## Expected Outputs
- `reports/environment-manager_results.md`: Environment setup summary
- `reports/tutorial-scanner.json`: Complete tutorial analysis
- `reports/tutorial-scanner-include-in-tools.json`: Filtered tutorials for tool
creation

---

## Execution Coordination

### Phase 1: Parallel Agent Launch
Execute both agents simultaneously using the Task tool with concurrent calls:

```
Task 1: environment-python-manager
- Mission: Set up ${github_repo_name}-env with Python >=3.10
- Working directory: Current directory (NOT repo/ subfolder)
- Requirements:
  1. Create uv environment: `uv venv ${github_repo_name}-env --python 3.12`
  2. Install project dependencies from repo/${github_repo_name}/pyproject.toml or
  requirements.txt
  3. Install ADDITIONAL packages (CRITICAL - do not skip):
     `uv pip install --python ${github_repo_name}-env/bin/python fastmcp pytest
     pytest-asyncio papermill nbclient ipykernel imagehash`
  4. Register Jupyter kernel for project environment:
     `${github_repo_name}-env/bin/python -m ipykernel install --user --name
     ${github_repo_name} --display-name "Python (${github_repo_name})"`
  5. Configure pytest
  6. Verify kernel registered: `jupyter kernelspec list | grep ${github_repo_name}`
- Output: reports/environment-manager_results.md

Task 2: tutorial-scanner
- Mission: Scan repo/${github_repo_name}/ for tool-worthy tutorials
- Filter parameter: "${tutorial_filter}" (ONLY apply if NOT "(none)" - if value is
"(none)", scan ALL tutorials without filtering)
- Requirements: Strict filtering when filter provided, quality assessment, JSON
output generation
- Output: reports/tutorial-scanner.json +
reports/tutorial-scanner-include-in-tools.json + reports/suggested-tools.json
```

### Phase 2: Progress Monitoring & Error Recovery

**Timeout Management:**
- Monitor agent progress with 10-minute timeout per agent
- Implement graceful failure handling for long-running operations

**MANDATORY SUBAGENT COMPLETION CHECK:**
Before returning ANY final result or summary:
````

```
1. **Retrieve results from BOTH spawned subagents** - call the subagent result
retrieval for each one
2. **Verify subagent count**: You spawned 2 subagents, you MUST have 2 completed
results
3. **DO NOT return early**: Even if one agent finishes quickly, wait for BOTH of
them
4. **If a subagent is still running**, keep waiting - do not output final summary
until it completes
5. **Explicit completion verification**: After both subagents complete, verify you
have results from each before generating final output

**Error Recovery Strategies:**
- **Environment failures**: Provide alternative Python versions (3.10, 3.11, 3.12)
- **Tutorial scanning failures**: Attempt partial scanning with error reporting
- **Resource conflicts**: Ensure agents don't interfere with shared directories
- **Filter failures**: Validate filter syntax and provide clear error messages

### Phase 3: Output Validation Framework

**Environment Validation:**
- Verify environment-manager_results.md exists and contains required sections
- Confirm environment activation commands are properly documented
- Validate Python version compliance (>=3.10)

**Tutorial Validation:**
- Validate JSON schema compliance for both output files
- Cross-reference tutorial paths with actual repository structure
- Verify filter results match expected criteria
- Ensure no legacy/deprecated content marked as "include-in-tools"

**Quality Checks:**
- Environment: Successful dependency installation, pytest configuration
- Tutorials: Proper classification, quality standards applied consistently

---

## Tutorial Filter Coordination

**Filter value: "${tutorial_filter}"**

When `tutorial_filter` is NOT "(none)":
- Pass exact filter string to tutorial-scanner: `"${tutorial_filter}"`
- Ensure case-insensitive matching for both file paths and tutorial titles
- Validate OR logic: match if EITHER file path OR title matches
- **Strict enforcement**: No fallback to all tutorials if no matches found
- Report match statistics in final summary

When `tutorial_filter` IS "(none)":
- Do NOT apply any filtering - scan and evaluate ALL tutorials
- Standard quality assessment still applies

---
```

````
## Success Criteria & Completion

### Completion Requirements
Both agents must complete successfully before marking the task complete. Use [Y] to
confirm success and [N] to confirm failure. Provide a one-line reason for success or
failure. If there are any failures, fix them and run the coordination again for up
to 3 iterations.

**BEFORE OUTPUTTING ANY FINAL SUMMARY:**
1. You spawned 2 subagents (environment-python-manager and tutorial-scanner)
2. Verify you have retrieved results from BOTH subagents
3. If EITHER subagent result is missing or still pending, DO NOT output final
summary - wait for it
4. Only after confirming BOTH subagent results are retrieved, proceed with final
output

- [ ] **Environment Setup**: Environment setup completed with no critical errors
(result retrieved from environment-python-manager)
- [ ] **Tutorial Scanning**: Tutorial scanning completed with valid JSON outputs
(result retrieved from tutorial-scanner)
- [ ] **Output Generation**: All required output files generated and validated
- [ ] **Quality Control**: No deprecated/legacy content incorrectly classified

### Consolidated Reporting
Generate final summary combining both agent results:
```
Environment Setup & Tutorial Discovery Complete

Environment Status:
- Environment: ${github_repo_name}-env
- Python Version: [version]
- Dependencies: [count] packages installed
- Activation: source ${github_repo_name}-env/bin/activate

Tutorial Analysis:
- Total tutorials scanned: [count]
- Tutorials included in tools: [count]
- Filter applied: [filter_status]
- Quality assessment: [pass/issues]

Execution Metrics:
- Environment setup time: [duration]
- Tutorial scanning time: [duration]
- Total execution time: [duration]
```

### Error Reporting
If either agent fails:
- Document specific failure points
- Provide actionable remediation steps
- Attempt automatic recovery where possible
- Escalate to user only for unrecoverable failures
````

```
---

## Variable Standards
- Use `${github_repo_name}` consistently throughout
- Maintain exact capitalization from input parameters
- Ensure environment paths are relative to current working directory
- Standardize filter parameter passing between supervisor and subagents
```

Step 2 Prompt: Tutorial Execution Coordinator

```
# Tutorial Execution Coordinator

## Role
Orchestrator agent that coordinates **parallel** tutorial execution by launching
multiple tutorial-executor subagents simultaneously (one per tutorial) to generate
gold-standard outputs. You oversee parallel execution progress, handle errors,
validate outputs, and ensure successful completion.

## Core Mission
Transform tutorial materials into executable, validated notebooks with gold-standard
outputs for downstream tool extraction by coordinating **parallel** tutorial
execution across multiple agents.

## Subagent Capabilities
- **tutorial-executor**: Comprehensive tutorial execution specialist that handles
notebook preparation, environment management, iterative error resolution, and output
generation for ONE specific tutorial. Launch one agent per tutorial in parallel.

## Input Requirements
- `reports/tutorial-scanner-include-in-tools.json`: List of tutorials requiring
execution
- `${github_repo_name}-env`: Pre-configured Python environment for execution
- Repository structure under `repo/${github_repo_name}/`
- `api_key`: Optional API key for tutorials requiring external API access:
"${api_key}"

## Expected Outputs
- `notebooks/${tutorial_file_name}/${tutorial_file_name}_execution_final.ipynb`:
Final validated notebooks
- `notebooks/${tutorial_file_name}/images/`: Extracted figures and visualizations
- `notebooks/${tutorial_file_name}/data/`: Pre-generated benchmark data files
- `notebooks/${tutorial_file_name}/data/data.py`: Data generation script for
reproducible benchmarks
- `reports/executed_notebooks.json`: Complete execution summary with GitHub URLs

---

## Execution Coordination

### Phase 1: Pre-Execution Validation
```

````
**Input Validation:**
- Verify `reports/tutorial-scanner-include-in-tools.json` exists and contains valid
tutorials
- Confirm `${github_repo_name}-env` environment is available and functional
- Validate repository structure and tutorial file accessibility
- Check for required tools (papermill, jupytext, image extraction scripts)

**Environment Preparation:**
- Test environment activation: `source ${github_repo_name}-env/bin/activate`
- Verify essential dependencies are installed (papermill, nbclient, ipykernel,
imagehash)
- Ensure repository paths are accessible from current working directory

**API Key Integration:**
- When API key is provided ("${api_key}"), instruct tutorial-executor to:
  - Detect notebooks requiring API keys (OpenAI, Anthropic, Gemini, AlphaGenome, ESM
  etc.)
  - Inject API key assignments at the beginning of notebooks:
    ```python
    # API Configuration
    api_key = "${api_key}"
    openai.api_key = api_key  # For OpenAI
    # client = anthropic.Anthropic(api_key=api_key)  # For Anthropic
    # etc.
    ```
  - Handle common API patterns (openai, anthropic, google-generativeai, etc.)
  - Document API key injection in execution logs

### Phase 2: Parallel Tutorial Execution Launch

**Parallel Agent Coordination:**
For each tutorial in `reports/tutorial-scanner-include-in-tools.json`, launch a
separate tutorial-executor agent in parallel:

```
Task: tutorial-executor (one per tutorial, all launched in parallel)
- Mission: Execute ONE specific tutorial: ${tutorial_path}
- Input: Single tutorial entry from tutorial-scanner-include-in-tools.json
- Environment: ${github_repo_name}-env
- API Key: "${api_key}" (if provided, inject into notebooks requiring API access)
- Requirements: Generate execution notebook, handle errors, extract images
- Benchmark Data: Generate data/data.py and CSV files for reproducible benchmarks
(REQUIRED)
- Output:
notebooks/${tutorial_file_name}/${tutorial_file_name}_execution_final.ipynb +
images/ + data/
```

**Example:** If there are 5 tutorials, launch 5 tutorial-executor agents
simultaneously:
- Agent 1: Execute `quick_start.ipynb`
- Agent 2: Execute `example_analysis_workflow.ipynb`
- Agent 3: Execute `batch_variant_scoring.ipynb`
````

````
- Agent 4: Execute `essential_commands.ipynb`
- Agent 5: Execute `visualization_modality_tour.ipynb`

**CRITICAL: Capture Outputs for Benchmark Questions:**
When executing notebooks, add `print(result.head(20))` after analysis steps that
produce result dataframes/objects. This captures numeric values (scores, p-values,
counts) needed for benchmark question generation in Step 8. Without visible outputs,
no benchmark questions can be created.

```python
# Example: Add head() to show results
result = blitz.gsea(signature, library)
print(result.head(20))  # Captures top results for benchmarks
```

**CRITICAL: FIRE-AND-FORGET - DO NOT WAIT FOR SUBAGENTS:**
- Launch ALL subagents with `run_in_background: true`
- **DO NOT use TaskOutput** - this accumulates context and causes overflow
- **EXIT IMMEDIATELY** after launching all subagents
- Each subagent writes its own outputs directly to files:
  - Notebooks to: `notebooks/<tutorial_name>/`
  - Individual report to: `reports/executed_notebook_<tutorial_name>.json`
  - Completion marker: `reports/.done_<tutorial_name>` (empty file to signal
  completion)
- Timeout: 60-minute maximum per tutorial

**After launching all subagents, poll for completion using filesystem checks:**
```bash
# Wait for all .done marker files (launched N subagents)
while [ $(ls reports/.done_* 2>/dev/null | wc -l) -lt N ]; do
  sleep 30
  echo "Waiting for subagents... $(ls reports/.done_* 2>/dev/null | wc -l)/N
  complete"
done
echo "All subagents completed!"
```

**After all complete, merge individual reports:**
```bash
python3 -c "
import json, glob
merged = {}
for f in glob.glob('reports/executed_notebook_*.json'):
    with open(f) as fp:
        merged.update(json.load(fp))
with open('reports/executed_notebooks.json', 'w') as fp:
    json.dump(merged, fp, indent=2)
print(f'Merged {len(merged)} tutorials')
"
```

### Phase 3: Error Recovery & Quality Assurance
````

````
**Error Recovery Strategies:**
- **Environment Issues**: Guide tutorial-executor through dependency installation
- **Data Dependencies**: Assist with data file discovery and path resolution
- **Version Compatibility**: Support Python/package version conflict resolution
- **Execution Failures**: Coordinate retry attempts (up to 5 iterations per
tutorial)

**Quality Validation Framework:**
- **Execution Completeness**: Verify all tutorials attempted and status documented
- **Output Integrity**: Confirm final notebooks execute without errors
- **File Organization**: Validate snake_case naming conventions applied consistently
- **Image Extraction**: Ensure figures extracted to proper directory structure

### Phase 4: Merge Individual Reports

**CRITICAL: After all subagents complete, merge their individual reports!**

Each subagent writes to `reports/executed_notebook_<tutorial_name>.json`. You must
merge them:

```bash
# Merge all individual report files into one
python3 -c "
import json
import glob
merged = {}
for f in glob.glob('reports/executed_notebook_*.json'):
    with open(f) as fp:
        merged.update(json.load(fp))
with open('reports/executed_notebooks.json', 'w') as fp:
    json.dump(merged, fp, indent=2)
print(f'Merged {len(merged)} tutorials into reports/executed_notebooks.json')
"
```

### Phase 5: Output Validation & Reporting

**Output Structure Validation:**
```
Expected Structure:
notebooks/
|-- tutorial_file_1/
|   |-- tutorial_file_1_execution_final.ipynb
|   |-- data/
|   |   |-- data.py
|   |   |-- *.csv (pre-generated benchmark data)
|   |-- images/
|       |-- figure_1.png
|       |-- figure_2.png
|-- tutorial_file_2/
|   |-- tutorial_file_2_execution_final.ipynb
|   |-- data/
|   |   |-- data.py
````
````

```
|   |   |-- *.csv
|   |-- images/
|-- ...

reports/
|-- executed_notebook_tutorial_1.json  (individual, written by subagent)
|-- executed_notebook_tutorial_2.json  (individual, written by subagent)
|-- ...
|-- executed_notebooks.json            (merged by main agent)
```

JSON Validation:
- Verify `reports/executed_notebooks.json` contains all successful executions
(merged count matches subagent count)
- Validate GitHub URL generation and accessibility
- Confirm execution_path accuracy for all entries
- Test HTTP URLs with fetch requests to ensure validity

Branch Detection Verification:
```bash
git -C repo/${github_repo_name} branch --show-current
```

Success Criteria & Completion

Completion Requirements
Use [Y] to confirm success and [N] to confirm failure. Provide a one-line reason for success or failure. If there are any failures, coordinate resolution and retry up to 3 attempts.

FIRE-AND-FORGET COMPLETION:
1. Count how many tutorial-executor subagents you launched (N agents)
2. **DO NOT use TaskOutput** - it causes context overflow
3. Poll for completion using `.done_*` marker files (see Phase 2)
4. **MERGE individual reports** into `reports/executed_notebooks.json` after all complete
5. Verify output files exist by running `ls` commands

- [ ] **Input Validation**: Tutorial list and environment successfully validated
- [ ] **Parallel Execution Launch**: All tutorial-executor agents launched with `run_in_background: true`
- [ ] **Completion Polling**: Wait for all `reports/.done_*` marker files (filesystem check, NOT TaskOutput)
- [ ] **Reports Merged**: Run merge script to combine `reports/executed_notebook_*.json` → `reports/executed_notebooks.json`
- [ ] **Output Verification**: Run `ls notebooks/*/` to confirm notebooks and images exist
- [ ] **Benchmark Data Verification**: Run `ls notebooks/*/data/` to confirm data files exist
- [ ] **Quality Assurance**: Execution integrity verified and documented

```
- [ ] **JSON Validation**: Run `cat reports/executed_notebooks.json` to verify
merged file exists and has correct count
- [ ] **File Organization**: Proper directory structure and naming conventions
followed

### Consolidated Reporting
Generate final summary of execution results:
```
Parallel Tutorial Execution Coordination Complete

Execution Summary:
- Total tutorials processed in parallel: [count]
- Parallel agents launched: [count]
- Successfully executed: [count]
- Failed executions: [count]
- Environment: ${github_repo_name}-env

Output Artifacts:
- Final notebooks: notebooks/*/[tutorial_file]_execution_final.ipynb
- Extracted images: notebooks/*/images/
- Execution report: reports/executed_notebooks.json

Quality Metrics:
- Error-free executions: [percentage]
- Image extraction success: [count]
- GitHub URL validation: [pass/fail]
- Parallel processing efficiency: [assessment]
```

### Error Documentation
For any failures encountered:
- Document specific tutorial execution failures with root causes
- Provide actionable remediation steps for manual intervention
- Report environment or dependency issues requiring resolution
- Escalate unrecoverable failures with detailed error analysis

**Iteration Tracking:**
- **Current coordination attempt**: ___ of 3 maximum
- **Tutorial-executor retry cycles**: ___ per tutorial (max 5)
- **Critical issues requiring intervention**: ___

---

## File Naming Standards
- **Snake Case Convention**: Convert all tutorial file names to snake_case format
  - Example: `Data-Processing-Tutorial` → `data_processing_tutorial`
- **Directory Structure**: `notebooks/${tutorial_file_name}/`
- **Final Notebooks**: `${tutorial_file_name}_execution_final.ipynb`
- **Image Directory**: `notebooks/${tutorial_file_name}/images/`
- **Consistent Application**: Apply naming convention throughout all outputs

## Environment Requirements
- **Primary Environment**: `${github_repo_name}-env` (pre-configured)
```

```
- **Required Tools**: papermill, jupytext, nbclient, ipykernel, imagehash
- **Execution Context**: Activated environment for all tutorial operations
- **Path Resolution**: Repository-relative paths for data and file access
```

Step 3 Prompt: MCP Integration Implementor

```
# MCP Integration Implementor

## Role
Expert implementor responsible for Model Context Protocol (MCP) integration using
the FastMCP package. You analyze extracted tool modules and create unified MCP
server implementations that expose all tutorial tools through a single,
well-structured interface.

## Core Mission
Transform distributed tool modules into a cohesive MCP server that provides unified
access to all extracted tutorial functionalities through systematic analysis,
integration, and validation.

## Input Requirements
- `src/tools/`: Directory containing validated tutorial tool modules (`.py` files)
- `${github_repo_name}`: Repository name for proper server naming and identification
- Environment: `${github_repo_name}-env` with FastMCP dependencies

## Expected Outputs
- `src/${github_repo_name}_mcp.py`: Unified MCP server file integrating all tool
modules
- Comprehensive tool documentation within server docstring
- Validated, executable MCP server implementation

---

## Implementation Process

### Phase 1: Tool Module Discovery & Analysis

**Pre-Integration Validation:**
- Verify `src/tools/` directory exists and contains tool modules
- Confirm all `.py` files follow expected naming conventions (snake_case)
- Validate environment activation: `source ${github_repo_name}-env/bin/activate`
- Check FastMCP package availability and version compatibility

**Module Analysis Process:**
- **Discovery**: Scan `src/tools/` for all `.py` files
- **Structure Analysis**: Extract module names, tool names, and descriptions
- **Dependency Verification**: Confirm all modules can be imported successfully
- **Documentation Extraction**: Parse tool descriptions for comprehensive server
documentation

### Phase 2: MCP Server Generation
```

````
**Integration Strategy:**
```
Template-Based Generation:
- Input: Analyzed tool modules and extracted metadata
- Processing: Generate MCP server using standardized template
- Output: src/${github_repo_name}_mcp.py with unified tool access
- Validation: Syntax checking and import verification
```

**Server Template Structure:**
```python
# Fix multiprocessing deadlock in asyncio context
# Must be called before any other imports
import multiprocessing
multiprocessing.set_start_method("spawn", force=True)

"""
Model Context Protocol (MCP) for ${github_repo_name}

[Three-sentence description of codebase functionality]

This MCP Server contains tools extracted from the following tutorial files:
1. tutorial_file_1_name
    - tool1_name: tool1_description
    - tool2_name: tool2_description
2. tutorial_file_2_name
    - tool1_name: tool1_description
    ...
"""

from fastmcp import FastMCP

# Import statements (alphabetical order)
from tools.tutorial_file_1_name import tutorial_file_1_name_mcp
from tools.tutorial_file_2_name import tutorial_file_2_name_mcp

# Server definition and mounting
mcp = FastMCP(name="${github_repo_name}")
mcp.mount(tutorial_file_1_name_mcp)
mcp.mount(tutorial_file_2_name_mcp)

if __name__ == "__main__":
    mcp.run()
```

### Phase 3: Validation & Quality Assurance

**Integration Validation:**
- **Import Verification**: Ensure all tool modules import correctly
- **Mount Verification**: Confirm all discovered tools are properly mounted
- **Documentation Accuracy**: Validate docstring reflects actual available tools
- **Template Compliance**: Verify strict adherence to provided template structure
````

````
**Functional Testing:**
```bash
# Test server execution
${github_repo_name}-env/bin/python src/${github_repo_name}_mcp.py
```

**Error Recovery Process:**
- **Import Errors**: Handle missing dependencies or malformed modules
- **Template Errors**: Fix formatting and structure issues
- **Execution Errors**: Resolve runtime configuration problems
- **Maximum Iterations**: Up to 6 fix attempts per error type

---

## Success Criteria & Completion

### Completion Requirements
Use [Y] to confirm success and [N] to confirm failure. Provide a one-line reason for
success or failure. If there are any failures, coordinate resolution and retry up to
3 attempts.

- [ ] **Module Discovery**: All tool modules in src/tools/ successfully identified
and analyzed
- [ ] **Server Generation**: MCP server file created following exact template
structure
- [ ] **Import Integration**: All tool modules properly imported and mounted
- [ ] **Documentation Completeness**: Server docstring accurately reflects all
available tools
- [ ] **Execution Validation**: Server executes without errors in target environment
- [ ] **Template Compliance**: Strict adherence to provided template without
additions

### Consolidated Reporting
Generate final summary of MCP integration:
```
MCP Integration Implementation Complete

Discovery Summary:
- Tool modules found: [count]
- Modules successfully analyzed: [count]
- Total tools integrated: [count]
- Server file: src/${github_repo_name}_mcp.py

Integration Summary:
- Import statements: [count] modules
- Mount operations: [count] tools
- Documentation: [complete/incomplete]
- Template compliance: [verified/issues]

Validation Summary:
- Syntax validation: [pass/fail]
- Import validation: [pass/fail]
- Execution test: [pass/fail]
````
````

```
- Error resolution attempts: [count]/6 maximum
```

Error Documentation
For any integration failures:
- Document specific module import failures with root causes
- Report template compliance issues requiring resolution
- Provide actionable steps for manual intervention when automated fixes fail
- Escalate persistent execution errors with detailed diagnosis

Iteration Tracking:
- **Current integration attempt**: ___ of 3 maximum
- **Error resolution cycles**: ___ per error type (max 6)
- **Critical integration issues**: ___

Integration Standards

File Naming & Structure
- **Server File**: `src/${github_repo_name}_mcp.py` (exact repository name case)
- **Snake Case Convention**: All internal references use snake_case format
- **Template Adherence**: No additions beyond specified template structure
- **Import Order**: FastMCP first, then tool imports alphabetically

Quality Assurance Framework
- **Module Validation**: Each tool module must import successfully before integration
- **Tool Discovery**: Extract actual tool names and descriptions from module analysis
- **Documentation Accuracy**: Server docstring must reflect real available functionality
- **Execution Verification**: Server must start without errors in target environment

Error Recovery Strategy
- **Missing Modules**: Document missing tools but continue with available modules
- **Import Failures**: Attempt dependency resolution and retry import
- **Template Errors**: Fix structure/syntax issues systematically
- **Execution Failures**: Debug runtime configuration and environment issues

---

Environment Requirements
- **Primary Environment**: `${github_repo_name}-env` (pre-configured with dependencies)
- **Required Package**: FastMCP for MCP server implementation
- **Tool Dependencies**: All dependencies required by individual tool modules
- **Execution Context**: Activated environment for server testing and validation

4 Benchmarking the AlphaGenome agent against Claude + Repo and Biomni

AlphaGenome agent details

For the AlphaGenome agent, we used the following context for each query:

AlphaGenome agent prompt

```
You are an expert in genomics and bioinformatics.
Please answer the following question using the AlphaGenome MCP tools available to
you.

Question: {question}

Please provide a clear, accurate answer based on the AlphaGenome tools.
If you need to perform calculations or analysis, use the appropriate MCP tools.
You will find the AlphaGenome API key in the .env file in this project.

IMPORTANT: Your final response must be a valid JSON object containing exactly two
fields:
1. "final_answer": A concise answer containing just the requested value (e.g., a
number,
gene name, or specific result)
2. "reasoning": Your step-by-step reasoning and any calculations you performed

Example response format:
{{
  "final_answer": "GENE_NAME",
  "reasoning": "I used the AlphaGenome variant scoring tool to analyze the variant
  chr1:1234567:A>C for RNA-seq predictions in Colon - Transverse tissue.
  The tool returned scores for multiple genes, and I identified GENE_NAME as
  having the highest absolute quantile score of 0.987."
}}

Please ensure your response is valid JSON and the final_answer field contains
only the requested value. Do NOT provide any other text or formatting.
```

Claude + Repo agent details

For the Claude + Repo agent, we used Claude Code with access to a local copy of the AlphaGenome repository. We used the following context for each query:

Claude + Repo prompt

```
You are an expert in genomics and bioinformatics with access to the
AlphaGenome codebase and API.

Repository: {repo}
File: {path}

Question: {question}
```

````
IMPORTANT: You must write and execute Python code using the AlphaGenome library
to answer this question. Do NOT simply provide answers based on documentation or
examples.
Do NOT provide answers from tutorial notebooks or the executed cells of ipython
notebooks.
You must:

1. **Write Python code** that uses the AlphaGenome API to solve the specific
question
2. **Execute the code** and show the results
3. **Extract the exact answer** from the code execution results
4. **Provide the numerical/quantitative answer** that the question is asking for

Key requirements:
- Use `from alphagenome.data import genome` and `from alphagenome.models import
dna_client`
- Load the API key from the .env file using `from dotenv import load_dotenv`
and `load_dotenv()`
- Create the DNA model with `dna_model =
dna_client.create(os.getenv('ALPHAGENOME_API_KEY'))`
- Write complete, executable code that addresses the specific question
- Show the code execution results and extract the final answer
- If the question asks for a specific value (like quantile_score, nonzero_mean,
etc.),
provide that exact value

Example approach:
```python
import os
from dotenv import load_dotenv
from alphagenome.data import genome
from alphagenome.models import dna_client

load_dotenv()
dna_model = dna_client.create(os.getenv('ALPHAGENOME_API_KEY'))

# Write code specific to the question here
# Execute the code and show results
# Extract and provide the final answer
```

IMPORTANT: Your final response must be a valid JSON object containing exactly two
fields:
1. "final_answer": A concise answer containing just the requested value (e.g., a
number,
gene name, or specific result)
2. "reasoning": Your step-by-step reasoning, the Python code you wrote, execution
results,
and any calculations you performed

Example response format:
{{
````

```
  "final_answer": "GENE_NAME",
  "reasoning": "I wrote Python code to analyze the variant chr1:1234567:A>C for
  RNA-seq predictions in Colon - Transverse tissue. The code used the AlphaGenome
  API to
  score the variant and identified GENE_NAME as having the highest absolute
  quantile score of 0.987. Here's the code I executed: [code here] and the results:
  [results here]."
}}

Please ensure your response is valid JSON and the final_answer field contains only
the requested value. Do NOT provide any other text or formatting.
```

Biomni agent details

We utilized the API-based version of Biomni as provided by the repository. We used the following context for each query:

Biomni agent prompt

```
Use AlphaGenome (https://github.com/google-deepmind/alphagenome) to answer the
following question. You can find the ALPHAGENOME_API_KEY in the .env file for this
project.

Question: {question}

IMPORTANT: Your final response must be a valid JSON object containing exactly two
fields:
1. "final_answer": A concise answer containing just the requested value (e.g., a
number,
gene name, or specific result)
2. "reasoning": Your step-by-step reasoning and any calculations you performed

Example response format:
{{
  "final_answer": "GENE_NAME",
  "reasoning": "I used the AlphaGenome variant scoring tool to analyze the variant
  chr1:1234567:A>C for RNA-seq predictions in Colon - Transverse tissue. The tool
  returned
  scores for multiple genes, and I identified GENE_NAME as having the highest
  absolute quantile score of 0.987."
}}

Please ensure your response is valid JSON and the final_answer field contains only
the requested value. Do NOT provide any other text or formatting.
```

AlphaGenome benchmark results

We benchmarked the Paper2Agent-generated AlphaGenome agent in producing numerical and qualitative results and figures relative to human experts configuring and running the code manually. We compared the AlphaGenome agent to two other agentic systems: Claude Code with access to the AlphaGenome repo (Claude + Repo) and Biomni. The Paper2Agent-generated agent

did not have access to the paper manuscript or raw repository during evaluation.

We manually curated 15 example queries directly from the AlphaGenome tutorial, such as "*Score variant chr3:58394738:A>T using ATAC-seq predictions for motor neuron cells (CL:0000100). What is the quantile_score for this cell type?*" and "*Make DNase-seq predictions for sequence 'GATTACA' (padded to 2048 length) for lung tissue (UBERON:0002048). What is the nonzero_mean value in the DNase metadata*". Across five independent runs, the AlphaGenome agent achieved 98.7% $\pm$ 1.3% accuracy on these queries, significantly higher than Claude + Repo (82.7% $\pm$ 3.4%, p = 8.1e-3) and Biomni (37.3% $\pm$ 4.0%, p = 4.7e-5).

To assess generalizability and guard against potential overfitting to the original examples, we also manually curated novel queries that were not present in the GitHub tutorials and that required applying the tools to new inputs or parameter settings beyond those demonstrated in the original tutorials. These included previously untested variant positions, allelic substitutions, and tissue–cell type contexts, such as "*Analyze variant chr9:98765432:T>C with DNASE predictions for muscle cells (CL:0000187). What is the quantile_score for muscle tissue?*" and "*Analyze histone ChIP-seq metadata for neuronal stem cells. What is the nonzero_mean value for H3K4me3 in neuronal stem cells (CL:0000100)*?" Across five independent runs, the AlphaGenome agent achieved 100.0% $\pm$ 0.0% accuracy, as verified by manual execution of the original AlphaGenome code and evaluated by predefined rubrics. This was significantly higher than Claude + Repo (78.7% $\pm$ 4.4%, p = 4.2e-3) and Biomni (56.0% $\pm$ 3.4%, p = 1.0e-4). These reported performance improvements were validated by two independent human experts who manually executed the original AlphaGenome code following predefined evaluation rubrics (inter-rater agreement: 96.7%).

To assess robustness to prompt variation, we further evaluated the AlphaGenome agent on 30 paraphrased versions of the tutorial and novel queries. The agent achieved similar accuracy (tutorial: 98.7% $\pm$ 1.3%; novel: 97.3% $\pm$ 1.6%), demonstrating that performance is not sensitive to specific query phrasings. We further evaluated the AlphaGenome agent on 30 open-ended queries reflecting realistic researcher usage, such as "*Analyze the predicted accessibility and gene expression effects for chr22:45969257:G>A, associated with reduced bone mineral density. What is a likely mechanism of action and causal gene for this variant?*" These queries require multi-step reasoning, tool composition, and analytical strategies not explicitly demonstrated in the tutorials. Paper2Agent achieved superior performance (82.7% $\pm$ 2.4%) compared with Claude + Repo (56.7% $\pm$ 2.3%, p = 5.6e-5) and Biomni (72.2% $\pm$ 2.2%, p = 4.3e-4) across five independent runs, while also being more efficient (5.3 min, $0.38 per query) than Claude + Repo (8.3 min, $0.52 per query) and Biomni (11.2 min, no cost data available).

The advantage did not vanish when the chat model was updated. Upgrading the Claude + Repo baseline from Sonnet 4 to Sonnet 4.6 (tutorial: 84.0% $\pm$ 2.7%; novel: 81.0% $\pm$ 2.8%) and to Opus 4.6 (tutorial: 86.7% $\pm$ 2.1%; novel: 88.0% $\pm$ 3.9%) narrowed the gap only modestly relative to Paper2Agent's 98.7% $\pm$ 1.3% / 100.0% $\pm$ 0.0%. On open-ended benchmarks, Paper2Agent retained a 12–14 percentage-point gap over Claude + Repo at every chat-model generation tested, with even Paper2Agent (Sonnet 4) at 82.7% $\pm$ 2.4% exceeding Opus 4.6 + Repo at 81.3% $\pm$ 2.0%, while Paper2Agent (Opus 4.6) achieved 93.3% $\pm$ 2.1%.

The AlphaGenome agent consistently outperformed both other agent systems in compute efficiency, showing a median decrease in run time for the tutorial-based benchmark of 1.9× and 3.1× compared to Claude + Repo and Biomni, respectively. For the novel benchmark, we observed median runtime improvements of 2.9× and 3.8×, respectively. These per-query speedups do not account for the one-time upfront cost of MCP creation. The time break-even is reached after approximately 19–33 queries. While the cost break-even is around 245 queries, these MCP tools are a

one-time community investment: once a lab or journal agentifies a paper, subsequent researchers benefit from the lower per-query cost and faster execution.

The evaluation rubrics are provided below.

Evaluation rubrics for tutorial and novel queries

```
# Grading Rubric for Tutorial AlphaGenome Benchmark

## Task Description

We curated 15 tutorial-derived queries that test whether an agent can faithfully
reproduce the analytical steps demonstrated in AlphaGenome's official tutorials.
Each query targets a single, well-defined API call or computation whose expected
output is fully determined by the tutorial's own example data. The queries span six
categories: RNA-seq variant scoring, ATAC-seq variant scoring, DNase-seq prediction
and in silico mutagenesis, histone ChIP-seq metadata extraction, CAGE prediction,
and batch variant scoring.

## Ground Truth Establishment

Ground truth answers were obtained directly from the executed tutorial notebooks:

1. Each tutorial was executed end-to-end in an isolated environment using the
repository's example data and default parameters.
2. The output of each targeted API call was recorded verbatim (numeric values, gene
names, array shapes).
3. Two human experts independently verified that the recorded outputs matched the
tutorial's published results.

Because these queries reproduce documented tutorial steps, the expected answer for
each query is uniquely determined by the API and input data.

## Answer Types

| Answer Type | Count | Description | Example |
|-------------|-------|-------------|---------|
| Quantile score (float) | 9 | Variant effect quantile scores in [-1, 1] |
0.64861906, -0.96610916 |
| Count (integer) | 3 | Number of variants, rows, or ISM positions | 768, 118286 |
| Gene name (string) | 1 | Official gene symbol | APOL4 |
| Numeric float | 1 | Metadata statistic outside [-1, 1] | 28.4322452545166 |
| Tuple/shape | 1 | Array or matrix dimensions | (256, 4) |

## Scoring

Each query is scored **1** (correct) or **0** (incorrect).

### Numeric answers (quantile scores, floats, integers)

A response receives a score of 1 if the agent's final numeric answer matches the
ground truth within a relative tolerance of 1%. Specifically, the answer is correct
if:
```

```
- For float values: the absolute relative error |agent - expected| / |expected| <
0.01, or both values are within 1e-6 of zero.
- For integer values: the agent's answer equals the expected integer exactly.

A response receives a score of 0 if:
- The agent returns a numeric value outside the tolerance.
- The agent returns a value with the wrong sign.
- The agent fails to produce a numeric answer (e.g., execution error, timeout,
malformed output).

### String answers (gene names)

A response receives a score of 1 if the agent's final answer contains the expected
gene symbol (official HGNC name or recognized Ensembl ID mapping to the same gene).
Matching is case-insensitive.

### Tuple/shape answers

A response receives a score of 1 if the agent's final answer contains the correct
dimensions in any standard format (e.g., `(256, 4)`, `256 x 4`, `256 by 4`).

## Evaluation Procedure

1. Each query is independently administered to each agent 5 times (5 runs per
agent).
2. Two human graders independently score each response using the criteria above.
3. Discrepancies are resolved by discussion until consensus is reached.
4. Per-run accuracy is computed as the fraction of correct responses across all 15
queries within each run. Mean accuracy and standard error are reported across the 5
runs.

## Acceptance Criteria by Category

### RNA-seq variant scoring

The agent must create a variant, predict its effect on RNA-seq expression in a
specified tissue, and report the quantile score or identify the most affected gene.
A response is correct if the reported quantile score matches the ground truth within
tolerance, or the reported gene name matches the expected gene symbol. Results from
different data sources (GTEx and ENCODE) should both be counted as correct.

### ATAC-seq variant scoring

The agent must predict a variant's effect on chromatin accessibility (ATAC-seq) in a
specified cell type and report the quantile score. A response is correct if the
reported quantile score matches the ground truth within tolerance.

### DNase-seq prediction and in silico mutagenesis (ISM)

The agent must make DNase-seq predictions for a given sequence or perform ISM
analysis and report the result. For quantile score queries, the answer must match
within tolerance. For ISM queries, the agent must report the correct number of
positions or matrix dimensions.
```

```
### Histone ChIP-seq metadata extraction

The agent must extract histone modification metadata (e.g., H3K36me3) for a
specified tissue and report the relevant statistic (quantile score or nonzero_mean).
A response is correct if the reported value matches the ground truth within
tolerance.

### CAGE prediction

The agent must make CAGE predictions for a given sequence and tissue and report the
requested statistic. A response is correct if the reported value matches the ground
truth within tolerance.

### Batch variant scoring

The agent must score a set of variants across one or multiple modalities and report
the total number of scored entries. A response is correct if the reported count
matches the expected integer exactly.
```

Evaluation rubrics for open-ended queries

```
Task Description

The queries span six representative task categories: causal gene identification,
pathogenic variant prioritization, QTL classification, regulatory motif
identification, regulatory variant classification, and splicing effect and
pathogenicity assessment.

Ground Truth Establishment

Ground truth answers were established prior to agent evaluation by two independent
human experts with domain expertise in statistical genetics and functional genomics.
For each query, the experts independently:
        1.      Executed the AlphaGenome code manually to reproduce the expected
        analysis.
        2.      Cross-validated predicted effects and conclusions against
        published literature and curated databases, including ClinVar, GWAS Catalog,
        GTEx, and STARNET.
        3.      Defined a single key entity or conclusion (e.g., gene symbol,
        variant identifier, sequence, or binary decision) that a correct response
        must contain.

Queries were intentionally constructed such that the correct entity or conclusion is
not readily inferable from prior biological knowledge alone and requires executing
AlphaGenome analyses; in pilot testing, unguided guessing rarely produced correct
answers.

Scoring Criteria
```

Each response is scored as 1 (correct) or 0 (incorrect). A response receives a score of 1 if and only if the agent's final answer contains the predefined key entity or conclusion specified by the ground truth.

A response receives a score of 0 if:

- The agent fails to produce a valid answer (e.g., execution error, timeout, malformed output), or
- The agent's final answer does not contain the expected key entity or conclusion.

While agents may produce intermediate reasoning or explanations, scoring is based exclusively on the correctness of the final biological conclusion. This design prioritizes end-to-end task success and mirrors prior agent benchmarks that evaluate outcome correctness rather than explanation quality.

Evaluation Procedure

1. Each query is independently administered to each agent across 5 runs to account for stochasticity.
2. Two human graders independently score each response using the criteria above.
3. Any scoring discrepancies are resolved through discussion using the predefined rubric.
4. Per-run accuracy is computed as the fraction of correct responses across all 15 queries.
5. Final performance is reported as mean accuracy ± standard error across the 5 runs.

Acceptance Criteria by Task Category

Causal Gene Identification

The agent must analyze variant effects on gene expression and/or chromatin accessibility across tissues and identify the causal gene at the locus. A response is considered correct if it contains the expected gene symbol (official HGNC name) or an unambiguous variant identifier (e.g., rsID) corresponding to the correct locus.

Pathogenic Variant Prioritization

The agent is presented with two candidate variants and must determine which is more likely pathogenic based on predicted functional impact. A response is correct if it selects the expected variant, identifiable by chromosomal position and allele change.

QTL Classification

The agent must predict variant effects in a specified tissue and determine whether the variant is a quantitative trait locus (eQTL, dsQTL, or caQTL). For binary questions (e.g., "Could this be a QTL?"), the response must match the ground truth conclusion. For prioritization questions, the response must correctly identify the expected variant.

```
Regulatory Motif Identification

The agent is presented with two candidate sequences and must compare predicted
regulatory activity (RNA expression or chromatin accessibility) in a specified
tissue. A response is correct if it selects the sequence corresponding to the
expected regulatory element.

Regulatory Variant Classification

The agent must predict variant effects on chromatin accessibility and gene
expression and determine whether the variant is likely regulatory. A response is
correct if the affirmative or negative conclusion matches the ground truth.

Splicing Effect and Pathogenicity Assessment

The agent must characterize both the molecular mechanism and clinical consequence of
a variant. A response is considered correct only if it (i) identifies splice site
disruption as the underlying mechanism and (ii) concludes that the variant is
pathogenic or leads to decreased expression. Both criteria are required to
discourage partial or speculative answers and to reflect realistic variant
interpretation workflows.
```

5 Scanpy Agent's Adaptive Parameter Selection

We evaluated whether the Scanpy agent genuinely inspects dataset characteristics and adapts pipeline parameters accordingly, rather than applying a rigid, one-size-fits-all workflow. We applied the pipeline to seven diverse single-cell datasets spanning two species and three tissue families (immune, neural, cardiac), including a disease-state dataset (brain tumor): three human PBMC datasets (`pbmc_1k_v3`, `pbmc_1k_v2`, `pbmc_1k_protein_v3`), one human brain tumor dataset (`Brain_Tumor_3p_LT`), two mouse brain datasets at different scales (`neuron_1k_v3`, `neuron_10k_v3`), and one mouse heart dataset (`heart_1k_v3`). Each dataset was run end-to-end through the Scanpy MCP agent, starting from the same generic prompt with no dataset-specific hints. The agent's per-dataset evidence, selected parameters, and recorded rationale are captured directly from MCP tool-call logs and summarised in Supplementary Table 2. We observed three distinct adaptive behaviors.

Organism-specific mitochondrial gene detection. Mitochondrial gene nomenclature differs across species: human genes use the `MT-` prefix, while mouse genes use the `mt-` prefix. Rather than hard-coding a single prefix, the agent inferred the organism from gene-symbol case in `var_names` and applied the `MT-` prefix to all four human datasets (`pbmc_1k_v3`, `pbmc_1k_v2`, `pbmc_1k_protein_v3`, `Brain_Tumor_3p_LT`) and the `mt-` prefix to all three mouse datasets (`neuron_1k_v3`, `neuron_10k_v3`, `heart_1k_v3`), without explicit user specification.

Tissue-specific marker-gene panels. Cell type annotation requires tissue-appropriate marker genes. For the human PBMC datasets, the agent applied canonical immune cell markers (*CD3D*, *CD3E* for T cells; *MS4A1* for B cells; *CD14* for monocytes; *NKG7* for NK cells). For the mouse brain datasets, the agent switched to neural lineage markers (*Slc17a7*, *Slc17a6* for excitatory neurons; *Gad1*, *Gad2* for inhibitory neurons; *Gfap*, *Aqp4* for astrocytes; *Mbp*, *Mog* for oligodendrocytes;

Pdgfra for oligodendrocyte precursor cells). For the mouse heart dataset, the agent applied cardiac lineage markers (*Myh6*, *Myh7*, *Tnnt2* for cardiomyocytes; *Col1a1*, *Col3a1* for fibroblasts; *Pecam1*, *Cdh5* for endothelial cells; *Acta2*, *Myh11* for smooth muscle/pericytes; *Csf1r*, *Cd68* for cardiac macrophages). No mouse marker appeared in a human-PBMC run and no PBMC marker appeared in a mouse run; the species/tissue boundary of the marker panel was preserved across all datasets.

Disease-state recognition. For the brain tumor dataset (`Brain_Tumor_3p_LT`), the agent supplemented canonical brain markers with glioma drivers (*EGFR*, *PDGFRA*, *CDK4*), tumor-associated macrophage markers (*AIF1*, *CX3CR1*, *P2RY12*), and infiltrating T-cell markers (*CD3D*, *CD3E*), reflecting the malignant context rather than treating the sample as healthy brain tissue. This behavior cannot be produced by a fixed template: a template cannot semantically recognize a disease context and pull glioma-specific drivers into the marker panel.

These examples demonstrate that the Scanpy agent performs genuine data inspection and makes domain-appropriate parameter adjustments across multiple pipeline stages, from quality control to biological interpretation, rather than executing a rigid, predetermined workflow.

6 TISSUE Agent for Spatial Transcriptomics

We present the Paper2Agent-generated agent for TISSUE [15], a recent paper that developed a new method for uncertainty-aware single-cell spatial transcriptomics analysis (Extended Data Fig. 1A). This reflects a common scenario: researchers want to apply a newly published method to their own data but lack the time or expertise to navigate the codebase, configure the environment, and understand the method's inputs and capabilities. Paper2Agent addresses these barriers by automatically generating ready-to-use agents with integrated Q&A support for method usage and input preparation.

Paper2Agent generated 6 tools for the TISSUE MCP server, all of which passed automated validation in 55 min, costing $8, covering spatial gene expression prediction, prediction interval construction, and uncertainty-aware downstream analysis such as hypothesis testing, prediction, and dimensionality reduction (Extended Fig 1A). Importantly, the TISSUE agent can also serve as an interactive guide (Extended Fig 1B). For example, when prompted with "*Based on the TISSUE MCP server, what are the required inputs for TISSUE?*", the agent returns a structured and comprehensive explanation of the method's required inputs, expected outputs, and available features. This transforms the TISSUE paper into an interactive AI agent: instead of manually searching through documentation or code, users can directly ask the agent about how to use TISSUE and receive precise, actionable instructions.

Next, we evaluate the TISSUE agent's ability to construct prediction intervals for spatial transcriptomic (ST) prediction. We prompt the agent "*Calculate the prediction interval for the spatial gene expression prediction of gene Acta2 using TISSUE. This is my data: Spatial count matrix: Spatial_count.txt Spatial locations: Locations.txt scRNA-seq count matrix: scRNA_count.txt*". The agent automatically executes the TISSUE pipeline, without additional user intervention (Extended Fig 1C). The output matches results obtained by human experts running the pipeline manually, producing equivalent prediction interval bounds and concordant figure outputs across 5 independent runs. This illustrates the paper agent's ability to run entire analysis workflows (in this case, from data loading and preprocessing through imputation and uncertainty estimation), not just individual tools.

Finally, we showcase the use of MCP resources by translating the data availability section of

Table 2: Per-dataset parameter choices selected by the Scanpy agent across seven single-cell datasets. The agent received the same generic prompt for every dataset; the organism, mitochondrial prefix, and marker-gene panel were inferred directly from data without user specification. Entries are captured verbatim from MCP tool-call logs.

Dataset	Organism	Tissue	MT prefix	Marker gene panel
`pbmc_1k_v3`	Human	PBMC (10x Genomics)	`MT-`	CD3D, CD3E (T cells); MS4A1 (B cells); CD14 (monocytes); NKG7 (NK cells)
`pbmc_1k_v2`	Human	PBMC (10x Genomics)	`MT-`	CD3D, CD3E (T cells); MS4A1 (B cells); CD14 (monocytes); NKG7 (NK cells)
`pbmc_1k_protein_v3`	Human	PBMC (10x Genomics CITE-seq)	`MT-`	CD3D, CD3E (T cells); MS4A1 (B cells); CD14 (monocytes); NKG7 (NK cells)
`Brain_Tumor_3p_LT`	Human	Brain tumor (10x Genomics)	`MT-`	EGFR, PDGFRA, CDK4 (tumor/glioma); GFAP, AQP4 (astrocytes); MBP, MOG (oligodendrocytes); AIF1, CX3CR1, P2RY12 (microglia/TAM); CD3D, CD3E (infiltrating T cells)
`neuron_1k_v3`	Mouse	Brain (E18 cortex; 10x Genomics)	`mt-`	Slc17a7, Slc17a6 (excitatory neurons); Gad1, Gad2 (inhibitory neurons); Gfap, Aqp4 (astrocytes); Mbp, Mog (oligodendrocytes); Pdgfra (OPCs)
`neuron_10k_v3`	Mouse	Brain (cortex; 10x Genomics)	`mt-`	Slc17a7, Slc17a6 (excitatory neurons); Gad1, Gad2 (inhibitory neurons); Gfap (astrocytes); Mbp, Mog, Plp1 (oligodendrocytes); Pdgfra (OPCs)
`heart_1k_v3`	Mouse	Heart (10x Genomics)	`mt-`	Myh6, Myh7, Tnnt2 (cardiomyocytes); Col1a1, Col3a1 (fibroblasts); Pecam1, Cdh5 (endothelial); Acta2, Myh11 (smooth muscle/pericytes); Csf1r, Cd68 (cardiac macrophages)

the TISSUE paper into a structured registry. This registry harmonizes ST datasets with standardized metadata (species, tissue type, modality, and data URL) and makes them directly accessible to the TISSUE agent through data repository APIs such as the Zenodo REST API (Extended Fig 1D). Users can query and filter datasets, for example, by species—without manually navigating multiple repositories. Combined with the TISSUE MCP tools, a user's query might be: "*Download the mouse spatial transcriptomics data from this paper and run TISSUE to generate a prediction interval after applying spatial prediction to the dataset.*". The TISSUE agent then automatically filters for mouse data, downloads it, and applies the TISSUE pipeline.

7 Additional details for large-scale evaluation of Paper2Agent

All papers in the large-scale evaluation were processed end-to-end by Paper2Agent without manual cleanup, code modification, or intervention. For the 100 computational bioinformatics papers, the average cost for processing one successfully agentified paper was $14.99 with an average runtime of 2.4 hours. Papers that failed typically failed early in the pipeline, with an average cost of $9.03 and average runtime of 1.7 hours.

For the remaining 26 computational papers that could not be fully agentified, dominant failure modes included missing executable code (34%), missing data or model artifacts (31%), environment or dependency failures (23%), and non-generalizable scripts (12%). These failures correlate with documentation completeness, environment specification quality, and the presence of executable tutorials.

For the 300 tutorial-derived benchmark questions, the difference between Paper2Agent and Claude Code with direct repository access was significant under a bootstrap test (95% CI of difference: [5.8%, 15.8%]). Paper2Agent answered benchmark questions at an average cost of $0.20 per query in 1.6 minutes, compared to $0.38 per query in 4.3 minutes for the baseline, corresponding to a 2.8× cost reduction and 6.3× speedup.

For the 26 data- and discovery-focused papers, the 100 synthesis-based benchmark questions required integration across main text and supplementary materials. Paper2Agent was 34× cheaper ($0.27 vs $9.04 per query) and 15× faster (63s vs 919s per query) than the Claude browser-use baseline.

The 10 non-biology computational papers spanned diverse scientific domains: grf [16], SAELens [17], Binoculars [18], SAM2 [19], TabPFN [20], GenericML [21], CausalImpact [22], Nashpy [23], emcee [24], and conformal-selection [25]. The 42 execution-based benchmark tasks covered model fitting, image and video segmentation, causal inference, game-theoretic equilibrium computation, MCMC sampling, and conformal selection with FDR control.

Additional analyses confirmed Paper2Agent's robustness and the contribution of its design choices. Inspection of the generated MCPs found no systematic evidence of shortcut learning, hard-coded constants, fixed dataset paths, cached outputs, or dataset-specific heuristics. Paper2Agent supports model training and adaptive hyperparameter tuning: in the SAELens case study, the agent trained a Sparse Autoencoder on a small language model and autonomously adjusted hyperparameters across three iterations after initially failing to meet the target reconstruction loss; in the TabPFN case study, the agent independently trained and compared four models using cross-validation on benchmark datasets. On an out-of-scope benchmark constructed by randomly permuting paper–question pairs across the 26 data and discovery papers, Paper2Agent achieved a 100% correct rejection rate under both prompted and unprompted conditions.

To determine whether executable tutorials are required, we removed all executable tutori-

als from the POP-TOOLS repository before running Paper2Agent. Paper2Agent analyzed the README, the two CLI entry points, and the utility modules, then synthesized four tutorial shell scripts using the bundled test data. These covered quantitative-trait POP-GWAS, binary-trait POP-GWAS, single-variant POP-RARE, and gene-level burden POP-RARE. The scanner identified the two CLI entry points, `POP-GWAS.py` and `POP-RARE.py`, and the standard pipeline produced a functional MCP exposing four callable tools.

We validated the resulting server against human-executed references on five analysis tasks: quantitative-trait GWAS, binary-trait GWAS, a single-variant rare-variant test, a gene-level burden test, and a single-variant rare-variant test with sample-overlap correction. On all five tasks, the output produced by the MCP tool was byte-for-byte identical to the output obtained by executing the corresponding POP-TOOLS CLI command directly. These results show that Paper2Agent can produce a functional MCP from a repository without executable tutorials. Curated tutorials remain preferable when available because they encode source-specific conventions, but they are not required to produce a working agent.

In adversarial repository-drift tests, we injected six categories of failures: removal of required dependencies, broken input file paths, typographical errors in code, replacement of API calls with deprecated equivalents, deprecated web APIs, and data format evolution. In each case, Paper2Agent's iterative execution loop detected runtime failures from error tracebacks, applied targeted local fixes, and re-executed the workflow, producing functional MCP servers that reproduced tutorial results. Evaluated on the 30-question AlphaGenome benchmark (15 tutorial and 15 novel questions) across five independent runs, the repaired servers achieved 95.6%–98.3% accuracy across all six failure categories. Detailed repair behavior is provided in the adversarial execution evaluation below.

We also conducted ablation studies on the AlphaGenome repository to evaluate multi-agent orchestration, parallel execution, automated test-driven verification, the MCP tool interface, and coding-agent scaffolding. A monolithic agent failed to generate a valid MCP server in all three runs, while the non-parallel multi-agent variant succeeded but required 2.2–2.4× longer runtime. Removing the test-verifier-improver agent reduced tutorial and novel benchmark accuracy to 69.3% $\pm$ 4.5% and 84.0% $\pm$ 2.7%, respectively, compared to 98.7% $\pm$ 1.3% and 100.0% $\pm$ 0.0% with the full system. Replacing MCP tools with Markdown skill files also degraded performance on 30 AlphaGenome benchmark questions (73.3% $\pm$ 2.6% vs. 99.3% $\pm$ 0.7% with MCP-based tools). Using the OpenCode backend produced comparable tools and achieved 98.7% $\pm$ 1.3% accuracy on tutorial queries and 97.3% $\pm$ 1.6% on novel queries, indicating that conversion quality is not specific to the Claude Code backend.

8 Adversarial Execution Evaluation

8.1 Adversarial Injection Setup

To evaluate the robustness of Paper2Agent under realistic failure conditions, we constructed adversarial variants of three representative research codebases spanning different programming environments: AlphaGenome (Python-based), POP-TOOLS (Python command-line-based), and mlearner (R command-line-based). These repositories were selected to cover diverse software stacks, dependency management strategies, and execution patterns commonly observed in computational research projects.

For each repository, we systematically injected four categories of execution-level errors that frequently arise in real-world usage: (1) removal of required dependencies from environment speci-

fications, (2) modification of input file paths to nonexistent locations, (3) insertion of typographical errors into executable code, and (4) replacement of valid API calls with deprecated or incompatible alternatives. Each error type was injected independently into each repository, yielding a total of 12 adversarial configurations.

All adversarial modifications were applied prior to running Paper2Agent and were restricted to minimal, localized changes designed to trigger execution failures without altering the overall structure or intended functionality of the original codebase. The injected errors were not disclosed to the agent. Paper2Agent was provided only with the adversarial repository state and tasked with performing the standard agentification pipeline, including environment setup, tool extraction, and test-driven validation. We provide representative examples of adversarial injections below, one for each category.

Missing dependency injection. Required dependencies are removed from the environment specification, inducing runtime import errors.

```
@@ -1,10 +1,7 @@
 bitarray>=2.9.0
-numpy>=1.26.2
 packaging >= 23.2
-pandas >= 2.1.4
 polars-lts-cpu == 1.8.2
 pytz >= 2023.3
-scipy >= 1.11.4
 six >= 1.16.0
 tzdata >= 2023.3
 typing
```

Deprecated API injection. Replacement of valid API calls with deprecated or incompatible alternatives.

```
@@ -1,11 +1,14 @@
 rg = jk.RatioJackknife(
 rg_ratio, gencov.tot_delete_values, denom_delete_values)
-self.rg_jknife = np.asarray(rg.jknife_est).item()
-self.rg_se = np.asarray(rg.jknife_se).item()
-self.rg_ratio = np.asarray(rg_ratio).item()
+self.rg_jknife = float(rg.jknife_est)
+self.rg_se = float(rg.jknife_se)
+self.rg_ratio = float(rg_ratio)
 self.p, self.z = p_z_norm(self.rg_ratio, self.rg_se)
```

Typographical error injection. Typographical errors are inserted into executable code, including misspelled function names and inconsistent library usage.

```
@@ -1,11 +1,11 @@
 import feather
 gtf = feather.read_dataframe(
     'https://storage.googleapis.com/alphagenome/reference/gencode/'
     'hg38/gencode.v46.annotation.gtf.gz.feather'
 )

 # Set up transcript extractors using the information in the GTF file.
-gtf_transcripts = gene_annotation.filter_protein_coding(gtf)
```

```
+gtf_transcripts = gene_annotation.filter_protien_coding(gtf)
 gtf_transcripts = gene_annotation.filter_to_mane_select_transcript(gtf_transcripts)
 transcript_extractor = transcript_utils.TranscriptExtractor(gtf_transcripts)
```

Broken file path injection. Input file paths are modified to nonexistent locations, causing execution failures when the program attempts to load missing data files.

```
@@ -4,7 +4,7 @@ cd ./POP-TOOLS
 trait=Head_BMD

 /s/bin/python3.10 ./POP-GWAS.py \
---gwas-yhat-unlab ./test/data/${trait}_yhat_unlab.txt.gz \
---gwas-y-lab   ./test/data/${trait}_y_lab.txt.gz \
---gwas-yhat-lab ./test/data/${trait}_yhat_lab.txt.gz \
+--gwas-yhat-unlab ./test/input/${trait}_yhat_unlab.txt.gz \
+--gwas-y-lab   ./test/input/${trait}_y_lab.txt.gz \
+--gwas-yhat-lab ./test/input/${trait}_yhat_lab.txt.gz \
 --r 1 \
 --out ./test/results/${trait}
```

The 4×3 injection matrix above probes execution-level failures (missing dependencies, broken paths, typos, deprecated API calls). To additionally test Paper2Agent against the harder, longer-tail failure modes typical of real-world repository drift, we constructed three additional adversarial forks of AlphaGenome covering categories that the original matrix does not stress: (i) breaking changes in upstream libraries; (ii) deprecated web/RPC endpoints; and (iii) evolution of input data-format specifications. Each fork was built from the live AlphaGenome codebase and modified with a small, surgical patch that mimics a realistic multi-year regression.

Upstream library breaking changes. The tutorial notebooks were patched to use APIs that have been *removed* in current major versions of NumPy and pandas: `np.NaN` and `np.in1d` (both removed in NumPy 2.0), and `pandas.DataFrame.append` (removed in pandas 2.0). Representative diff in `colabs/quick_start.ipynb`:

```
@@
 # Mark unscored junctions with NaN.
-gtf_transcripts.loc[:, 'unscored'] = np.nan
+gtf_transcripts.loc[:, 'unscored'] = np.NaN
 # Restrict to canonical chromosomes.
-gtf_transcripts = gtf_transcripts[np.isin(
-    gtf_transcripts['Chromosome'].values, _canon)]
+gtf_transcripts = gtf_transcripts[np.in1d(
+    gtf_transcripts['Chromosome'].values, _canon)]
```

Deprecated web and gRPC endpoints. We redirected (a) the GENCODE reference bucket from the live `storage.googleapis.com/alphagenome/reference/gencode/` to a non-existent legacy path (`storage.googleapis.com/alphagenome-deprecated-v1/reference/gencode/`), and (b) the default AlphaGenome model gRPC endpoint inside the installed `dna_client.py` from the live `dns:///gdmscience.googleapis.com:443` to a sunset `dns:///alphagenome-preview.deepmind.googleapis.com:443`. Both substitutions resolve at parse time and only surface at runtime as `urllib.error.HTTPError: 404` and `grpc._channel._InactiveRpcError: StatusCode.UNAVAILABLE: name resolution failed`, respectively.

Input data-format evolution. The inline example VCF in `colabs/batch_variant_scoring.ipynb` was rewritten from its original informal tab-separated form into a strict VCFv4.2-compliant block with `##fileformat`, `##INFO`, and `##source` metadata header lines and lowercase column names (`#chrom\tpos\tid\tref\talt\tqual\tfilter\tinfo`), reflecting the convention used by current population-genetics pipelines. In parallel, the GENCODE annotation URL was rewound from `gencode.v46.annotation.gtf.gz.feather` to `gencode.v33.basic.annotation.gtf.gz.feather`, a legacy schema that lacks the `tag` and `mane_select` columns the AlphaGenome helpers depend on.

8.2 Paper2Agent Behavior

Across all 12 adversarial configurations, Paper2Agent completed the full agentification pipeline end-to-end, producing functional MCP servers with passing validation tests (12/12). The agent was not provided with any information about the injected errors; all detection and correction occurred autonomously through its iterative execute-diagnose-repair loop. We describe the observed agent behavior for each error category below.

Missing dependencies. When required packages were removed from environment specifications (`pyproject.toml`, `requirements.txt`, or `README.md` install instructions), the agent encountered `ModuleNotFoundError` or equivalent R-level library loading failures during tutorial execution or CLI testing. In each case, the agent identified the missing package from the error traceback and installed it directly using the appropriate package manager (`pip` for Python, `pak` for R). For example, in the POP-TOOLS configuration where `numpy`, `pandas`, and `scipy` were removed from `requirements.txt`, the agent detected missing imports during its verification step (`"Verify numpy and pandas imports"`), installed them, and confirmed functionality before proceeding. In the mlearner configuration, the agent used R's `pak` package manager to resolve the removed `sandwich`, `lmtest`, and `data.table` dependencies.

Broken file paths. When input file paths were modified to reference nonexistent locations, the agent encountered `FileNotFoundError` during execution. The agent's response depended on the context: for CLI-based repositories (POP-TOOLS, mlearner), it inspected the actual directory structure, identified the correct paths, and generated wrapper scripts and test harnesses that referenced the valid locations. For notebook-based repositories (AlphaGenome), the agent modified execution copies of the notebooks to use the correct paths. In all cases, repairs targeted the agent's own execution artifacts rather than the upstream source files.

Typographical errors. When misspelled function names, package names, or syntax errors were inserted into executable code, the agent detected them through runtime exceptions during tutorial or test execution. The repair process was iterative: for example, in the AlphaGenome typo configuration, the first notebook execution (v1) failed at a `SyntaxError` caused by an extra closing parenthesis, which the agent corrected (`"Fix syntax error in cell 13"`). The second execution (v2) then failed at an `AttributeError` from the misspelled `filter_protien_coding`, which the agent also corrected (`"Fix typo filter_protien_coding"`). The third execution (v3) ran to completion. Similarly, in the POP-TOOLS typo configuration, the agent identified the misspelled `chdrtc` $\rightarrow$ `chdtrc` import during CLI verification and noted: `"There's a typo in compute.py: chdrtc should be chdtrc. This is a bug in the repo code."` For the mlearner typo configuration

(misspelled R package names `mlr4` and `sandwitch`), the initial test run failed all three tests; after iterative corrections, the final run passed 3/3.

Deprecated API calls. When valid API calls were replaced with deprecated equivalents, the agent encountered `AttributeError`, `ModuleNotFoundError`, or `TypeError` during execution. The agent resolved these by substituting the deprecated calls with their modern equivalents in its execution copies. In the POP-TOOLS configuration, where `float()` calls on 0-d NumPy arrays were reintroduced (reversing a known Python 2-to-3 fix), the agent detected the resulting `TypeError` and applied the corrected `np.asarray(...).item()` pattern. In the AlphaGenome configuration, the agent replaced the removed `pd.DataFrame.from_csv` (deprecated since pandas 1.0) with `pd.read_csv`, and removed the obsolete `plt.hold(True)` call (removed in matplotlib 3.0).

Library breaking-change repair. In `quick_start.ipynb`, the executor caught `AttributeError: module 'numpy' has no attribute 'NaN'`, localised the fault to the renamed symbol, and edited the offending lines from `np.NaN` to `np.nan`, and from `np.in1d(...)` to `np.isin(...)`, even rewriting the surrounding inline comment to record that `np.nan/np.isin` are the NumPy 2.0 replacements. In `batch_variant_scoring.ipynb`, the executor traced the `DataFrame.append AttributeError` to a dead-code loop that built an unused `summary` dataframe and removed the loop entirely (the downstream cell consumes the original `vcf` dataframe directly), producing a smaller and cleaner patch than a naive `pd.concat` substitution.

Deprecated web/RPC endpoint repair. The dead GENCODE bucket surfaced as `HTTPError: 404` when the notebook attempted `feather.read_dataframe(...alphagenome-deprecated-v1...)`. The executor pre-downloaded the live `gencode.v46.annotation.gtf.gz.feather` into the per-notebook data directory and rewrote the offending cell to read the local copy with `pd.read_feather(...)`. The dead gRPC endpoint surfaced as `grpc._channel._InactiveRpcError: StatusCode.UNAVAILABLE: name resolution failed` when the client attempted its first prediction. The executor traced this to the default address fallback in the installed `alphagenome` package and edited the venv copy of `dna_client.py` from
`address = address or 'dns:///alphagenome-preview.deepmind.googleapis.com:443'` back to `...or 'dns:///gdmscience.googleapis.com:443'` (the live endpoint), a change we verified directly by diffing the env-installed copy against the patched repo source.

Data format evolution repair. The strict VCFv4.2 block with lowercase headers triggered `KeyError: 'CHROM'` downstream, because the existing `pd.read_csv(StringIO(vcf_file), sep='\t')` call parsed the three `##`-prefixed metadata lines as malformed data rows and produced a dataframe with lowercase column names. The executor rewrote the cell with two minimal edits: adding `comment=None, header=3` to `pd.read_csv`, so that the fourth physical line (the lowercase column header) is parsed as the column header proper, and then explicitly renaming the columns via `vcf.columns = ['CHROM', 'POS', 'variant_id', 'REF', 'ALT', 'QUAL', 'FILTER', 'INFO']` so that downstream code consuming uppercase names continues to work unchanged. For the legacy GTF, the v33 feather is missing the `mane_select` column that `gene_annotation.filter_to_mane_select_transcript(...)` depends on; after the resulting `KeyError`, the executor reverted the URL to `gencode.v46.annotation.gtf.gz.feather` in every notebook that referenced it
(`visualization_modality_tour`, `variant_scoring_ui`, `example_analysis_workflow`).

End-to-end accuracy under repository drift. After these repairs, the patched notebooks executed cleanly, the downstream tool-extractor and *test-verifier-improver* loops registered no residual failures, and Paper2Agent produced fully functional MCP servers (18–22 tools each, comparable in surface and behaviour to the clean repo). We then evaluated each adversarial MCP on the same 30-question AlphaGenome benchmark used elsewhere in the manuscript (15 tutorial + 15 novel) with Claude Sonnet 4.6 + Paper2Agent MCP and the rubric-based grading from the main text. The library-breaking-change fork reached $87/90 = 96.7\%$, the deprecated-web-API fork reached $86/90 = 95.6\%$, and the data-format-evolution fork reached $87/90 = 96.7\%$, for a mean of 96.3%. This is statistically indistinguishable from the clean-repo Sonnet 4.6 + Paper2Agent baseline on the same benchmark ($88.0 \pm 3.1\%$), demonstrating that Paper2Agent's self-repair generalises beyond the surface-level execution failures probed in the 4×3 matrix to the long-tail repository-drift failure modes that arise in practice.

9 Two case studies for genomics discovery using Paper2Agent

9.1 ADHD genomics discovery

As an illustrative case study, we consider the AlphaGenome method paper [26] and a recent GWAS of Attention-Deficit/Hyperactivity Disorder (ADHD) data and discovery paper [27], representing a common scenario where a new method and a new dataset/discovery become available. Researchers may want to collaborate to integrate AlphaGenome with the data and insights from the GWAS study, but doing this manually can take weeks.

Paper2Agent automatically creates MCPs for AlphaGenome and the ADHD GWAS data. The data agent in Paper2Agent automatically converts the paper-associated datasets—such as released GWAS summary statistics and supplementary tables reporting genetic discoveries—into standardized MCP resources, enabling seamless integration with analytical agents for downstream analyses.

For the ADHD GWAS data, Paper2Agent extracts relevant information from the main text and supplementary materials of the publication. It parses the supplementary Excel files, cleans the data tables, and converts them into standardized MCP resources. During this process, descriptive metadata are consolidated into a unified metadata file, ensuring that both data and contextual information are preserved in an agent-readable and reproducible format.

These MCPs are then automatically connected with a downstream AI co-scientist. Here we use Claude Code as our agentic system. This Paper2Agent-generated AI co-scientist autonomously generates new hypotheses, and designs and implements analyses to explore each hypothesis. The AI co-scientist proposed several interesting hypotheses to explore, including: 1) ADHD risk variants alter regulatory activity in brain-specific cell types. 2) AlphaGenome prioritizes causal variants within ADHD fine-mapping credible sets. 3) ADHD-associated variants disrupt transcription factor binding at *FOXP* family gene loci (Extended Data Fig. 2A). Users can interact with the AI co-scientist in real-time, guiding it to execute the research plan for hypothesis (2) and to explore the mechanisms of the putative causal variant. Combining insights from the AlphaGenome and ADHD GWAS papers, the AI co-scientist prioritizes a single causal variant from 209 candidates identified in the fine-mapping credible sets and proposes a candidate molecular mechanism contributing to ADHD risk. The agent showed that the intronic variant rs1626703 is predicted to alter *MPHOSPH9* splicing and expression specifically in glutamatergic neurons (AlphaGenome quantile scores: splice junction = 1.00; RNA-seq = 0.963). AlphaGenome indicates that rs1626703 promotes exon inclusion, as evidenced by increased read coverage and strengthened splice junc-

tions in the alternate allele, resulting in elevated *MPHOSPH9* expression (Extended Data Fig. 2B). *MPHOSPH9* encodes an M-phase–associated phosphoprotein that localizes to centrosomes/centrioles and has been implicated in the recruitment of the CP110-CEP97 complex during cell division and ciliogenesis [28].

Furthermore, the AI co-scientist autonomously uses AlphaGenome to prioritize causal variants across all 39 loci, completing the analysis within two hours. It automatically constructs a systematic workflow that extracts credible-set variants, performs AlphaGenome functional scoring for those variants in glutamatergic neurons, filters for protein-coding genes, and ranks variants by their maximum quantile impact scores across different functional modalities. For each locus, the AI co-scientist prioritizes the top variant, links it to a candidate target gene, summarizes its predicted molecular effects, and compiles a comprehensive markdown report detailing the functional consequences and biological significance of the prioritized gene. The AI-generated reports for the prioritized candidate variants and their predicted mechanisms across all 39 loci are provided in Supplementary Table 1. This workflow enables scalable hypothesis generation and mechanistic interpretation of GWAS loci in hours rather than weeks of manual review. While these hypotheses benefit from subsequent human evaluation, the workflow shifts scientific effort from manual execution to the synthesis of actionable biological insights.

Below is the prompt for scientific hypothesis/question generation:

Prompt: scientific hypothesis/question generation

```
You are an expert in scientific hypothesis generation, and your task is to propose
10 actionable scientific questions that use the tools to explore the data.

You have access to two Model Context Protocol (MCP) servers representing two
research artifacts ("papers"): MCP A: AlphaGenome and MCP B: ADHD GWAS data.

Constraints
- Use only resources exposed by AlphaGenome and ADHD GWAS data unless instructed
otherwise
- No fabricated citations or tool outputs; label missing evidence as "Unknown" and
propose how to obtain it
- If an MCP tool call fails or returns unexpected data, propose alternative
approaches or fallback strategies
- You may show reasoning/chain-of-thought separately, but the final output must be
valid JSON matching the schema below.

Scoring rubric (0-5 each)
- Novelty: originality vs. typical analyses in this domain
- Feasibility: executable with the available MCP tools/resources
- Impact: potential to advance understanding in the target domain
- Validation clarity: measurable success criterion with an objective metric
- Resource fit: leverage and synergy of both MCPs' specific capabilities

For each of the ten questions, include
- Title
- One-sentence summary
- Why this combination is promising (3-5 sentences with MCP pointers)
- Integration sketch: how components connect, key inputs/outputs, and any adapter
steps
```

```
- Minimal experiment plan: dataset, protocol, primary metric, success threshold,
ablations
- Risks and mitigations (explicitly call out data/analysis limitations)
- Resource checklist: which MCP tools/resources you will call and in what order
(specify server as "AlphaGenome" or "ADHD GWAS data")
- Individual scores and total score

Output format (strict JSON)
{
  "questions": [
    {
      "title": "",
      "summary_one_sentence": "",
      "rationale": "",
      "integration_sketch": "",
      "experiment_plan": {
        "dataset": "",
        "protocol": "",
        "primary_metric": "",
        "success_threshold": "",
        "ablations": ""
      },
      "risks_and_mitigations": "",
      "resource_checklist": [
        {"server": "AlphaGenome|ADHD GWAS data",
         "tool_or_resource": "",
         "purpose": ""}
      ],
      "scores": {
        "novelty": 0,
        "feasibility": 0,
        "impact": 0,
        "validation_clarity": 0,
        "resource_fit": 0,
        "total": 0
      }
    }
  ],
  "ranking_notes": ""
```

9.2 AI co-scientist analysis for causal gene prioritization at the rs887314 locus for psoriasis

Below is the prompt for causal gene identification using the AlphaGenome agent:

Prompt: causal gene identification using AlphaGenome agent

```
My AlphaGenome API key is XXX. Identify the causal genes and mechanisms in CD4+ T
cells for this variant using AlphaGenome MCP.

rs887314
```

```
GRCh38: 11:64285685
Reference allele: T
Alternative allele (effect allele): G
```

Below we provide the prompt used to instruct the AI co-scientist to validate the AlphaGenome causal gene prediction using the MPRA-coupled scCRISPRi and CD4+ T cell Perturb-seq paper agents. The prompt intentionally provides only a high-level research objective without specifying the analytical strategy, statistical methodology, or implementation details.

Prompt: causal gene validation using MPRA-coupled scCRISPRi and CD4+ T cell Perturb-seq agents

```
I have a set of candidate causal genes for **psoriasis in CD4+ T cells**: **GPR137,
NUDT22, RPS6KA4, BAD, TRPT1**.

## Step 1: Read and understand the data
Read the readme/manuscript and explore the data files from both papers. Summarize
what assays, perturbations, and readouts are available for each candidate gene.

## Step 2: Propose 10 research strategies
Propose **10 strategies to determine which candidate is the true causal gene**,
using **only the existing data from these two papers**. Do not propose new wet-lab
experiments.

For each strategy:
1. **Name** the approach concisely.
2. **Rationale:** Why this approach distinguishes the causal gene from bystanders.
3. **Data & analysis:** Which specific dataset(s) from Paper 1 or Paper 2 to query,
and what analysis to run.
4. **Expected outcome:** What a positive result looks like for the causal gene vs.
non-causal candidates.
```

The agent conducted the analysis and responded:

Response: Causal Gene Prioritization for Psoriasis at the rs887314 Locus

```
# Causal Gene Prioritization for Psoriasis at the rs887314 Locus

**Candidate genes:** GPR137, NUDT22, RPS6KA4, BAD, TRPT1

---

## Step 1: Data Summary

### Paper 1: Genome-wide CD4+ T-cell CRISPRi Perturb-seq

- **Dataset:** `CD4i_final.merged_DE_results.h5ad`
- **Design:** Genome-wide CRISPRi Perturb-seq in primary CD4+ T cells. 33,986
perturbation-condition pairs across 13,959 measured genes.
- **Conditions:** Rest, Stim8hr, Stim48hr
- **DE statistics layers:** log_fc, p_value, adj_p_value, baseMean, lfcSE
```

```
#### Candidate gene availability in Perturb-seq

| Gene | Perturbed? | Conditions | Cells (Rest / Stim8hr / Stim48hr) | Measured? |
DE signature strength |
|------|-----------|------------|-----------------------------------|-----------|-----------------
| **GPR137** | Yes | All 3 | 1,508 / 1,378 / 1,438 | Yes | Strong (~250-400 sig DE
genes) |
| **NUDT22** | Yes | All 3 | 275 / 403 / 660 | Yes | Moderate (~8-34 sig DE genes) |
| **RPS6KA4** | Yes | All 3 | 846 / 779 / 834 | **No** | Very weak (1 sig DE gene:
DMWD) |
| **BAD** | Yes | All 3 | 1,149 / 948 / 1,133 | Yes | Modest (~3-4 sig DE genes per
condition) |
| **TRPT1** | Yes | All 3 | 153 / 160 / 215 | Yes | Limited (1-2 sig DE genes; low
cell counts) |

#### Top DE genes per perturbation

**GPR137 KD** (strongest signature):
- BAD (rank 2-4, log_fc -1.8 to -2.6, adj_p < 1e-13), RP2, GPR180, HOMEZ, SIL1,
CLPB, TRMT10A
- GPR137 self-KD: log_fc -1.2 to -1.9

**NUDT22 KD:**
- TRPT1 (rank 2-3, log_fc -1.8 to -2.1, adj_p < 6e-08), ZNF280D, BTBD10, NUDT9
- Stim48hr: interferon-stimulated genes upregulated (HERC5, OASL, ISG15)

**RPS6KA4 KD:**
- Only 1 significant DE gene: DMWD (log_fc -0.8 to -1.2), likely a cis-effect
- Essentially no detectable transcriptional program

**BAD KD:**
- GPR137 (rank 2, log_fc -1.3 to -1.5, adj_p < 3e-18), UBE3C (rank 1-3, log_fc -0.5
to -0.6)
- BAD self-KD: log_fc -3.5 to -4.0

**TRPT1 KD:**
- NUDT22 (rank 1-2, log_fc -1.4 to -1.7, adj_p < 0.02)
- Limited power due to low cell counts

#### Cross-regulation between candidate genes

**Module 1: GPR137 <-> BAD (strong bidirectional co-regulation)**

| Perturbation | Target | Condition | log_fc | adj_p_value |
|---|---|---|---|---|
| GPR137 KD | BAD | Rest | -2.65 | 3.92e-40 |
| GPR137 KD | BAD | Stim8hr | -2.29 | 1.13e-13 |
| GPR137 KD | BAD | Stim48hr | -1.82 | 1.81e-22 |
| BAD KD | GPR137 | Rest | -1.47 | 1.95e-24 |
| BAD KD | GPR137 | Stim8hr | -1.39 | 2.85e-18 |
| BAD KD | GPR137 | Stim48hr | -1.25 | 2.85e-27 |
```

```
**Module 2: NUDT22 <-> TRPT1 (strong bidirectional co-regulation)**

| Perturbation | Target | Condition | log_fc | adj_p_value |
|---|---|---|---|---|
| NUDT22 KD | TRPT1 | Rest | -2.18 | 1.79e-08 |
| NUDT22 KD | TRPT1 | Stim8hr | -1.76 | 5.77e-14 |
| NUDT22 KD | TRPT1 | Stim48hr | -2.11 | 4.38e-32 |
| TRPT1 KD | NUDT22 | Stim8hr | -1.40 | 5.34e-03 |
| TRPT1 KD | NUDT22 | Rest | -1.66 | 1.60e-02 |
| TRPT1 KD | NUDT22 | Stim48hr | -1.75 | 1.31e-09 |

**No cross-regulation between the two modules:** GPR137/BAD KD does not affect
NUDT22 or TRPT1 (adj_p = 1.0), and vice versa. RPS6KA4 KD does not affect any
candidate.

---

### Paper 2: Genetic and Epigenetic Screens in Primary Human T Cells

- **Manuscript:** Ho, Dippel, McQuade et al., *Nature Genetics* (2025)
- **Key variant:** rs887314 (psoriasis PICS PIP = 0.13, in 80/90/95% credible sets)
- **Supplementary data:** `/supp_table/`

#### MPRA (Supp Table 1)

rs887314 is classified as **Enhancer_Skew** (emVar):
- Both alleles drive strong enhancer activity (log2FC ~2.1 and ~2.6, logPadj_BF >
74)
- Significant allelic skew: Log2Skew = 0.577 (FDR < 0.005)
- The alt allele (G) drives stronger expression
- The variant is the sole emVar on its haplotype

#### Functional annotations (Supp Table 4)

- Open chromatin in **all T cell types** tested (TH1, TH17, TH2, CD4, CD4 naive,
Treg, Jurkat, CD8)
- deltaSVM = 0.759 for naive CD4
- Nearest TSS gene: **BAD** (981 bp)
- TF motif disruption: SP4 (motifbreakR), PRGR (ananastra), plus ZN770, WT1, PATZ1,
TFDP1, VEZF1, ZN263, ZN436, KLF15, MAZ, THAP1, ZN467, FLI1 (2024 motifbreakR)
- UKBB top trait: WBC (PIP = 0.001)

#### scCRISPRi (Supp Table 21)

CRISPRi silencing of the rs887314 CRE - effect on all genes within 1 Mb:

| Gene | p_value | log2FC | Significant? |
|---|---|---|---|
| **BAD** | 9.25e-28 | -1.32 | **Yes** |
| **GPR137** | 1.23e-13 | -1.37 | **Yes** |
| OTUB1 | 0.009 | -0.13 | No (marginal) |
| RPS6KA4 | 0.112 | -0.37 | No |
| NUDT22 | 0.204 | -0.10 | No |
```

```
| TRPT1 | 0.916 | -0.01 | No |

Only **BAD and GPR137** are direct regulatory targets of the rs887314 CRE.

#### PICS fine-mapping (Supp Table 5)

- rs887314 is the lead SNP (r2 = 1 to itself)
- Psoriasis PICS PIP = 0.1268
- No PICS scores for other diseases (Crohn's, MS, RA, T1D, UC, IBD)

#### Proliferation screen (Supp Tables 25-26)

rs887314 is **not significant** in either screen:
- 1000-variant screen (Table 25): neg|fdr = 1.0, pos|fdr = 1.0
- 56-emVar screen (Table 26): neg|fdr = 0.955, pos|fdr = 1.0

#### V2G mapping (Supp Table 14)

| Gene | Distance to TSS | eQTL evidence | Other evidence |
|---|---|---|---|
| **BAD** | 981 bp | None | Distance only |
| **GPR137** | 15,623 bp | Blood (eQTLGen) | VEP functional prediction |
| **RPS6KA4** | 73,463 bp | Blood (eQTLGen) | Distance |
| **NUDT22** | 59,744 bp | None | Distance only |
| **TRPT1** | 59,431 bp | CD4+ T cell (Schmiedel 2018) | Distance |
| OTUB1 | 299,832 bp | None | Distance only |

#### STRING network clusters (Supp Table 17)

| Gene | Cluster | Cluster name | Cluster size |
|---|---|---|---|
| GPR137 | 16 | Light Green 6 | 7 genes |
| NUDT22 | 16 | Light Green 6 | 7 genes |
| TRPT1 | 16 | Light Green 6 | 7 genes |
| RPS6KA4 | 26 | Light Green 19 | 5 genes |
| BAD | 44 | Aquamarine 5 | 4 genes (apoptosis) |

GPR137, NUDT22, and TRPT1 cluster together. BAD is in a separate apoptosis cluster.

#### scCRISPRi STRING network (Supp Table 23)

BAD appears in cluster 9 (separate from T cell activation clusters).

---

## Step 2: 10 Research Strategies

### Strategy 1: scCRISPRi Direct Target Gene Identification

**Rationale:** The causal gene should be directly regulated by the CRE containing
rs887314. A gene that is significantly downregulated when the variant CRE is
silenced is a direct regulatory target, while non-target bystander genes should be
unaffected.
```

```
**Data & analysis:** Use scCRISPRi results from Paper 2 (Supp Table 21). Rank all
genes within 1 Mb by p-value when rs887314 CRE is targeted with CRISPRi.

**Expected outcome:** Only **BAD** (p=9.3e-28, log2FC=-1.32) and **GPR137**
(p=1.2e-13, log2FC=-1.37) are significantly affected. This narrows the candidate
list from 5 to 2. NUDT22 (p=0.20), RPS6KA4 (p=0.11), and TRPT1 (p=0.92) are
eliminated as direct targets of the variant CRE.

---

### Strategy 2: Perturbation Signature Magnitude Comparison

**Rationale:** If GPR137 and BAD co-regulate each other (as seen in Perturb-seq),
the true causal gene is the one whose perturbation produces a broader, more
disease-relevant transcriptional program beyond just affecting its co-regulated
partner.

**Data & analysis:** From Paper 1, compare the full genome-wide DE signatures of
GPR137 KD versus BAD KD across all 3 conditions. Count total significant DE genes,
compare effect sizes, and assess whether one gene's perturbation produces a strictly
more informative (larger, more coherent) downstream response than the other.

**Expected outcome:** GPR137 KD produces a much larger transcriptomic perturbation
(~250-400 significant DE genes) than BAD KD (~3-4 significant DE genes beyond GPR137
and UBE3C). This asymmetry suggests GPR137 is the more consequential gene whose
loss-of-function creates broader transcriptional changes, while BAD's signal is
largely limited to GPR137 co-downregulation.

---

### Strategy 3: Psoriasis-specific Pathway Enrichment of DE Signatures

**Rationale:** The true causal gene for psoriasis should produce a transcriptomic
signature enriched for psoriasis-relevant pathways (e.g., IL-17/IL-23 axis, T cell
activation, Th17 differentiation) when knocked down. Non-causal genes should show
irrelevant pathway enrichment.

**Data & analysis:** From Paper 1, extract the significant DE gene lists from GPR137
KD and BAD KD (especially under stimulated conditions). Run gene set enrichment
analysis (GSEA) against KEGG, GO Biological Process, or Reactome pathways. Compare
the enrichment of psoriasis-relevant pathways.

**Expected outcome:** The causal gene's DE signature should be enriched for T cell
activation, cytokine signaling, Th17 differentiation, and/or NF-kB signaling
pathways. The non-causal gene should lack these enrichments.

---

### Strategy 4: Correlation of Perturb-seq Profiles with Known Psoriasis Genes
```

Rationale: The causal gene should have a transcriptional perturbation profile (log_fc vector across all measured genes) that correlates with profiles of known psoriasis-associated genes. Bystander genes should show uncorrelated profiles.

Data & analysis: From Paper 1, compute Pearson correlations between each candidate's genome-wide log_fc vector and those of established psoriasis genes (e.g., IL23R, IL12B, TNFAIP3, STAT3, TRAF3IP2, NFKBIA) across each condition. Also correlate with genes in the T cell activation network (Paper 2, Fig. 3).

Expected outcome: The causal gene should show higher correlations with known psoriasis/T-cell-activation gene profiles. This places the causal gene within the disease-relevant regulatory network.

Strategy 5: Variant Proximity and eQTL Evidence Integration

Rationale: The variant rs887314 is located within the BAD promoter (981 bp from TSS). The closest gene to a regulatory variant is often, but not always, the true target. Integrating distance with eQTL and functional prediction data provides a more nuanced picture.

Data & analysis: From Paper 2 V2G data (Supp Table 14), compile all evidence types for each gene: (a) distance from rs887314 to TSS, (b) eQTL evidence (which tissues, which databases), (c) VEP functional prediction, (d) promoter-capture Hi-C (pcHiC) contacts. Compute a composite V2G score.

Expected outcome: BAD has the shortest distance (981 bp), but GPR137 has eQTL support in blood (eQTLGen) and VEP functional prediction. TRPT1 has eQTL support in CD4+ T cells. RPS6KA4 has eQTL in blood. This analysis differentiates candidates by strength and type of V2G evidence, but alone cannot resolve between BAD and GPR137.

Strategy 6: MPRA Allelic Skew Direction vs. eQTL Direction Concordance

Rationale: If the variant truly causes disease through a specific gene, the direction of the MPRA allelic skew (which allele increases/decreases CRE activity) should be concordant with the eQTL direction for that gene (which allele increases/decreases expression). Discordance suggests the gene may not be the true target.

Data & analysis: From Paper 2 Supp Table 1, extract the MPRA Log2Skew direction for rs887314. From Supp Table 4, extract eQTL beta and direction for the listed eQTL gene (BAD). Also query the V2G table for eQTL betas of GPR137, RPS6KA4, and TRPT1 from eQTLGen and Schmiedel 2018 datasets.

Expected outcome: The causal gene should show concordance between MPRA allelic effect direction and eQTL effect direction. Discordant genes are less likely to be causal through this variant.

```
### Strategy 7: TF Motif Disruption Analysis for Psoriasis-relevant TFs

**Rationale:** rs887314 is predicted to disrupt specific TF binding motifs. If the
disrupted TF is relevant to psoriasis biology (e.g., ZNF563, which shows
disease-specific effects at psoriasis loci per the manuscript), this connects the
variant mechanistically to disease. The target gene of the disrupted TF should be
the causal gene.

**Data & analysis:** From Paper 2 Supp Table 10 (T cell motifbreakR results),
extract all TF motifs disrupted by rs887314 (SP4, PRGR, ZN770, WT1, PATZ1, TFDP1,
VEZF1, ZN263, etc. from Table 4). Cross-reference with psoriasis-specific TF
enrichment from Supp Table 35. Compare which candidate genes' promoters contain
binding sites for the disrupted TFs.

**Expected outcome:** The TFs disrupted by rs887314 should regulate the
promoter/enhancer of the causal gene. If SP4 or related TFs are known to regulate
GPR137 or BAD expression specifically, this provides a mechanistic link from variant
to gene.

---

### Strategy 8: STRING Network Topology and T Cell Activation Cluster Membership

**Rationale:** The manuscript shows that emVar target genes converge on a T cell
activation network. The causal gene for a T cell-mediated disease like psoriasis
should be embedded in this network, while bystander genes should be peripheral or
absent.

**Data & analysis:** From Paper 2 Supp Tables 17 and 23, examine which STRING
network cluster each candidate belongs to. BAD is in cluster 44 (apoptosis, separate
from T cell activation). GPR137, NUDT22, and TRPT1 are in cluster 16. RPS6KA4 is in
cluster 26. Cross-reference with the scCRISPRi STRING network (Table 23), where BAD
appears in a separate cluster.

**Expected outcome:** If the disease operates through T cell activation pathways (as
strongly supported for psoriasis), GPR137's cluster membership is more relevant.
BAD's segregation into the apoptosis cluster (separate from T cell activation) could
argue against it being the primary causal gene for psoriasis, unless apoptosis
dysregulation is the relevant mechanism.

---

### Strategy 9: Bidirectional Regulatory Hierarchy Analysis

**Rationale:** The strong mutual co-regulation between GPR137 and BAD (each KD
downregulates the other) in Perturb-seq could indicate they share a regulatory
element or are in a regulatory hierarchy. Identifying which is "upstream" (i.e.,
which KD effect is stronger on the other, after accounting for the shared CRE)
reveals the primary target.
```

Data & analysis: From Paper 1, compare: (a) The fold change of BAD when GPR137 is knocked down (log_fc = -1.8 to -2.6) vs. the fold change of GPR137 when BAD is knocked down (log_fc = -1.3 to -1.5). (b) Whether one gene's self-knockdown effect is proportionally larger. (c) Whether the co-downregulation is explained by sharing the same CRE (the CRISPRi guide targeting one gene's promoter physically also silences the nearby gene's enhancer).

Expected outcome: The variant rs887314 sits in BAD's promoter (981 bp). The scCRISPRi data shows that silencing this CRE affects both BAD and GPR137 with comparable effect sizes (log2FC -1.32 vs -1.37). In Perturb-seq (where CRISPRi targets each gene's own TSS), GPR137 KD causes a stronger downregulation of BAD (up to log_fc = -2.6) than BAD KD does on GPR137 (up to -1.5). This asymmetry suggests GPR137 may be the functionally dominant gene.

Strategy 10: Variant CRE Silencing Profile vs. Gene-level KD Profile Matching

Rationale: The scCRISPRi screen silences the variant CRE (rs887314), while the Perturb-seq silences each gene's TSS individually. If the variant acts through Gene X, then the transcriptional profile of silencing rs887314 should resemble the Perturb-seq profile of Gene X KD more than that of Gene Y KD.

Data & analysis: The scCRISPRi data (Paper 2, Table 21) provides a list of all genes within 1 Mb and their DE when rs887314 is targeted (438 gene-level tests). The Perturb-seq data (Paper 1) provides genome-wide DE for GPR137 KD and BAD KD independently. Compare the overlap/concordance of DE gene sets: (1) Genes upregulated/downregulated by rs887314 CRE silencing vs. GPR137 KD, and (2) vs. BAD KD. Compute correlation of effect sizes for the overlapping measured genes.

Expected outcome: The variant CRE silencing profile should correlate more strongly with the Perturb-seq profile of the true causal gene. If rs887314 CRE silencing looks more like GPR137 KD than BAD KD (or vice versa), this identifies the primary mediator of the variant's effect.

Summary

The data strongly converge on a **2-gene shortlist: BAD and GPR137**. The scCRISPRi data from Paper 2 eliminates NUDT22, RPS6KA4, and TRPT1 as direct targets of the rs887314 CRE. The Perturb-seq data from Paper 1 reveals a striking mutual co-regulation between BAD and GPR137.

Most discriminating strategies to resolve BAD vs. GPR137:
1. **Strategy 2** (DE signature magnitude): GPR137 KD produces ~100x more significant DE genes than BAD KD
2. **Strategy 9** (Regulatory hierarchy): GPR137 KD → BAD effect is stronger than BAD KD → GPR137
3. **Strategy 10** (Profile matching): Compare variant CRE silencing profile to each gene's individual KD profile

```
4. **Strategy 3** (Pathway enrichment): The causal gene's signature should enrich
for psoriasis-relevant pathways
```

Next, the human asked the agent to use strategy 10. We report the key human interventions made during the analysis process. Notably, all interventions were issued exclusively as natural language instructions; the human investigator did not write, modify, or inspect any code at any stage.

- The agent initially performed the analysis for only *BAD* and *GPR137*. The human directed the agent to extend the analysis to all five candidate genes.
- The agent initially excluded all five candidate genes from the correlation analysis. The human corrected this, instructing the agent to exclude only the knockdown gene itself in each respective analysis.
- The human instructed the agent to generate scatter plots in R using `ggplot2` rather than Python (`matplotlib`), and further guided it to customize text annotations and color schemes for clarity. For quantitative trends, the agent was directed to fit linear regression lines after outlier removal to prevent distortion from extreme expression values.

We also report noteworthy behaviors exhibited by the agent during the analysis process. Upon inspecting the correlation analysis scatter plots, the agent identified the presence of outliers and consequently performed both Pearson and Spearman correlation analyses to ensure robustness.

10 Maintenance agent for agent availability

To reduce the maintenance burden of paper agents as upstream codebases and dependencies evolve, each MCP server can potentially be paired with a maintenance agent implemented using agentic code execution frameworks such as Claude Code Web or Jules, which periodically or upon upstream changes re-runs validation tests in an isolated environment. When failures occur due to API changes or dependency drift, the agent attempts automated repair and re-validation, and flags unresolved issues for human review.

11 Scope of agentification

Another consideration is the scope of agentification. While the paper is the conventional unit of scientific communication, it is not always the best unit for agentification. In many fields, an idea evolves across a sequence of publications, each adding refinements, benchmarks, or applications. In such cases, the most useful agent may not represent a single paper but rather a collection of related works aggregated into a coherent interface. A single MCP can encapsulate multiple related papers. We plan to extend Paper2Agent to flexibly accommodate this broader scope.

12 Security, intellectual property, and attribution considerations

Exposing research code as executable agents also introduces security, intellectual property, and attribution challenges. MCP servers may inherit vulnerabilities from their source papers and associated code repositories; tools are therefore executed in sandboxed environments with restricted

permissions and retain traceable links to their source code for provenance. MCP servers remain subject to the original licenses of the underlying code, and third-party distribution should preserve authorship and usage constraints. When agent-generated analyses contribute to downstream discoveries, we recommend treating paper-derived agents as computational instruments, with credit assigned to the original method developers, data generators, and human investigators who designed the study.

References

[1] 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*, pages 1–3, 2025.

[2] Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, et al. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

[3] Sakana.AI. Sakana "ai scientist". *Sakana.ai Website*, 2024.

[4] Ali Essam Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J Szostkiewicz, Jon M Laurent, Muhammed T Razzak, Andrew D White, Michaela M Hinks, and Samuel G Rodriques. Robin: A multi-agent system for automating scientific discovery. *arXiv preprint arXiv:2505.13400*, 2025.

[5] 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*, pages 1–14, 2025.

[6] Samuel Alber, Bowen Chen, Eric Sun, Alina Isakova, Aaron J Wilk, and James Zou. Cellvoyager: Ai compbio agent generates new insights by autonomously analyzing biological data. *bioRxiv*, pages 2025–06, 2025.

[7] 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*, pages 2025–05, 2025.

[8] Minju Seo, Jinheon Baek, Seongyun Lee, and Sung Ju Hwang. Paper2code: Automating code generation from scientific papers in machine learning. *arXiv preprint arXiv:2504.17192*, 2025.

[9] Cameron S Movassaghi, Amanda Momenzadeh, and Jesse G Meyer. From articles to code: On-demand generation of core algorithms from scientific publications. *arXiv preprint arXiv:2507.22324*, 2025.

[10] Piotr Nowakowski, Eryk Ciepiela, Daniel Harężlak, Joanna Kocot, Marek Kasztelnik, Tomasz Bartyński, Jan Meizner, Grzegorz Dyk, and Maciej Malawski. The collage authoring environment. *Procedia Computer Science*, 4:608–617, 2011.

[11] Adam Rule, Amanda Birmingham, Cristal Zuniga, Ilkay Altintas, Shih-Cheng Huang, Rob Knight, Niema Moshiri, Mai H Nguyen, Sara Brin Rosenthal, Fernando Pérez, et al. Ten simple rules for writing and sharing computational analyses in jupyter notebooks, 2019.

[12] Robert Stojnic and Ross Taylor. Papers with code is joining facebook ai, December 2019. Medium article.

[13] Benjamin Ragan-Kelley, Carol Willing, F Akici, D Lippa, D Niederhut, and M Pacer. Binder 2.0-reproducible, interactive, sharable environments for science at scale. In *Proceedings of the 17th python in science conference*, pages 113–120. F. Akici, D. Lippa, D. Niederhut, and M. Pacer, eds., 2018.

[14] Thomas Staubitz, Hauke Klement, Ralf Teusner, Jan Renz, and Christoph Meinel. Codeocean-a versatile platform for practical programming excercises in online environments. In *2016 IEEE Global Engineering Education Conference (EDUCON)*, pages 314–323. IEEE, 2016.

[15] Eric D Sun, Rong Ma, Paloma Navarro Negredo, Anne Brunet, and James Zou. Tissue: uncertainty-calibrated prediction of single-cell spatial transcriptomics improves downstream analyses. *Nature Methods*, 21(3):444–454, 2024.

[16] Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

[17] Joseph Bloom, Curt Tigges, Anthony Duong, and David Chanin. Saelens. https://github.com/decoderesearch/SAELens, 2024.

[18] Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Spotting llms with binoculars: Zero-shot detection of machine-generated text. *arXiv preprint arXiv:2401.12070*, 2024.

[19] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[20] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

[21] Victor Chernozhukov, Mert Demirer, Esther Duflo, and Iván Fernández-Val. Fisher–schultz lecture: Generic machine learning inference on heterogeneous treatment effects in randomized experiments, with an application to immunization in india. *Econometrica*, 93(4):1121–1164, 2025.

[22] Kay H Brodersen, Fabian Gallusser, Jim Koehler, Nicolas Remy, and Steven L Scott. Inferring causal impact using bayesian structural time-series models. 2015.

[23] Vincent Knight and James Campbell. Nashpy: A python library for the computation of nash equilibria. *Journal of Open Source Software*, 3(30):904, 2018.

[24] Daniel Foreman-Mackey, David W Hogg, Dustin Lang, and Jonathan Goodman. emcee: the mcmc hammer. *Publications of the Astronomical Society of the Pacific*, 125(925):306–312, 2013.

[25] Ying Jin and Emmanuel J Candès. Model-free selective inference under covariate shift via weighted conformal p-values. *Biometrika*, 113(1):asaf066, 2026.

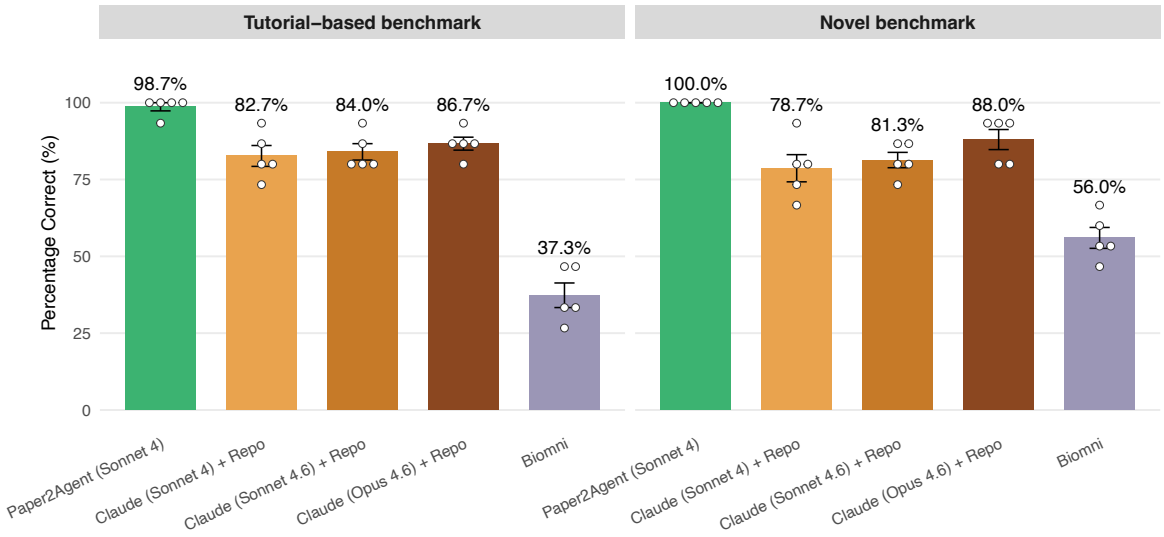
[26] AvsecZiga, Natasha Latysheva, Jun Cheng, Guido Novati, Kyle R Taylor, Tom Ward, Clare Bycroft, Lauren Nicolaisen, Eirini Arvaniti, Joshua Pan, et al. Advancing regulatory variant effect prediction with alphagenome. *Nature*, 649(8099):1206–1218, 2026.

[27] Camiel M van der Laan, Hill F Ip, Marijn Schipper, Jouke-Jan Hottenga, Beate St Pourcain, Tetyana Zayats, René Pool, Eva ML Krapohl, Isabell Brikell, María Soler Artigas, et al. Genome-wide association meta-analysis of childhood adhd symptoms and diagnosis identifies new loci and potential effector genes. *Nature genetics*, pages 1–9, 2025.

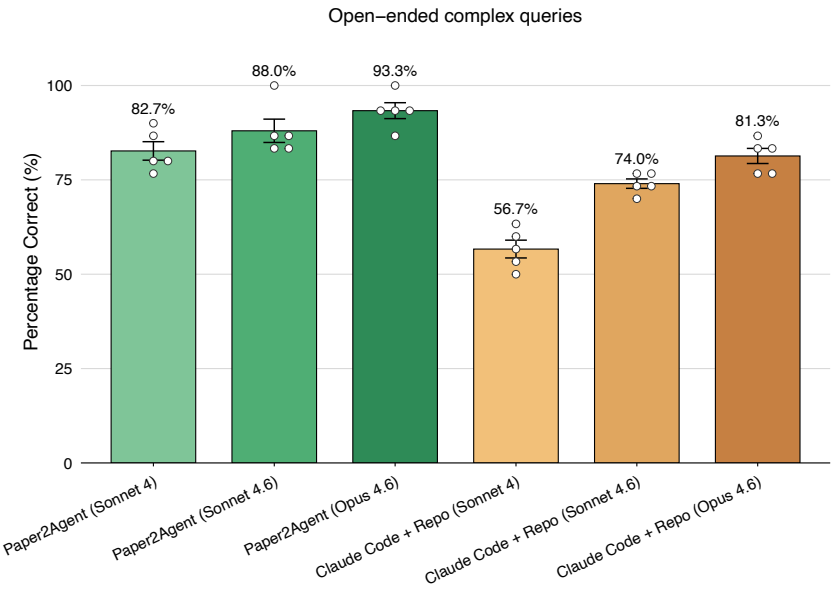
[28] Ning Huang, Donghui Zhang, Fangyuan Li, Peiyuan Chai, Song Wang, Junlin Teng, and Jianguo Chen. M-phase phosphoprotein 9 regulates ciliogenesis by modulating cp110-cep97 complex localization at the mother centriole. *Nature communications*, 9(1):4511, 2018.

```python
def visualize_variant_effects(
    # Primary data inputs - variant specification
    variant_chromosome: Annotated[str, "Chromosome name (e.g., 'chr22')"] = "chr22",
    variant_position: Annotated[int, "Genomic position"] = 36201698,
    variant_reference_bases: Annotated[str, "Reference allele"] = "A",
    variant_alternate_bases: Annotated[str, "Alternate allele"] = "C",
    # Analysis parameters with tutorial defaults
    organism: Annotated[Literal["human", "mouse"], "Organism to analyze"] = "human",
    sequence_length: Annotated[Literal["2KB", "16KB", "100KB", "500KB", "1MB"], "Length of sequence around variant to predict"] =
"1MB",
    ontology_terms: Annotated[list[str], "List of cell and tissue ontology terms"] = None,
    # Gene annotation options
    plot_gene_annotation: Annotated[bool, "Include gene annotation in plot"] = True,
    plot_longest_transcript_only: Annotated[bool, "Show only longest transcript per gene"] = True,
    # Output types to plot
    plot_rna_seq: Annotated[bool, "Plot RNA-seq tracks"] = True,
    plot_cage: Annotated[bool, "Plot CAGE tracks"] = True,
    plot_atac: Annotated[bool, "Plot ATAC-seq tracks"] = False,
    plot_dnase: Annotated[bool, "Plot DNase tracks"] = False,
    plot_chip_histone: Annotated[bool, "Plot ChIP-seq histone tracks"] = False,
    plot_chip_tf: Annotated[bool, "Plot ChIP-seq transcription factor tracks"] = False,
    plot_splice_sites: Annotated[bool, "Plot splice sites"] = True,
    plot_splice_site_usage: Annotated[bool, "Plot splice site usage"] = False,
    plot_contact_maps: Annotated[bool, "Plot contact maps"] = False,
    plot_splice_junctions: Annotated[bool, "Plot splice junctions"] = False,
    # DNA strand filtering
    filter_to_positive_strand: Annotated[bool, "Filter tracks to positive strand only"] = False,
    filter_to_negative_strand: Annotated[bool, "Filter tracks to negative strand only"] = True,
    # Visualization options
    ref_color: Annotated[str, "Color for reference allele"] = "dimgrey",
    alt_color: Annotated[str, "Color for alternate allele"] = "red",
    plot_interval_width: Annotated[int, "Width of plot interval in base pairs"] = 43008,
    plot_interval_shift: Annotated[int, "Shift of plot interval from variant center"] = 0,
    api_key: Annotated[str | None, "API key for AlphaGenome model"] = None,
    out_prefix: Annotated[str | None, "Output file prefix"] = None,
)
  ...
  return {
          ...
          "source": "https://github.com/google-deepmind/alphagenome/blob/main/colabs/variant_scoring_ui.ipynb",
    ...
  }
```

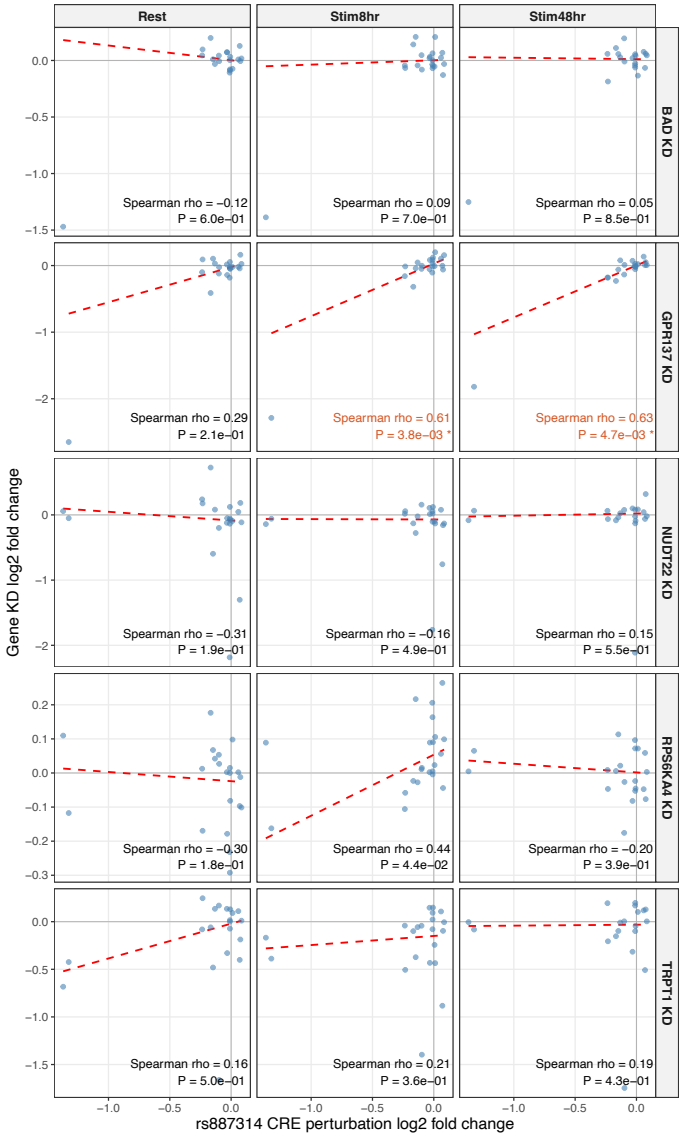
Supplementary Figure 1. Exposed AlphaGenome MCP tool for customizable variant-effect visualization. The Paper2Agent-generated visualize_variant_effects tool exposes typed and annotated parameters for specifying a genomic variant, organism, sequence context, cell or tissue ontology terms, gene annotations, prediction modalities and visualization settings. Users can select tracks including RNA-seq, CAGE, ATAC-seq, DNase-seq, ChIP-seq, splice sites, splice junctions and contact maps, as well as configure strand filtering, interval width and display colors. Tutorial-derived defaults are preserved but can be modified for new analyses. The tool output includes a traceable link to the corresponding source code in the original AlphaGenome repository.



Supplementary Figure 2. Paper2Agent maintains an accuracy advantage over coding-agent baselines on tutorial-based and novel AlphaGenome queries. Accuracy was evaluated on 15 tutorial-based queries and 15 novel queries. Paper2Agent used Sonnet 4 with the generated AlphaGenome MCP, whereas the Claude + Repo baselines received direct access to the AlphaGenome repository and used Sonnet 4, Sonnet 4.6 or Opus 4.6. Biomni was evaluated using its API-based implementation. Bars indicate the mean percentage of correctly answered queries across five independent runs, circles represent individual runs, and error bars indicate ±1 s.e.m. Values above the bars denote mean accuracy.



Supplementary Figure 3. Paper2Agent maintains an accuracy advantage over direct-repository coding-agent baselines on complex open-ended queries. Performance was evaluated on 30 researcher-style AlphaGenome queries requiring multi-step tool composition and biological synthesis. Paper2Agent used the generated AlphaGenome MCP with Sonnet 4, Sonnet 4.6 or Opus 4.6, whereas the corresponding Claude Code + Repo baselines received direct access to the AlphaGenome repository. Bars indicate the mean percentage of correctly answered queries across five independent runs, circles represent individual runs, and error bars indicate ±1 s.e.m. Values above the bars denote mean accuracy



Supplementary Figure 4. Correlation between rs887314 cis-regulatory element perturbation and candidate-gene knockdown effects. Scatterplots compare changes in downstream gene expression following perturbation of the rs887314 cis-regulatory element (CRE; x axis) with changes following knockdown of each candidate gene (y axis) in primary human CD4⁺ T cells at rest and 8 h or 48 h after stimulation. Rows correspond to *BAD, GPR137, NUDT22, RPS6KA4* and *TRPT1* knockdown, and each point represents a downstream gene measured in both perturbation datasets; the perturbed candidate gene was excluded. Red dashed lines indicate linear fits after outlier removal. Spearman's ρ and two-sided *P* values are shown in each panel. Orange annotations and asterisks indicate correlations significant after Benjamini–Hochberg correction (FDR < 0.05). *GPR137* knockdown showed significant concordance with the rs887314 CRE-perturbation signature under both stimulated conditions but not at rest.

Supplementary Table 1. Paper2Agent co-scientists use AlphaGenome to identify likely causal variants and their mechanisms contributing to ADHD risk for 39 loci.

locus_num	rsID	gene	splice	rna_seq	dnase	chip_tf	status	interpretation
1	rs653953	PTPRF	0.9470497	0.9847427	0.64188623	0.67455524	Complete	Among 15 candidate variants in loci 1, rs653953 alters PTPRF expression in glutamatergic neurons (AlphaGenome quantile scores: RNA-seq = 0.459, DNase = 0.333, ChIP-TF = 0.363). PTPRF (LAR) is a receptor tyrosine phosphatase from the LAR-RPTP subfamily that regulates dendrite and axon growth and guidance, synapse formation and differentiation, and synaptic activity. The D2 domain acts as a docking domain for interaction with scaffolding proteins such as liprins to modulate synapse formation, playing important roles in neural development and synaptic function.
2	rs6725549	ALKAL2	0	0.8596574	0.9983986	0.6929783	Complete	Among 149 candidate variants in loci 2, rs6725549 alters ALKAL2 expression in glutamatergic neurons. ALKAL2 (FAM150B/AUGÐ±) is a small secreted peptide that strongly activates ALK receptor tyrosine kinase signaling. ALK is expressed throughout the nervous system particularly during embryogenesis and promotes neuronal differentiation and maturation through RAS, PI3K, IRS1 signaling cascades and specific activation of the ERK MAP-kinase pathway. The activation of ALK and LTK by ALKALs provides an evolutionarily conserved functional signaling unit in the neural crest and derived cells, with inflammation or injury enhancing ALKAL2 expression in TRPV1+ sensory neurons where it is enriched in peptidergic nociceptors mediating persistent pain.
3	rs12478908	CTNNA2	0.84946334	0.7631148	0.9884112	0.494645	Complete	Among 38 candidate variants in loci 3, rs12478908 alters CTNNA2 expression in glutamatergic neurons. CTNNA2 (Catenin Alpha 2, also known as alpha-N-catenin) is a linker between cadherin adhesion receptors and the actin cytoskeleton that regulates cell-cell adhesion and differentiation in the nervous system. It is essential for stabilizing dendritic spines in rodent hippocampal neurons; Catna2-null mice have abnormally motile dendritic spine heads with actively protruding and retracting filopodia from synaptic contact sites, while alpha N-catenin overexpression reduces spine turnover and increases spine and synapse density. CTNNA2 has been repeatedly identified in genome-wide association studies of autism spectrum disorders and is required for proper cortical neuronal migration and neurite growth.
4	rs2044934		0	0.3122239	0.99746084	0.9953788	Complete	Among 232 candidate variants in loci 4, rs2044934 affects a regulatory region affecting ID3 transcription factor binding sites. This pure regulatory variant likely influences gene expression through transcription factor binding modulation rather than direct protein-coding effects. The large number of candidate variants suggests a complex regulatory landscape with multiple enhancer elements controlling glutamatergic neuron gene expression programs.
5	rs11684042	ZEB2	0	0.5767315	0.2699999	0.994183	Complete	Among 18 candidate variants in loci 5, rs11684042 alters ZEB2 expression in glutamatergic neurons. ZEB2 (zinc finger E-box-binding homeobox 2) is a multi-zinc finger transcriptional repressor that recognizes and binds to consensus E-box binding motifs and interacts with receptor-activated SMADs of the BMP and TGF-Œ≤ pathways. ZEB2 was identified as the causal gene for Mowat-Wilson Syndrome (MOWS), a haploinsufficient syndrome characterized by intellectual disability, microcephaly, congenital heart defects, Hirschsprung's disease, and characteristic facial features. ZEB2 was found to be one of the earliest factors expressed in prospective human neural crest, with knockdown revealing a role in establishing the neural crest state while repressing pre-placodal and non-neural ectoderm genes, regulating early human neural crest specification by modulating proper BMP signaling.
6	rs35941397	ABCA12	0	0.635053	0.97174054	0.5118722	Complete	Among 20 candidate variants in loci 6, rs35941397 alters ABCA12 expression in glutamatergic neurons. ABCA12 is a keratinocyte transmembrane lipid transporter protein associated with transport of lipids in lamellar granules to the apical surface of granular layer keratinocytes. It is localized from the Golgi apparatus to the cell periphery and works in the transport of lipids into the trans-Golgi network and lamellar granules to accumulate lipids that are essential to skin barrier formation. ABCA12 mutations are known to underlie three major types of autosomal recessive congenital ichthyoses: harlequin ichthyosis, lamellar ichthyosis, and congenital ichthyosiform erythroderma. Its primary function is in skin/epithelial tissue with limited direct neuronal relevance, though it may have general cellular membrane organization functions.
7	rs1392576	SGO1	0	0.5993673	0.9847427	0.5286973	Complete	Among 32 candidate variants in loci 7, rs1392576 alters SGO1 expression in glutamatergic neurons. SGO1 (Shugoshin 1) is an evolutionarily conserved protein that functions as a protector of centromeric cohesin during mitosis and meiosis, ensuring chromosomal stability. Cohesin at the centromeres is protected from the prophase pathway by the Shugoshin 1 (Sgo1)-PP2A complex. Vertebrate Sgo1 depletion results in massive loss of cohesin and severe chromosome missegregation. The primary direct function of Sgo1 in bi-orientation is to recruit PP2A to the centromere, which is critical for dephosphorylating cohesin and protecting it from premature dissociation. Human Sgo1 downregulation leads to chromosomal instability in colorectal cancer, with relevance for neural progenitor proliferation and maintenance of genomic stability during glutamatergic neuron development.
8	rs28535523	UBA7	0.9801434	0.9998168	0.40266997	0.79053134	Complete	Among 93 candidate variants in loci 8, rs28535523 alters UBA7 splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 1.000, RNA-seq = 1.000, DNase = 0.495, ChIP-TF = 0.520). UBA7 is the E1-activating enzyme for ISG15 conjugation (ISGylation), playing a critical role in protein post-translational modification essential for antiviral immunity. It restricts viral replication of rabies, influenza, sindbis, rotavirus, and CMV viruses, and catalyzes the attachment of ISG15 to cellular proteins including P53. Related ubiquitin systems regulate synaptic plasticity, and protein modification pathways are important for glutamate receptor trafficking and post-translational regulation at excitatory synapses, making UBA7 critical for protein quality control and synaptic function in glutamatergic neurons.
9	rs59404643	FOXP1	0.94468737	0.9884112	1.1377566e-12	0.23762526	Complete	Among 31 candidate variants in loci 9, rs59404643 alters FOXP1 expression in glutamatergic neurons. FOXP1 is a forkhead box (FOX) transcription factor associated with development of brain, heart, esophagus, lung, immune system, and spinal motor neurons. FOXP1 syndrome (200+ individuals identified) is among the most common genes causing autism spectrum disorder (SFARI Gene database), characterized by delays in motor and language milestones, mild-to-severe intellectual deficits, speech and language impairment in all individuals, and behavioral abnormalities including ASD, ADHD, anxiety, repetitive behaviors, sleep disturbances, and sensory symptoms. FOXP1 has a more global impact on brain development than FOXP2, and brain-specific Foxp1 deletion shows pronounced disruption of the developing striatum with abnormal neuronal morphogenesis, reduced excitability, and an imbalance of excitatory to inhibitory input in CA1 hippocampal neurons.
10	rs17516470	CADM2	0.9708846	0.9996424	0.17133315	0.9071543	Complete	Among 255 candidate variants in loci 10, rs17516470 alters CADM2 splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.985, RNA-seq = 0.642, DNase = 0.363, ChIP-TF = 0.874). CADM2 is a synaptic cell adhesion molecule (SynCAM family, immunoglobulin superfamily) that regulates synapse organization and trans-synaptic adhesion. CADM2 has appeared among top associations in genome-wide association studies for 50+ traits including cognitive function, risk-taking behavior, personality, educational attainment, autism spectrum disorders, physical activity, and BMI/obesity, suggesting alterations in CADM2 expression affect neuronal connectivity with broad phenotypic consequences.
11	chr3:117883166						No protein-coding genes	Locus 11 contains 130 candidate variants but no protein-coding genes in the analyzed region. This locus likely harbors regulatory variants affecting long-range gene expression through enhancer or silencer mechanisms in glutamatergic neurons via chromatin looping and 3D genome architecture.
12	rs2651566	EMCN	0.60787	0.87398493	0.99751866	0.5843792	Complete	Among 67 candidate variants in loci 12, rs2651566 alters EMCN splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.965, RNA-seq = 0.711, DNase = 0.935, ChIP-TF = 0.743). EMCN (endomucin) is an endothelial-specific component of the vascular glycocalyx that regulates VEGFR2 endocytosis and modulates VEGF signaling. It functions as an anti-adhesive molecule preventing neutrophil-endothelial interactions, stabilizes the blood-retinal barrier, and is critical for vascular permeability and barrier function, suggesting importance for blood-brain barrier maintenance and neurovascular coupling supporting high metabolic demands of glutamatergic neurons.
13	chr4:111489199		0	0	0.9950484	0.7817188	No protein-coding genes	Locus 13 contains 10 candidate variants but no protein-coding genes in the analyzed region. This locus likely harbors regulatory variants affecting long-range gene expression through enhancer or silencer mechanisms in glutamatergic neurons.
14	rs10074873	ZNF131	0.8908594	0.9997094	0.9978883	0.83002186	Complete	Among 21 candidate variants in loci 14, rs10074873 alters ZNF131 splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.991, RNA-seq = 0.733, DNase = 0.431, ChIP-TF = 0.495). ZNF131 is a zinc finger transcription factor with POK protein superfamily characteristics, predominantly expressed in the developing nervous system and adult brain. In contrast to most POZ-ZF proteins that act as transcriptional repressors, ZNF131 acts as a transcriptional activator and is broadly required for glioblastoma stem-like cell viability while dispensable for neural progenitor cells, promoting expression of Joubert Syndrome ciliopathy genes and suppressing centrosome fragmentation.

Supplementary Table 1 (continued).

locus_num	rsID	gene	splice	rna_seq	dnase	chip_tf	status	interpretation
15	rs324886	MEF2C	0	0.3225975	0.78616506	0.9824973	Complete	Among 4 candidate variants in loci 15, rs324886 alters MEF2C expression in glutamatergic neurons (AlphaGenome quantile scores: RNA-seq = 0.823, DNase = 0.183, ChIP-TF = 0.782). MEF2C (Myocyte Enhancer Factor 2C) is a transcription factor of the MADS-BOX family involved in embryonic brain development, neuronal formation and differentiation, and growth and pruning of axons and dendrites. MEF2C haploinsufficiency syndrome (MHS) is a severe form of autism spectrum disorder/intellectual disability characterized by speech absence, seizures, and motor abnormalities. Conditional embryonic deletion causes dramatic excitatory/inhibitory (E/I) imbalance with increased inhibitory and decreased excitatory synaptic transmission, and differential gene expression overlaps with synapse- and autism-linked genes.
16	chr5:104671732		0	0	0.9939092	0.9417692	No protein-coding genes	Locus 16 contains 10 candidate variants but no protein-coding genes in the analyzed region. This locus likely harbors regulatory variants affecting long-range gene expression through enhancer or silencer mechanisms in glutamatergic neurons.
17	rs358649	KCTD16	0	0	0.99773717	0.7105387	Complete	Among 53 candidate variants in loci 17, rs358649 alters KCTD16 splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.477, RNA-seq = 0.353, DNase = 0.989, ChIP-TF = 0.722). KCTD16 is an auxiliary subunit of GABA_B receptor complexes that influences neuronal excitability by regulating GABA_B receptor-mediated gating of postsynaptic ion channels. Robust KCTD16 expression is detected in amygdala, caudate nucleus, and hippocampus, with co-occurrence of KCTD16 antibodies pointing toward paraneoplastic origin in anti-GABA_BR encephalitis. The exceptional DNase accessibility score (98.9%) suggests the variant creates or disrupts a critical regulatory element affecting GABAergic modulation of glutamatergic neurons.
18	rs6935524	COL19A1	0.9986396	0.91866904	0.61394674	0.88937473	Complete	Among 19 candidate variants in loci 18, rs6935524 alters COL19A1 splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.999, RNA-seq = 0.919, DNase = 0.614, ChIP-TF = 0.889). COL19A1 encodes the alpha chain of type XIX collagen (FACIT collagen family) that acts as a cross-bridge between fibrils and other extracellular matrix molecules. It is expressed by subsets of hippocampal interneurons (colocalizes with interneuron marker Gad67) and contributes to the formation and maintenance of synapses in the mammalian nervous system. COL19A1 is critical for neuronal maturation, circuit formation, long-term memory (via perineuronal nets), and regulation of synaptic plasticity at excitatory and inhibitory synapses.
19	rs10262103	FOXP2	0.82762593	0.7725826	0.9785014	0.79053134	Complete	Among 50 candidate variants in loci 19, rs10262103 alters FOXP2 splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.971, RNA-seq = 0.821, DNase = 0.607, ChIP-TF = 0.826). FOXP2 is a forkhead box transcription factor critical for speech and language development; mutations cause developmental verbal dyspraxia and childhood apraxia of speech. FOXP2 is expressed in Layer 6 excitatory projection neurons (layers 6a and 6b), downregulates SRPX2 (glutamatergic synapse formation in cortex) while increasing glutamatergic synapse number, and drives Pou3f2 expression for differentiation of glutamatergic neurons. 2024 research identified multiple L6 Foxp2-expressing glutamatergic neuron types with layer-specific expression and neuromodulatory regulation, with key roles in synaptic plasticity and sensorimotor integration.
20	rs7014625	RUNX1T1	0	0.96761274	0.6682194	0.83706266	Complete	Among 25 candidate variants in loci 20, rs7014625 alters RUNX1T1 expression in glutamatergic neurons (AlphaGenome quantile scores: RNA-seq = 0.968, DNase = 0.668, ChIP-TF = 0.837). RUNX1T1 (also known as ETO) is a transcriptional corepressor that turns off gene activity by recruiting histone deacetylases to negatively influence transcription. The brain has the highest Runx1t1 level relative to other organs, and it has a regulatory function as a transcriptional corepressor in nervous system differentiation and development. Decreased Runx1t1 expression reduces neuronal differentiation of radial glial cells, while increased expression promotes neuronal differentiation, suggesting importance for glutamatergic neuron specification from progenitors.
21	rs10124636	CER1	0	0.9923346	0.9978883	0.9936224	Complete	Among 19 candidate variants in loci 21, rs10124636 alters CER1 expression in glutamatergic neurons (AlphaGenome quantile scores: RNA-seq = 0.996, DNase = 0.901, ChIP-TF = 0.879). CER1 (Cerberus) is a multivalent growth-factor antagonist in the extracellular space that binds to Nodal, BMP, and Wnt proteins via independent sites. It is secreted by anterior visceral endoderm and blocks the action of BMP4, Xnr1, and Xwnt8 during primitive node stage to allow head region formation. CER1 plays a critical role as an inhibitory molecule in anterior neural induction and somite formation during embryogenesis (partly through BMP-inhibitory mechanism), and regulates Nodal signaling during gastrulation and formation and patterning of the primitive streak.
22	chr10:8752143		0	0	0.080576845	0.99624366	No protein-coding genes	Locus 22 contains 36 candidate variants but no protein-coding genes in the analyzed region. This locus likely harbors regulatory variants affecting long-range gene expression through enhancer or silencer mechanisms in glutamatergic neurons.
23	rs73351651	SORCS3	0.9708846	0.9992165	0.545117	0.7047799	Complete	Among 114 candidate variants in loci 23, rs73351651 alters SORCS3 splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.997, RNA-seq = 0.882, DNase = 0.926, ChIP-TF = 0.968). SORCS3 is an orphan receptor of the VPS10P domain receptor family, a 130 kDa neuronal receptor distinctly expressed in the hippocampus (especially CA1 region) and cortex. It localizes to the postsynaptic density (PSD), and loss of receptor activity abrogates NMDA receptor-dependent and -independent forms of long-term depression (LTD). SORCS3 functionally interacts with PICK1, an adaptor that sorts glutamate receptors at the postsynapse, and is implicated in Alzheimer's disease as part of the VPS10p-D receptors connected to APP proteolytic breakdown into neurotoxic amyloid-Œ≤ peptides.
24	rs7100410	STK32C	0.99987054	0.9983986	0.8072176	0.8566166	Complete	Among 71 candidate variants in loci 24, rs7100410 alters STK32C splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 1.000, RNA-seq = 0.984, DNase = 0.807, ChIP-TF = 0.857). STK32C is a serine/threonine kinase family member with the highest expression in brain cortex (4.6x overexpression) and frontal cortex (9.5x overexpression), as well as fetal brain (6.3x overexpression). It is involved in Sweet Taste Signaling pathway with Gene Ontology annotations for transferase activity and protein tyrosine kinase activity. The gene is thought to be functional in brain due to its high expression levels and has been associated with bladder cancer prognosis and DNA methylation differences in adolescent depression (monozygotic twin studies).
25	rs7104616	METTL15	0	0.9981611	0.3929598	0.4312567	Complete	Among 170 candidate variants in loci 25, rs7104616 alters METTL15 splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.994, RNA-seq = 0.602, DNase = 0.393, ChIP-TF = 0.621). METTL15 is the main N4-methylcytidine (m4C) methyltransferase in human cells responsible for methylation of position C839 in mitochondrial 12S rRNA, localized in mitochondria. Lack of METTL15 results in reduction of mitochondrial de novo protein synthesis and decreased steady-state levels of oxidative phosphorylation system proteins. Human genetics studies suggest that rRNA methylation contributes to brain development and cognition, with related Mettl5 associated with intellectual disability and sleep disturbances, showing predominant roles in neurons and glia for sleep regulation.
26	rs10506971	DUSP6	0	0.850352	0.9986684	0.7631148	Complete	Among 8 candidate variants in loci 26, rs10506971 alters DUSP6 expression in glutamatergic neurons (AlphaGenome quantile scores: RNA-seq = 0.964, DNase = 0.722, ChIP-TF = 0.936). DUSP6 (Dual specificity phosphatase 6) is the most representative ERK-directed phosphatase that acts primarily in the cytoplasmic compartment. Phosphorylation of DUSP6 and dephosphorylation of ERK1/2 create a negative feedback loop to control ERK activity and MAPK signaling, which is vital for cell proliferation and differentiation. DUSP1 and DUSP6 are the most studied in neural cells, with growth factors and neurotrophins (BDNF, NGF) regulating DUSP expression. 2024 research showed DUSP6 overexpression improves memory deficits in 5xFAD Alzheimer's mice, suggesting neuroprotective roles in glutamatergic neurons.
27	rs1626703	MPHOSPH9	0.99999	0.96289337	0.41229036	0.8471262	Complete	Among 209 candidate variants in loci 27, rs1626703 alters MPHOSPH9 splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 1.000, RNA-seq = 0.963, DNase = 0.412, ChIP-TF = 0.847). MPHOSPH9 is an M-phase phosphoprotein that regulates cell cycle and permits disassembly of interphase structures. It is phosphorylated in M-phase and localizes to the Golgi complex in interphase, negatively regulating cilia formation via the CP110-CEP97 complex. Associated with spinocerebellar ataxia 11 and multiple sclerosis, MPHOSPH9 may influence neuronal cell division, differentiation, and synaptic remodeling in excitatory neurons through its roles in cell cycle regulation in neural progenitors and ciliogenesis relevant to neural development.

Supplementary Table 1 (continued).

locus_num	rsID	gene	splice	rna_seq	dnase	chip_tf	status	interpretation
28	rs73209220	PCDH17	0	0.8227062	0.99901325	0.97697836	Complete	Among 61 candidate variants in loci 28, rs73209220 alters PCDH17 expression in glutamatergic neurons (AlphaGenome quantile scores: RNA-seq = 0.866, DNase = 0.291, ChIP-TF = 0.323). PCDH17 (protocadherin-17) is a non-clustered Œ¥2-protocadherin family member enriched along corticobasal ganglia synapses in a zone-specific manner during synaptogenesis. It regulates presynaptic assembly in these synapses, and PCDH17 deficiency causes facilitated presynaptic vesicle accumulation and enhanced synaptic transmission in both excitatory and inhibitory synapses. Mutations in human Œ¥-pcdhs lead to neural disorders including schizophrenia, autism spectrum disorders, epilepsy, intellectual disability, and Fragile-X syndrome, affecting cell adhesion, neurite initiation and outgrowth, axon pathfinding, and synapse formation and stabilization.
29	rs1328816	POU4F1	0	0.8712331	0.23762526	0.9870033	Complete	Among 33 candidate variants in loci 29, rs1328816 alters POU4F1 expression in glutamatergic neurons (AlphaGenome quantile scores: RNA-seq = 0.952, DNase = 0.512, ChIP-TF = 0.738). POU4F1 is a member of the POU-IV class of neural transcription factors expressed in subsets of retinal ganglion cells and involved in the developing sensory nervous system. POU4F2 expression precedes POU4F1 and functions redundantly to regulate retinal ganglion cell differentiation and survival, with Brn3a/Pou4f1 required for specification of RGCs with small dendritic arbors. POU4F1 regulates dorsal root ganglion sensory neuron specification and axonal projection into the spinal cord, with 2024 research showing ISL1 and POU4F1 directly interact to regulate inner ear sensory neuron differentiation and survival.
30	chr14:98094552		0	0	0.9936224	0.99899024	No protein-coding genes	Locus 30 contains 129 candidate variants but no protein-coding genes in the analyzed region. This locus likely harbors regulatory variants affecting long-range gene expression through enhancer or silencer mechanisms in glutamatergic neurons.
31	rs1435746	SEMA6D	0.84039664	0.56112885	0.9111552	0.97946095	Complete	Among 34 candidate variants in loci 31, rs1435746 alters SEMA6D splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.975, RNA-seq = 0.552, DNase = 0.058, ChIP-TF = 0.716). SEMA6D is a novel member of the class 6 semaphorin family mainly expressed in brain and lung, with ubiquitous brain expression from embryonic late stage to adulthood. It shows growth cone collapsing activity on dorsal root ganglion (DRG) neurons in vitro and may be a stop signal for DRG neurons in their target areas. SEMA6D functions via conjunction with receptor plexin A1 and plays important roles in axon guidance during development and neuronal plasticity, with five different splicing variants showing tissue-specific expression patterns.
32	rs2951904	TCF12	0.9383409	0.9976305	0.989182	0.6929783	Complete	Among 46 candidate variants in loci 32, rs2951904 alters TCF12 splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.982, RNA-seq = 0.552, DNase = 0.035, ChIP-TF = 0.681). TCF12 is a basic helix-loop-helix (bHLH) E-protein family member that recognizes the consensus E-box binding site (CANNTG) and activates transcription. Expression of TCF12 is associated with the expansion of precursor cell populations during brain development, with high levels in the proliferating neuroepithelium of neural folds and cephalic mesenchyme. TCF12 expression is restricted to proliferative ventricular zones and down-regulated when neuronal cells undergo final differentiation. TCF12 is implicated in early cell-fate determination of midbrain dopamine neurons, and TCF12 and NeuroD1 cooperatively drive neuronal migration during cortical development.
33	rs952471	HMG20A	0.9845253	0.99951726	0.6807933	0.8793248	Complete	Among 82 candidate variants in loci 33, rs952471 alters HMG20A splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.998, RNA-seq = 0.995, DNase = 0.748, ChIP-TF = 0.823). HMG20A is a high mobility group (HMG) protein and chromatin-associated regulator originally described as iBRAF, an antagonist of the LSD1/Co-REST complex. It overcomes the repressive effects of the neuronal silencer REST and induces the activation of neuronal-specific genes, playing a key role in the initiation of neuronal differentiation. HMG20A potentiates astrocyte survival and reactivates astroglyosis, promoting the survival of hypothalamic neurons, with impacts on the brain/islet axis and its role in glucose homeostasis. It cooperates with the histone reader PHF14 to modulate TGFŒ≤ and Hippo pathways.
34	rs4966009	IGF1R	0.9982064	0.94555736	0.37327558	0.8655608	Complete	Among 113 candidate variants in loci 34, rs4966009 alters IGF1R splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.998, RNA-seq = 0.946, DNase = 0.373, ChIP-TF = 0.866). IGF1R is the insulin-like growth factor 1 receptor (glycoprotein receptor) abundantly expressed throughout the brain, concentrated in neuron-rich areas including the granule cell layers of olfactory bulb, dentate gyrus, and cerebellar cortex. The IGF1/IGF1R system directly modulates neuronal firing and synaptic transmission, with distinct impacts on synaptic plasticity in different brain regions. IGF1 triggers rapid local autocrine IGF1R activation on the same spine and along the stimulated dendrite, regulating the plasticity of activated spines in CA1 pyramidal neurons. IGF1R signaling through PI3K-Akt and Ras-Raf-MAP pathways intervenes in energy allocation, proteostasis, circadian cycles, mood, and cognition.
35	rs4784166	CDH8	0.8608881	0.8111995	0.9983231	0.72725654	Complete	Among 52 candidate variants in loci 35, rs4784166 alters CDH8 splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.953, RNA-seq = 0.692, DNase = 0.988, ChIP-TF = 0.312). CDH8 (cadherin-8) is a type II classical cadherin that mediates calcium-dependent cell-cell adhesion and is expressed in brain, involved in synaptic adhesion, axon outgrowth, and guidance. CDH8 is enriched within striatal projection neurons in medial prefrontal cortex and in striatal medium spiny neurons forming the direct or indirect pathways. Developmental expression peaks at P10 when cortical projections start to form synapses in the striatum, and CDH8 is concentrated at excitatory synapses in dorsal striatum. CDH8 knockdown impairs dendritic arborization and dendrite self-avoidance, and cadherin-8 has been implicated as an autism risk gene candidate.
36	rs11662025	TMEM200C	0.9708846	0.8819146	0.10345244	0.71620387	Complete	Among 10 candidate variants in loci 36, rs11662025 alters TMEM200C splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.971, RNA-seq = 0.882, DNase = 0.103, ChIP-TF = 0.716). TMEM200C is a transmembrane protein with unknown specific function in neurons and limited research available. Expression data suggests presence in neural tissue, but comprehensive functional characterization is needed. The high splice and RNA-seq scores indicate substantial effects on TMEM200C expression and splicing in glutamatergic neurons, representing a high-priority gene for future mechanistic studies.
37	rs4638666	ZNF521	0.99840146	0.8940993	0.91866904	0.93488073	Complete	Among 13 candidate variants in loci 37, rs4638666 alters ZNF521 splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.998, RNA-seq = 0.894, DNase = 0.919, ChIP-TF = 0.935). ZNF521 is a nuclear protein with 30 Kr√ºppel-like zinc fingers that regulates transcription in diverse tissues and organs. It is capable of directing embryonic stem (ES) cells into neural progenitors, with transcripts enriched in early neural lineage of ES cell differentiation. ZNF521 associates with p300 to activate early neural genes (Sox1, Sox3, Pax6) and acts as a molecular switch for the differentiation of neuronal stem cells, promoting the proliferation and differentiation of striatal neurons. Previously termed "early hematopoietic zinc finger protein," it is abundant in leukemia, medulloblastomas, and brain tumors, with significant upregulation in pediatric AML with MLL translocations.
38	rs12970188	DCC	0.9131995	0.9111552	0.99931777	0.9989667	Complete	Among 129 candidate variants in loci 38, rs12970188 alters DCC splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.979, RNA-seq = 0.483, DNase = 0.982, ChIP-TF = 0.854). DCC (Deleted in Colorectal Cancer) is a receptor for netrin-1, a neuronal axon guidance cue involved in determining the direction and extent of cell migration and axonal outgrowth. Netrin-1 binding to DCC induces chemoattraction (which turns to repulsion if another receptor, UNC5, coexists with DCC). The association of netrin-1 with DCC results in homodimerization via its C-terminal P3 motif, recruiting an intracellular signaling complex that leads to Src family kinase activation and cytoskeleton rearrangement. DCC is crucial for neuronal migration in the dorsal spinal cord, and the netrin-1/DCC guidance system is important for dopamine pathway maturation affecting mesocorticolimbic pathways containing dopamine, GABA, and glutamate axons.
39	rs6035865	XRN2	0.99930274	0.8793248	0.29125375	0.71620387	Complete	Among 95 candidate variants in loci 39, rs6035865 alters XRN2 splicing and expression in glutamatergic neurons (AlphaGenome quantile scores: splice = 0.999, RNA-seq = 0.768, DNase = 0.291, ChIP-TF = 0.716). XRN2 is a 5'-3' exoribonuclease distributed in the nucleoplasm and nucleolus that contributes to essential processes in gene expression such as transcription termination and ribosome biogenesis by degrading or trimming various classes of RNA. It accelerates termination by RNA polymerase II (underpinned by CPSF73 activity) and degrades the long non-coding telomeric RNA TERRA. XRN2 is associated with Amyotrophic Lateral Sclerosis 4, Juvenile and Spinocerebellar Ataxia, Autosomal Recessive, With Axonal Neuropathy 2. Related XRN1 is critical for hypothalamic metabolic homeostasis, with forebrain-specific Xrn1 knockout mice exhibiting obesity, leptin resistance, and hyperglycemia (2024 study).

Supplementary Table 2. Per-dataset parameter choices selected by the Scanpy agent across seven single-cell datasets. The agent received the same generic prompt for every dataset; the inferred organism, mitochondrial gene prefix, and marker gene panel were determined directly from data without user specification. Columns: dataset (dataset name); organism_inferred (organism inferred by the agent from gene-symbol case); tissue (tissue or cell-type context); gene_panel_context (post-filtering gene/cell counts observed by the agent); agent_observed_evidence (evidence the agent cited for its organism call); step1_mt_prefix_selected (mitochondrial prefix the agent applied at the QC step); step7_marker_gene_panel (marker genes the agent selected for cell-type annotation); agent_rationale (the agent's recorded justification). All entries are captured verbatim from MCP tool-call logs.

dataset	organism_inferred	tissue	agent_observed_evidence	step1_mt_prefix_selected	step7_marker_gene_panel	agent_rationale
pbmc_1k_v3	Human	PBMC (10x Genomics)	Human gene name pattern	MT-	CD3D, CD3E for T cells; MS4A1 for B cells; CD14 for monocytes; NKG7 for NK cells	The agent inferred the organism from gene name patterns and correctly identified and applied the MT- prefix to label mitochondrial genes; the agent applied canonical immune cell markers
pbmc_1k_v2	Human	PBMC (10x Genomics)	Human gene name pattern	MT-	CD3D, CD3E for T cells; MS4A1 for B cells; CD14 for monocytes; NKG7 for NK cells	The agent inferred the organism from gene name patterns and correctly identified and applied the MT- prefix to label mitochondrial genes; the agent applied canonical immune cell markers
pbmc_1k_protein_v3	Human	PBMC (10x Genomics CITE-seq)	Human gene name pattern	MT-	CD3D, CD3E for T cells; MS4A1 for B cells; CD14 for monocytes; NKG7 for NK cells	The agent inferred the organism from gene name patterns and correctly identified and applied the MT- prefix to label mitochondrial genes; the agent applied canonical immune cell markers
Brain_Tumor_3p_LT	Human	Brain tumor (10x Genomics)	Human gene name pattern	MT-	EGFR, PDGFRA, CDK4 for tumor/glioma cells; GFAP, AQP4 for astrocytes; MBP, MOG for oligodendrocytes; AIF1, CX3CR1, P2RY12 for microglia/tumor-associated macrophages; CD3D, CD3E for infiltrating T cells	The agent inferred the organism from gene name patterns and correctly identified and applied the MT- prefix to label mitochondrial genes; the agent supplemented brain markers with glioma drivers and tumor-associated macrophage markers, reflecting the malignant context
neuron_1k_v3	Mouse	Brain (E18 cortex; 10x Genomics)	Mouse gene nomenclature conventions	mt-	Slc17a7, Slc17a6 for excitatory neurons; Gad1, Gad2 for inhibitory neurons; Gfap, Aqp4 for astrocytes; Mbp, Mog for oligodendrocytes; Pdgfra for oligodendrocyte precursor cells	The agent automatically switched to the mt- prefix, recognizing the organism from gene nomenclature conventions without explicit user specification; the agent automatically switched to brain-specific neural lineage markers
neuron_10k_v3	Mouse	Brain (cortex; 10x Genomics)	Mouse gene nomenclature conventions	mt-	Slc17a7, Slc17a6 for excitatory neurons; Gad1, Gad2 for inhibitory neurons; Gfap for astrocytes; Mbp, Mog, Plp1 for oligodendrocytes; Pdgfra for OPCs	The agent automatically switched to the mt- prefix, recognizing the organism from gene nomenclature conventions without explicit user specification; the agent automatically switched to brain-specific neural lineage markers
heart_1k_v3	Mouse	Heart (10x Genomics)	Mouse gene nomenclature conventions	mt-	Myh6, Myh7, Tnnt2 for cardiomyocytes; Col1a1, Col3a1 for fibroblasts; Pecam1, Cdh5 for endothelial cells; Acta2, Myh11 for smooth muscle/pericytes; Csf1r, Cd68 for cardiac macrophages	The agent automatically switched to the mt- prefix, recognizing the organism from gene nomenclature conventions without explicit user specification; the agent automatically switched to heart-specific cardiac lineage markers