Article

Reimagining research papers as interactive and reliable AI agents

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

Received: 13 October 2025

Accepted: 14 August 2026

Published online: 16 September 2026

Open access

Check for updates

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

Here we introduce Paper2Agent, an automated framework that converts research papers into artificial intelligence (AI) agents. Paper2Agent transforms research output from passive artefacts into active systems that accelerate use and discovery. Conventional research papers require readers to understand and adapt the paper's code, data and methods to their work, creating barriers to dissemination and reuse. Paper2Agent addresses this challenge by converting a paper into an AI agent that functions as a virtual corresponding author, exposing its manuscript, supplementary materials, datasets, code and workflows as active, agent-native knowledge rather than static text. It analyses the paper and codebase using multiple agents to construct a model context protocol (MCP) server, then generates and runs tests to refine and increase robustness of the MCP. These paper MCPs can be connected to a chat agent (such as Claude Code) to carry out complex scientific queries through natural language while invoking tools and workflows from the paper. We demonstrate Paper2Agent's effectiveness through case studies. Paper2Agent created an agent that leveraged AlphaGenome[1] to interpret genomic variants and agents based on Scanpy[2] and TISSUE (transcript imputation with spatial single-cell uncertainty estimation)[3] to conduct single-cell and spatial transcriptomics analyses. We validate that these agents reproduce the results of the original papers and carry out novel user queries. Paper2Agent created multiple agents that collaborate to prioritize a causal gene for psoriasis. By turning static papers into interactive AI agents, Paper2Agent introduces a paradigm for knowledge dissemination and a collaborative ecosystem of AI co-scientists.

The research paper is the traditional unit of scientific communication. It remains the norm for documenting methods, results and insights, and is the primary way in which research is shared with the broader community. However, papers are fundamentally passive objects: a reader must discover the paper (not an easy task given the flood of publications), parse its contributions and manually determine how to apply them to their own work. In particular, when a paper describes a new computational method, substantial technical barriers often remain before the method can be used on new data[4]. A reader might need to locate the corresponding code repository, install dependencies, configure environments and interpret the correct inputs and outputs[5]. Even with well-maintained repositories, this process is often non-trivial.

For instance, consider AlphaGenome, which provides a powerful framework for genome-scale foundation modelling[1]. Despite its utility, this system requires substantial technical expertise to set up and deploy, limiting accessibility for biologists who could otherwise benefit. Using AlphaGenome in code involves installing the environment, creating client objects with application programming interface (API) keys, constructing inputs such as variant objects and selecting desired output modalities. Users must understand the API hierarchy and parameter semantics, which imposes a learning curve for biologists unfamiliar with these abstractions.

This illustrates a broader challenge: research outputs are passively siloed behind technical barriers. Paper2Agent reimagines research dissemination by turning static papers into active AI agents. Each agent serves as an interactive expert on the corresponding paper, capable of demonstrating, applying and adapting its methods to new projects.

AI agents are autonomous systems that can reason about tasks and act to achieve goals by leveraging external tools and resources[6]. Modern AI agents are typically powered by large language models (LLMs) connected to external tools or APIs, and can adapt based on feedback[7]. Notably, because agents are built on top of LLMs, users can interact with agents through human language, substantially reducing usage barriers for scientists. A range of recent systems, including general-purpose scientist agents[8–12] and domain-specialized agents[13,14], have begun to demonstrate the potential of this paradigm, alongside efforts to automatically generate code from scientific text[15,16]. 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.

[1]Department of Genetics, Stanford University, Stanford, CA, USA. [2]Department of Biomedical Data Science, Stanford University, Stanford, CA, USA. [3]Department of Electrical Engineering, Stanford University, Stanford, CA, USA. [4]Department of Biology, Stanford University, Stanford, CA, USA. [5]Department of Computer Science, Stanford University, Stanford, CA, USA. ✉e-mail: jcmiao@stanford.edu; jamesz@stanford.edu

Paper2Agent provides an automated workflow for converting a scientific paper into an agent. The core idea is to represent the paper as an MCP server[17]. MCP is a standardized protocol that allows structured APIs and tools to be exposed in a way that is directly accessible to LLMs and agent frameworks. The conversion process identifies a paper's key contributions, encapsulates them through an MCP server and links the server to LLM-based agents for natural language querying and autonomous execution. Users can then interact with the paper by asking questions, requesting demonstrations, or applying the method to new data. As an illustration, applying Paper2Agent to AlphaGenome would expose its genome foundation model as an MCP, so that instead of cloning repositories and configuring dependencies, a user could simply ask: 'Interpret the expected effect of this variant on chromatin accessibility in muscle cells'.

Earlier efforts have sought to make research outputs more executable and accessible, including executable papers[18,19], the Papers with Code initiative[20], containerized artefact platforms such as Binder[21] and CodeOcean[22], and paper-to-code systems[15]. Although these efforts improved reproducibility, substantial barriers remained for understanding, customizing and applying the code to new projects.

Paper2Agent substantially extends this trajectory by providing a new framework: a paper can be transformed into a capable agent that is accessible via natural language. In contrast to previous efforts, Paper2Agent shifts the research output from a document or codebase encoding knowledge to a knowledgeable entity capable of execution and dialogue. This extends beyond retrieval-augmented generation over paper text. Paper2Agent agentifies the full research outputs, including manuscripts, supplementary materials, code, datasets, executable examples and analysis workflows. In this framework, a paper becomes an executable research artefact that can answer questions, reproduce analyses, apply methods to new data and interoperate with other paper agents. This represents a new mode of scientific communication, moving beyond static dissemination to interactive collaboration.

Overview of Paper2Agent

Paper2Agent is a multi-agent AI system that automatically transforms research papers into interactive AI agents with minimal human input. The paper agents created via this framework are:

1. Interactive and easy to use. Users can execute complex scientific analyses through natural language prompts, eliminating the need for programming expertise.
2. Reliable and reproducible. Each tool used by a paper agent is validated against the reference codebase's reported results and figures using example datasets, and then locked to ensure reproducibility. This design mitigates the risk of 'code hallucination', where executing inaccurate LLM-generated code could lead to incorrect scientific results. It also minimizes randomness in code generation, further strengthening reproducibility. Finally, every tool includes a code reference from the original paper to provide transparency and traceability.

MCP has recently become an industry standard for connecting LLM-based agents with external resources, providing a unified interface for accessing datasets and tools without custom integration[17]. Paper2Agent builds on this ecosystem with two components: (1) Paper2MCP, which extracts information from papers and their codebases to build remote MCP servers; and (2) an agent layer, which wraps each MCP server as a context provider to instantiate paper-specific AI agents (Fig. 1a). Any LLM or external agent can invoke the servers' tools through MCP without extra setup. For clarity of presentation, we assign one MCP server and one paper agent to each paper. The same approach can create MCPs and agents for a group of related papers. Each MCP server includes three core components:

1. MCP tools are executable functions that encapsulate a paper's methodological contributions. For example, one AlphaGenome MCP tool takes a genetic variant as input and generates predictions and visualizations of its effects on gene expression, chromatin accessibility and other modalities. These tools come with a pre-configured environment for seamless execution.
2. MCP resources serve as a repository of static assets, including the manuscript text, the associated codebase and supplementary materials such as datasets, tables and figures. As an illustration, the AlphaGenome MCP resources include links to the training data used to train the model. All resources are stored in accessible, standardized formats to enable efficient querying and integration by AI agents.
3. MCP prompts contain concise instructions that guide AI agents through complex, multi-step scientific workflows derived from a paper's text or codebase. For example, a Scanpy MCP prompt encodes the sequence of steps for preprocessing and clustering single-cell data, which we present later in the manuscript. These templates orchestrate tools and resources to ensure reproducible, systematic analyses while reducing the barriers to effective prompting.

The paper MCP servers can be hosted remotely on platforms such as Hugging Face Spaces (https://huggingface.co/spaces), eliminating local dependency issues. MCP standardizes communication, enabling secure and scalable integration with AI agents. The agent layer wraps each Paper2MCP server as a context provider, creating paper-specific conversational agents. Any compatible LLM or agent can connect to these servers to perform tasks such as reproducibility checks, new data analyses or figure regeneration. Here we use Claude Sonnet 4 for all of the Paper2Agent applications. For example, a user might ask 'Apply the method in this paper to the newly generated dataset', and the agent will automatically run the pipeline, produce results and present interpretable outputs. By abstracting away technical details, the agent lowers barriers to method adoption, ensures reproducibility and helps researchers focus on insights rather than implementation.

We implemented Paper2Agent with Claude Code[23], an AI coding agent specialized in managing complex coding tasks. The workflow begins by identifying the codebase associated with a paper (Fig. 1b). Two specialized agents are then invoked: the environment agent, which configures the necessary software environment, and the extraction agent, which translates core methods into implemented tools. These tools are validated through a testing agent that runs automated checks, refining both the code and environment until results match the reference outputs. A test passes when expected files are generated, numerical results fall within tolerance thresholds and figures match references; tools that repeatedly fail validation are excluded from the final MCP server. Once validated, the tools and environment are packaged into an MCP Python file that can be deployed on a remote server such as Hugging Face. Finally, the paper MCP server is connected with an AI agent to create a fully functional paper agent, enabling interactive access to the paper's knowledge and method through natural language queries. We use Claude Code as the downstream AI agent in our case studies, although the paper MCPs can be flexibly integrated with different chat agents. Because MCPs are modular, multiple MCPs can be connected to the same chat agent, enabling users to leverage tools and resources across multiple papers simultaneously.

Next, we present case studies demonstrating Paper2Agent's ability to convert diverse research papers into reliable, interactive AI agents for different scientific tasks.

AlphaGenome agent for genomics

Our first case study showcases the AlphaGenome agent. AlphaGenome is an AI model designed to predict the effects of single-nucleotide variants or mutations in human DNA sequences on a wide range of regulatory processes[1]. Paper2Agent transforms the AlphaGenome paper into

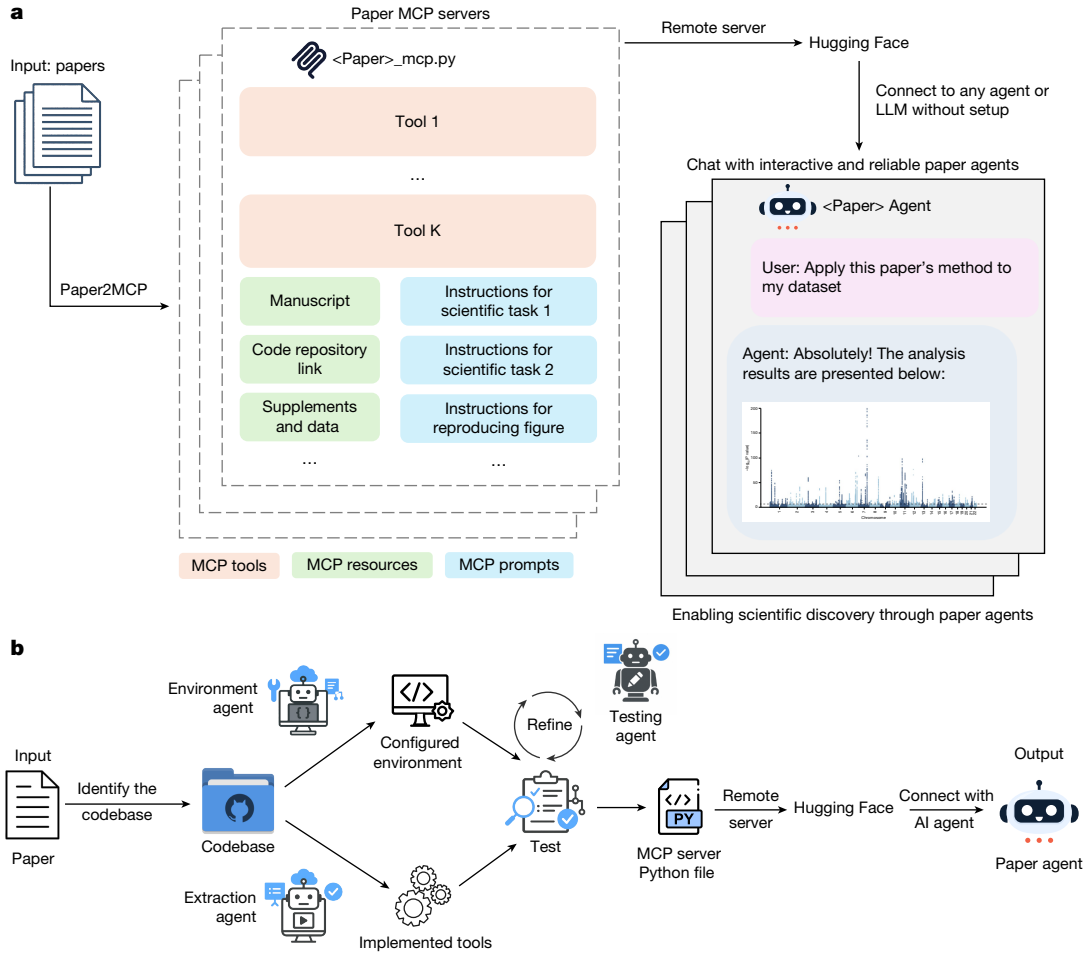


Fig. 1 | Overview of Paper2Agent. a, Paper2Agent turns research papers into interactive AI agents by building remote MCP servers with tools, resources and prompts. Connecting an AI agent to the server creates a paper-specific agent for diverse tasks. **b**, Workflow of Paper2Agent. It starts with codebase extraction and automated environment setup for reproducibility. Core analytical features are wrapped as MCP tools, then validated through iterative testing. The resulting MCP server is deployed remotely and integrated with an AI agent, enabling natural language interaction with the paper's methods and analyses.

an interactive AlphaGenome agent, enabling automated interpretation of genomics data. Through natural language queries, users can leverage this agent to prioritize causal genes for disease-associated variants, clarify the regulatory impact of individual variants and inform the design of synthetic DNA with specific regulatory functions.

Paper2Agent generated 22 AlphaGenome MCP tools, all of which passed automated validation, in around 45 min, costing US $14 on a personal laptop without human intervention, comprehensively covering its methodological innovations. This one-time process produced reusable tools for future applications. These MCP tools span single- and batch-variant scoring across functional assays, sequence-level prediction, tissue ontology exploration, and an extensive visualization suite (Fig. 2a). For example, score_variant_effect() is an MCP tool that predicts the functional consequences of genetic variants across multiple modalities—such as gene expression, splicing and chromatin accessibility—within a wide range of tissues and cell types. Complementing this, visualize_variant_effects() generates modality-specific visualizations that simplify the interpretation of regulatory effects.

Of note, the tools generated by Paper2Agent are designed with flexible, well-annotated input parameters. For example, the visualize_variant_effects() tool exposes a rich set of options that make it adaptable to diverse use cases (Supplementary Fig. 1). Given an input genetic variant, the AlphaGenome agent can adjust the sequence context length around the variant, and toggle different modalities—such as RNA sequencing (RNA-seq), assay for transposase-accessible chromatin with sequencing (ATAC-seq) or chromatin immunoprecipitation with sequencing (ChIP-seq) histone tracks. Moreover, each MCP tool embeds a traceable link to the original source code, ensuring transparency and reproducibility. By connecting an AI agent with the AlphaGenome MCP, the system creates the AlphaGenome agent.

Next, we benchmarked the AlphaGenome agent against human-executed ground truth, Claude Code with direct repository access (Claude + Repo) and Biomni (https://biomni.stanford.edu/) (Fig. 2b). The Paper2Agent-generated agent did not have access to the manuscript or raw repository during evaluation, and results were graded by two independent human experts using predefined rubrics (inter-rater agreement: 96.7%). Benchmark queries included tutorial-derived tasks 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 novel tasks such as 'Analyze variant chr9:98765432:T>C with DNASE predictions for muscle cells (CL:0000187). What is the quantile_score for muscle tissue?'. Across five independent runs, it achieved 98.7 ± 1.3% accuracy on 15 tutorial-derived queries and 100.0 ± 0.0% accuracy on 15 novel queries, outperforming Claude + Repo (82.7 ± 3.4% and 78.7 ± 4.4%) and Biomni (37.3 ± 4.0% and 56.0 ± 3.4%). On 30 open-ended researcher-style queries requiring multi-step tool composition and biological synthesis, 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?', Paper2Agent achieved 82.7 ± 2.4% accuracy, compared with 56.7 ± 2.3% for Claude + Repo and 72.2 ± 2.2% for Biomni. These gains

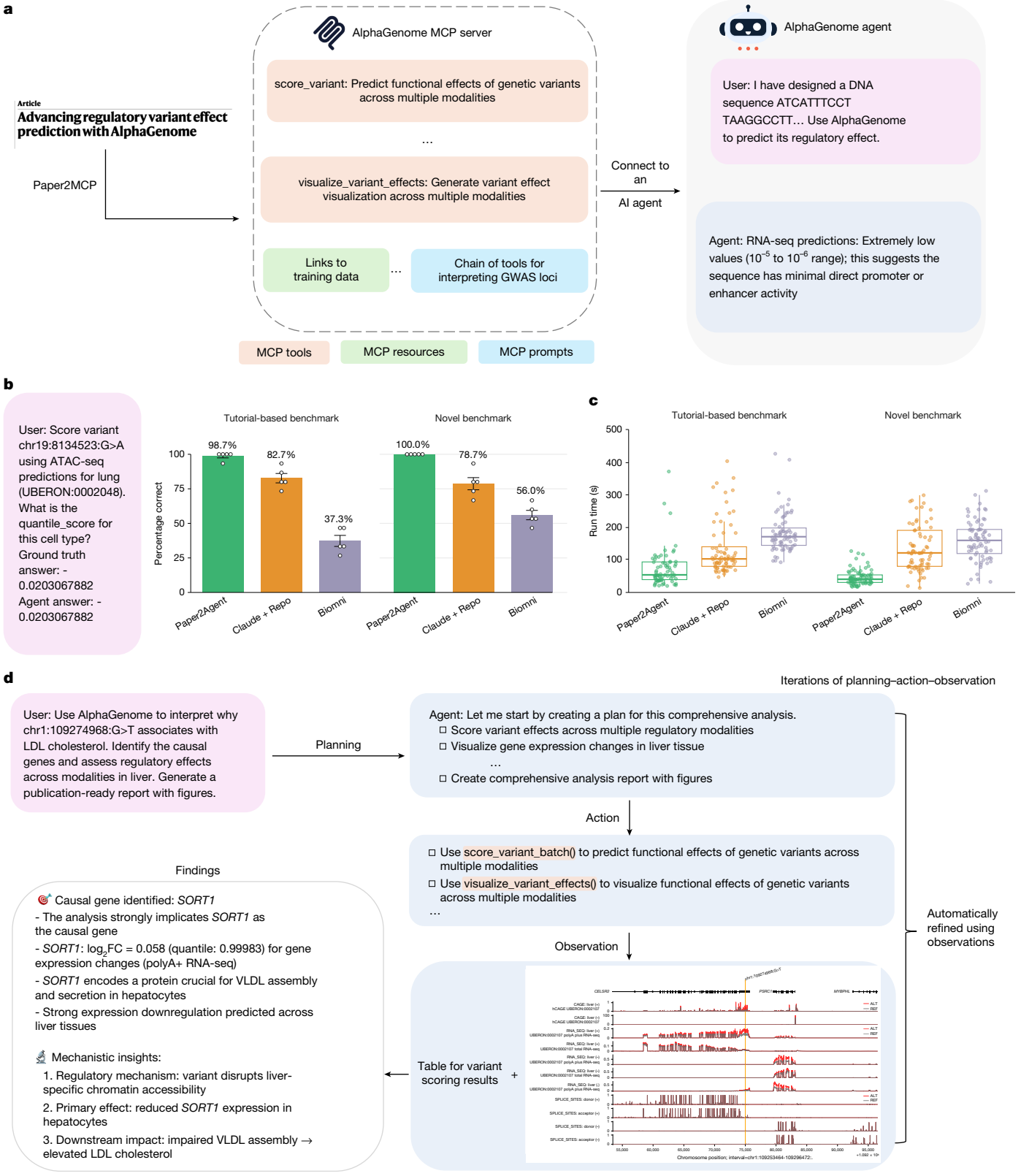


Fig. 2 | Overview of the Paper2Agent-generated AlphaGenome agent. **a**, Construction of the AlphaGenome MCP server and agent. **b**, Benchmark accuracy. Data are mean ± s.e.m.; $n = 5$ independent runs. Dots represent individual runs. **c**, Query run times. Centre lines show medians, boxes delineate the 25th–75th percentiles, whiskers extend to the most extreme values within 1.5× the interquartile range and dots beyond the whiskers represent outliers. $n = 75$ query runs per method per benchmark (15 queries across 5 independent runs). **d**, Automated planning and interpretation of GWAS loci through iterative planning–action–observation cycles.

were robust to prompt paraphrasing and persisted when the Claude + Repo baseline was upgraded to newer models such as Claude Opus 4.6 (Supplementary Figs. 2 and 3 and Supplementary Note). The AlphaGenome agent also reduces median runtime by 1.9× and 3.1× relative to Claude + Repo and Biomni, respectively, on tutorial-derived queries, and by 2.9× and 3.8× on novel queries (Fig. 2c). These results indicate that Paper2Agent improves both reliability and efficiency relative to using a general-purpose agent directly on the repository.

Finally, we demonstrated that the AlphaGenome agent enables automatic interpretation of genome-wide association study (GWAS) loci and validation of the analysis in the original paper. We considered the example of interpreting why the genetic variant chr1:109274968:G>T is associated with low-density lipoprotein (LDL) cholesterol that was presented in the original AlphaGenome paper (Fig. 2d). Based on the tools available, the AlphaGenome agent constructs a step-by-step plan to solve this task. This plan includes generating input files, scoring variants across multiple modalities, filtering results for trait-relevant tissues, creating modality-specific visualizations and assembling an interpretation report. The agent then executes these actions using implemented tools, automatically refining its strategy through iterative observation and feedback. A final report is then presented to provide a unified interpretation of the regulatory impact of the variant, integrating evidence across modalities and tissues.

Interestingly, the AlphaGenome agent prioritizes *SORT1* as the most likely causal gene, whereas the original paper emphasized *CELSR2* and *PSRC1*. The agent favours *SORT1* for two reasons: (1) a high quantile score (0.99983) indicating a strong predicted impact on *SORT1* expression in liver tissue. Here, the quantile score reflects how extreme the predicted effect of the variant is relative to other variants; (2) *SORT1* encodes sortilin, directly involved in LDL and very low-density lipoprotein (VLDL) secretion[24]. We queried the GTEx expression quantitative trait loci (eQTL) data and confirmed that this variant is a significant eQTL for *SORT1* ($P = 1.1 \times 10^{-65}$) in liver[25]. However, both *CELSR2* and *PSRC1* also exhibit high AlphaGenome quantile scores (0.99998 each) and significant eQTL associations in GTEx liver ($P = 4.7 \times 10^{-46}$ and 8.5×10^{-50}, respectively). This result shows the inherent difficulty in confidently assigning causal genes at complex GWAS loci where the variants are eQTLs for multiple nearby genes[26,27].

This discrepancy highlights a key strength of Paper2Agent: with a single prompt, users can re-evaluate published conclusions using independent model-based evidence, without the need to design new analysis pipelines. Rather than treating the original interpretation as fixed, the agent enables dynamic hypothesis re-assessment and, at scale, provides a systematic way to revisit conclusions across many studies.

Scanpy agent for single-cell analysis

Next, we demonstrate the application of the Paper2Agent-generated Scanpy agent for single-cell data analysis. Scanpy is a widely used package for analysing large-scale single-cell transcriptomic data[2]. We focus on Scanpy's most common use case: preprocessing and clustering single-cell data. Paper2Agent generated seven tools for this feature, all of which passed automated validation, in around 45 min costing US $13 on a personal laptop. The resulting tools included quality_control() for calculating and visualizing quality control metrics, filtering cells and genes, and detecting doublets (Fig. 3a). This enables users to prompt the Scanpy agent to perform quality control on their single-cell data.

In practice, many users prefer an end-to-end workflow for preprocessing and clustering, in which the implemented tools are executed sequentially in the correct order. This type of analysis workflow is not unique to single-cell analysis but is common across many scientific domains. However, executing such workflows can be challenging: the AI agent must either already 'know' the correct order of actions, or the user must provide a carefully structured prompt that explicitly specifies the sequence. To overcome this limitation, we use MCP prompts to guide the agent. MCP prompts offer a standardized way to encode workflows, ensuring that tools are executed in the proper order and relieving users from the burden of manually instructing the agent. Notably, these MCP prompts are inferred directly from the paper and codebase by Paper2Agent, without the need for manual curation. This design improves both reproducibility and usability, particularly for complex analyses such as single-cell data processing.

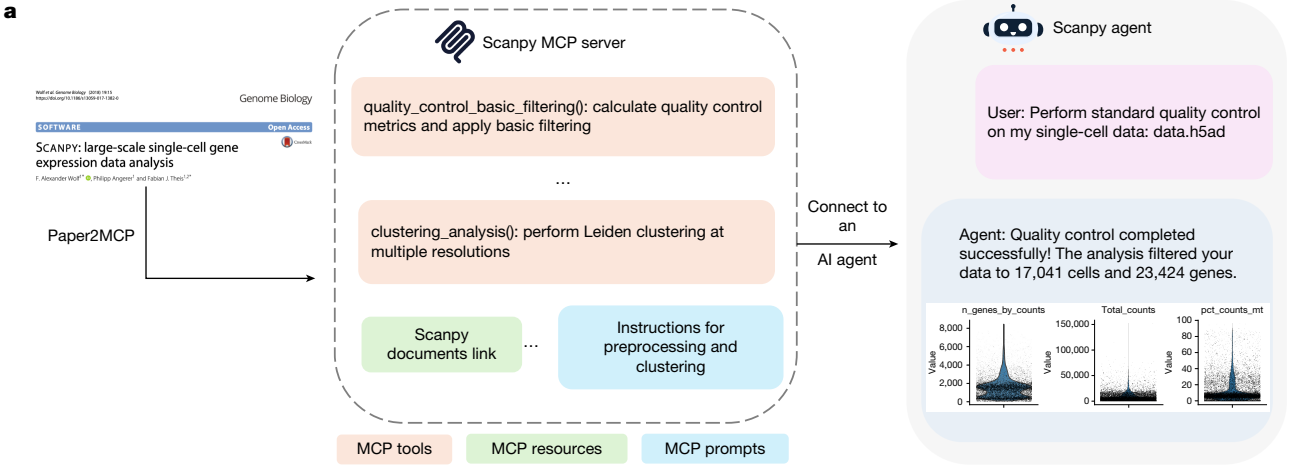
For example, the Paper2Agent-generated Scanpy MCP prompts encode a standard preprocessing and clustering pipeline, including quality control, normalization, feature selection, dimensionality reduction, graph construction, clustering and cell-type annotation in the correct order (Fig. 3b). The prompt also instructs the Scanpy agent to inspect the data before analysis to select appropriate parameters. Users only need to provide the data path (in this example, data.h5ad), and the Scanpy agent automatically runs the workflow and provides a summary of the analysis results.

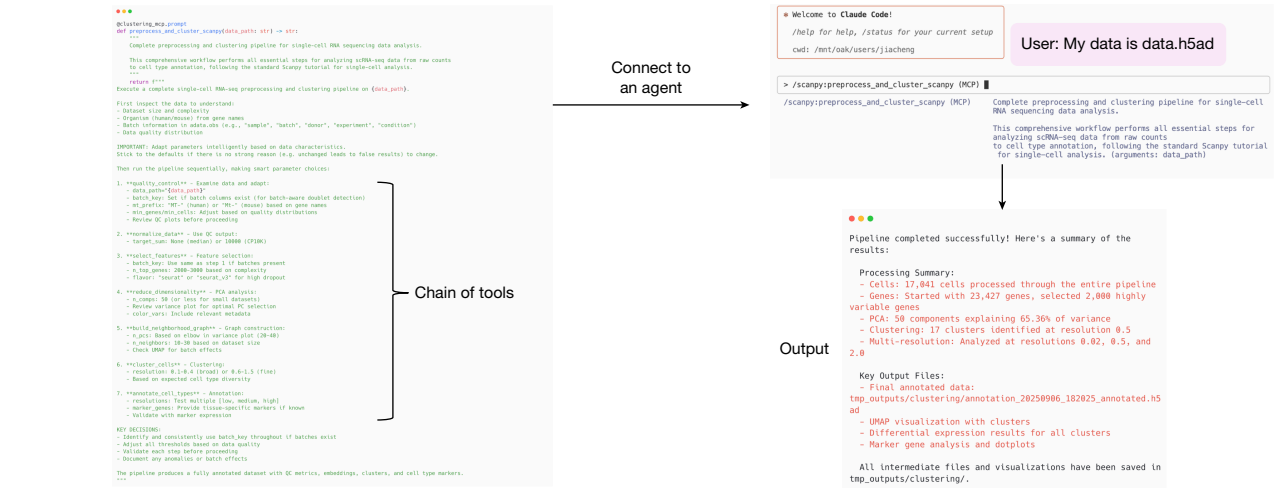
To evaluate the performance of the Scanpy agent, we applied it to preprocess and cluster four publicly available single-cell datasets (Data availability) that are not included in the Scanpy codebase. We invoked the Scanpy MCP prompts and queried the Scanpy agent 'Perform standard single-cell preprocessing and clustering pipeline on this single-cell data: data.h5ad'. As shown in Fig. 3c, the agent produced outputs that match those produced by human researchers when processing the same data, retaining equivalent cell and gene counts after quality control and recovering equivalent top differentially expressed marker genes per cluster with matched parameters. To assess generalizability, we applied it to seven diverse single-cell datasets and observed that the agent adaptively adjusts parameters based on data characteristics (Supplementary Table 2). This demonstrates how MCP prompt-powered Scanpy agents streamline workflow execution, making advanced single-cell analysis both accessible and reproducible. We further present a case study in which we agentified the TISSUE paper[3] for single-cell spatial transcriptomics analysis (Extended Data Fig. 1 and Supplementary Note).

Large-scale evaluation of Paper2Agent

To test scalability, we processed three heterogeneous paper corpora end-to-end without manual cleanup, code modification or intervention: 100 computational biology papers, 26 data- and discovery-focused papers, and 10 non-biology computational papers spanning AI, statistics, econometrics, game theory and astrophysics[28–37] (Supplementary Note). Paper2Agent exposes each paper through a structured resource layer and, when the associated codebase permits, an executable MCP tool layer.

Among the 100 computational biology papers, 74 were successfully agentified, yielding 599 proposed tools, of which 593 passed automated validation. The major failure modes among the remaining papers were missing executable code, missing data or model artefacts, environment or dependency failures, and non-generalizable scripts. On 300 tutorial-derived benchmark questions, Paper2Agent with Sonnet 4 achieved 91.2 ± 1.6% accuracy, outperforming Claude Code with direct repository access using Sonnet 4 (80.3 ± 2.3%; $P < 0.0001$) and Sonnet 4.6 (86.3 ± 1.1%; $P < 0.0001$). Paper2Agent also reduced query cost and latency (US $0.20 and 1.6 min per query, compared with US $0.38 and 4.3 min for using Sonnet 4 directly with the paper and repo). Across 42 execution-based tasks from 10 non-biology computational papers, Paper2Agent achieved 98.1 ± 0.8% accuracy across 5 independent runs, demonstrating generalization beyond computational biology. Paper2Agent also remained useful when executable tools could not be constructed. Across 26 data- and discovery-focused papers, the resource layer achieved 89.0 ± 3.1% accuracy on 100 synthesis-based questions, outperforming a Claude browser-use baseline (82.0 ± 3.8%; $P = 0.03$) while being 34× cheaper and 15× faster.





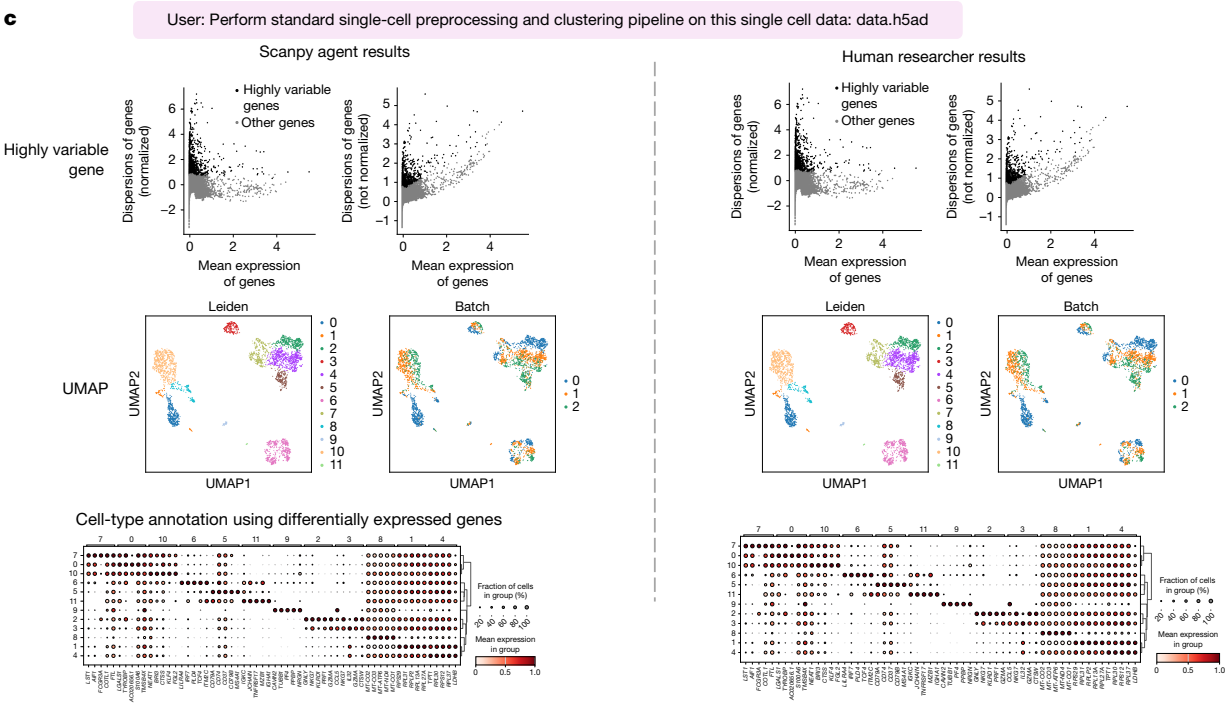


Fig. 3 | Overview of the Paper2Agent-generated Scanpy agent. a, Construction of the Scanpy MCP server and agent. **b**, MCP prompts encode a standardized single-cell preprocessing and clustering pipeline. **c**, Agent reproduces human researcher results, requiring only the dataset path as input. UMAP, uniform manifold approximation and projection.

Additional analyses demonstrated no systematic evidence of shortcut learning in generated MCPs, showed that Paper2Agent supports model training and adaptive hyperparameter tuning in examples, and found that Paper2Agent correctly rejected 100% of out-of-scope queries in a permuted paper–question benchmark. In adversarial repository-drift tests, Paper2Agent recovered functional MCP servers across injected dependency, file-path, typo and deprecated API failures. Paper2Agent also generated and validated a functional MCP from a repository without executable tutorials, demonstrating that curated tutorials are beneficial but not strictly required. Ablations further showed that automated validation and the multi-agent design contribute to performance (Supplementary Note).

Together, these results show that Paper2Agent can transform heterogeneous papers into useful queryable agents at scale, with executable tools when code is agentifiable and structured resources when it is not.

Paper agents collaborate for discovery

Human scientific collaboration often advances by combining insights from multiple, disparate papers—for instance, when a newly developed method is applied to a recently published dataset to generate new discoveries. However, this process is typically slow and labour-intensive. Paper2Agent enables a new mode of AI-driven collaboration in which AI paper agents can interact directly with each other. The agent of a new method paper can autonomously collaborate with the agent of a new data paper to perform analyses, test hypotheses and generate new insights.

Identifying the disease-relevant target gene at a GWAS locus is a long-standing problem in human genetics, because a single regulatory variant can affect multiple nearby genes, and disentangling them requires evidence beyond the original locus mapping[38]. As a discovery case study, we used three agents generated by Paper2Agent: AlphaGenome[1], massively parallel reporter assay (MPRA)-coupled single-cell CRISPR interference (scCRISPRi)[39], and Perturb-seq of $CD4^+$ T cells[40], to identify and validate the causal gene for a psoriasis-associated variant. The AlphaGenome agent predicted *GPR137* as the top affected gene at the rs887314 locus in $CD4^+$ T cells (RNA-seq quantile score = 0.997), ranking above other nearby genes. To validate this prediction, we instructed the AI co-scientist to cross-reference the AlphaGenome prediction with experimental data from two additional paper agents: an MPRA-coupled scCRISPRi screen that measured downstream gene expression changes upon *cis*-regulatory element (CRE) perturbation, and a genome-wide Perturb-seq dataset that profiled transcriptional consequences of gene knockdowns in primary human $CD4^+$ T cells.

After autonomously inspecting the manuscript, analysing the supplementary tables from the MPRA-coupled scCRISPRi paper[39] and examining the differential expression summary statistics from the $CD4^+$ T cell Perturb-seq paper[40], the AI co-scientist proposed ten candidate strategies to validate this gene (Supplementary Note). From these, the human researcher selected signature-correlation analysis to test whether the CRE perturbation signature matched any gene-knockdown signatures. For each of the five top-ranked candidate genes with available knockdown data, the agent correlated the downstream gene expression changes caused by perturbing the rs887314 CRE with those caused by knocking down each candidate gene across three culture conditions (Rest, Stim8hr and Stim48hr; Methods). Only *GPR137* knockdown showed significant concordance with the CRE perturbation signature under stimulated conditions (Spearman correlation = 0.613, $P = 3.79 \times 10^{-3}$ at Stim8hr; Spearman correlation = 0.630, $P = 4.71 \times 10^{-3}$ at Stim48hr; false discovery rate (FDR) < 0.05), whereas *BAD* knockdown and three other top-ranked candidate genes showed no significant correlation in any condition (Fig. 4b and Supplementary Fig. 4). These results demonstrate that Paper2Agent enables the autonomous integration of computational predictions with independent experimental validation, supporting *GPR137* as the probable causal gene for psoriasis. Notably, the *GPR137* knockdown effect reached significance only under stimulation and not at rest (Spearman correlation = 0.29, $P = 0.21$), so we interpret the role of *GPR137* at this locus as activation-dependent. This pattern is consistent with the finding from the Perturb-seq study that regulators of $CD4^+$ T cells vary substantially in their effects across stimulation conditions[40] and aligns with the established role of activated $CD4^+$ T cells in psoriasis pathogenesis[41,42].

The agent's proposed strategy is itself a new data integration approach. Rather than relying on a single readout, the agent integrated the two complementary datasets by correlating CRE perturbation signatures (from scCRISPRi) with gene-knockdown signatures (from Perturb-seq). This cross-screen, cross-modality signature-correlation procedure is not proposed in the source papers and is a flexible technique for transferring causal-gene prioritization across independent perturbation datasets at GWAS loci with multiple co-regulated candidates.

In the Supplementary Note, we present an additional case study in which Paper2Agent connects the AlphaGenome method paper[1] with a recent GWAS of attention-deficit hyperactivity disorder (ADHD)[43], enabling an AI co-scientist to prioritize candidate causal variants and mechanisms across ADHD loci (Extended Data Fig. 2). Together, these results exemplify a new collaborative paradigm in which human scientists use Paper2Agent to design their own AI co-scientists and jointly formulate high-level scientific hypotheses, while the AI co-scientists autonomously execute and interpret complex analytical tasks.

Discussion

Here we introduce Paper2Agent, a framework that transforms a research paper from a passive publication into an interactive AI agent. We demonstrate this approach by creating Paper2Agent instances for several methodological advances. These examples illustrate how a paper agent can embody the research contribution, making it directly accessible through natural language interaction. The generated paper MCPs are modular units that can be connected to diverse user-facing agents, enabling broad adoption. By lowering the barriers between publication and practical application, Paper2Agent helps bridge the gap between how scientific discoveries are disseminated and how they are used in practice. More broadly, Paper2Agent advocates a conceptual shift in how scientific knowledge is represented and reused: papers become agent-native research objects rather than static documents.

Not every paper can be seamlessly turned into a robust agent. As we demonstrate in the adversarial benchmarks, the iterative execute–diagnose–repair loop of Paper2Agent can detect and correct many inherited bugs that manifest as execution failures. Nevertheless, in our large-scale evaluation, a substantial fraction of repositories still could not be successfully agentified, typically owing to incomplete codebases, missing documentation or unresolvable environment configurations. These challenges suggest that the ease with which a paper can be transformed into an agent may itself serve as a practical measure of reproducibility. Just as the scientific community has come to expect clear data and code availability, we envision a natural extension: expecting contributions to be structured in agent-native artefacts that facilitate their translation into agents. Well-documented and transparent papers will naturally lend themselves to this new standard. We emphasize that open-ended scientific reasoning, including hypothesis generation and mechanistic interpretation, remains human-in-the-loop.

Paper2Agent can generate hypotheses, propose validation strategies, and execute analyses at scale, but researchers remain responsible for selecting directions and evaluating evidence. We therefore view Paper2Agent as a tool for augmenting scientific discovery and improving access, reproducibility and reuse of papers, rather than as an autonomous or authoritative source of scientific conclusions. For open-ended analyses, however, multiple answers may be defensible, so benchmarks based on agreement with a single reference should be interpreted primarily as measures of faithful execution rather than of

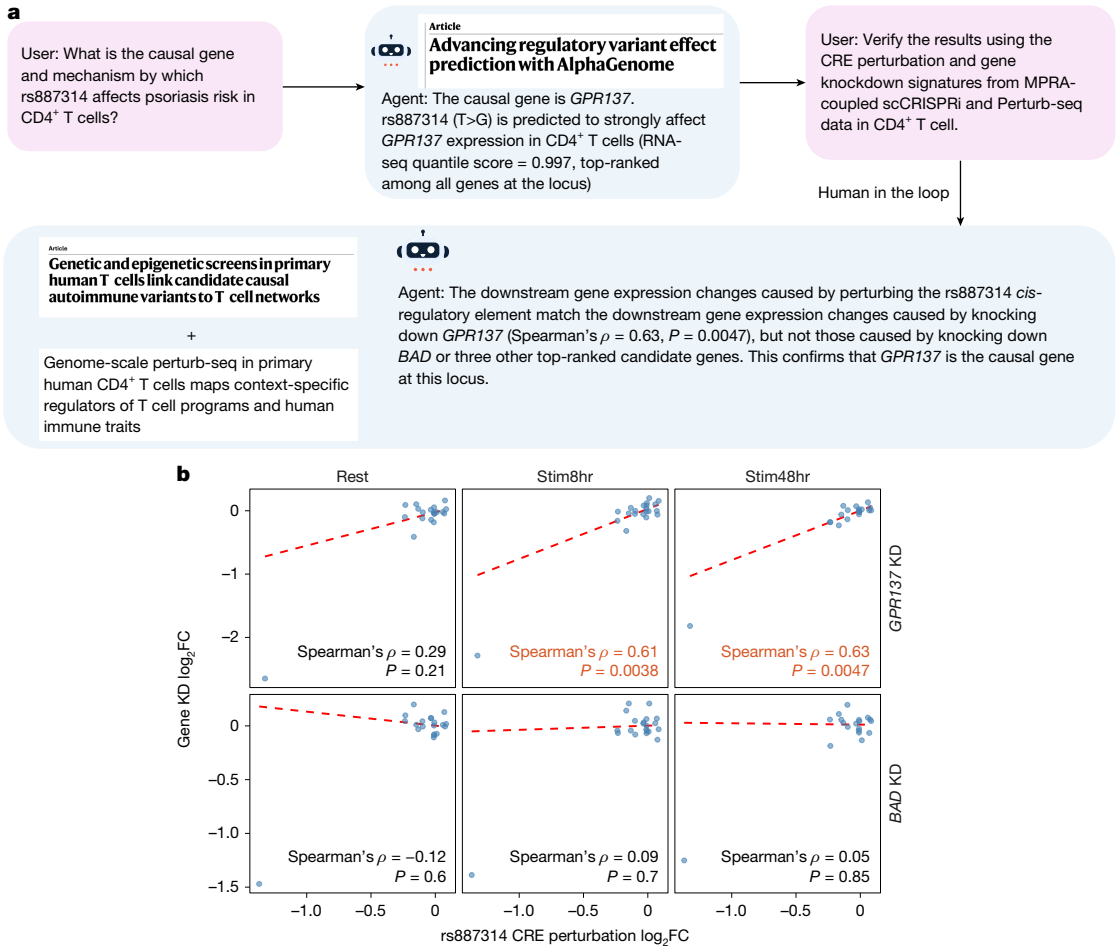


Fig. 4 | Paper2Agent enables prioritization and experimental support for a psoriasis-associated causal gene. a, Paper2Agent integrates AlphaGenome, MPRA-coupled scCRISPRi and Perturb-seq paper agents to prioritize *GPR137* at the psoriasis-associated rs887314 locus. **b**, Correlations between rs887314 CRE perturbation and gene knockdown (KD) signatures. Each dot represents one downstream gene. $n = 20$ (Rest); $n = 21$ (Stim8hr); $n = 19$ (Stim48hr). Spearman's ρ and two-sided P values are shown; Benjamini–Hochberg correction (orange, FDR < 0.05). Dashed lines show linear fits after outlier removal. The perturbed gene was excluded.

analytical validity. Evaluating the range of defensible answers will be important for assessing AI systems on open-ended scientific tasks.

Looking forward, just as many journals now require data and code availability sections, we anticipate the emergence of an 'agent availability' section that specifies whether and how the contribution has been embodied as an interactive agent. This would not only provide immediate utility to readers but also incentivize authors to present their work in a form conducive to agentification. Under this framing, agent-native artefacts become part of what authors publish and maintain alongside their papers, on par with code repositories and example notebooks. Similar to these existing artefacts, paper agents require ongoing maintenance as upstream codebases and dependencies evolve; we view this as an inherent feature of publishing executable research rather than a reason to forgo agentification. Exposing research code as executable agents also introduces security, intellectual property and attribution challenges that warrant careful handling (Supplementary Note).

Finally, once scientific knowledge is encoded in active agents rather than static artefacts, the potential extends beyond individual use. Agents could interact with one another, linking methods to datasets or combining insights from different domains. Communities of such agents could form a dynamic layer of scientific intelligence, accelerating connections across disciplines and enabling a new form of AI-driven collaboration. Paper2Agent thus points towards a future in which scientific communication is not only about describing results, but also about creating interactive, collaborative entities that embody and extend the research.

Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41586-026-11044-y.

1. Avsec, Ž et al. Advancing regulatory variant effect prediction with AlphaGenome. *Nature* **649**, 1206–1218 (2026).
2. Wolf, F. A., Angerer, P. & Theis, F. J. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol.* **19**, 15 (2018).
3. Sun, E. D., Ma, R., Navarro Negredo, P., Brunet, A. & Zou, J. TISSUE: uncertainty-calibrated prediction of single-cell spatial transcriptomics improves downstream analyses. *Nat. Methods* **21**, 444–454 (2024).
4. Trisovic, A., Lau, M. K., Pasquier, T. & Crosas, M. A large-scale study on research code quality and execution. *Sci. Data* **9**, 60 (2022).
5. Gomes, D. G. et al. Why don't we share data and code? Perceived barriers and benefits to public archiving practices. *Proc. R. Soc. B* **289**, 20221113 (2022).
6. Yao, S. et al. ReAct: synergizing reasoning and acting in language models. In *Proc. 11th International Conference on Learning Representations* (ICLR, 2023).
7. Yuksekgonul, M. et al. Optimizing generative ai by backpropagating language model feedback. *Nature* **639**, 609–616 (2025).
8. Lu, C. et al. Towards end-to-end automation of AI research. *Nature* **651**, 914–919 (2026).
9. Swanson, K., Wu, W., Bulaong, N. L., Pak, J. E. & Zou, J. The virtual Lab of AI agents designs new SARS-CoV-2 nanobodies. *Nature* **646**, 716–723 (2025).

10. Gottweis, J. et al. Accelerating scientific discovery with Co-Scientist. *Nature* **655**, 487–496 (2026).
11. Ghareeb, A. E. et al. A multi-agent system for automating scientific discovery. *Nature* **655**, 497–505 (2026).
12. Huang, K. et al. Autonomous biomedical research with an artificial intelligence agent. *Science* **393**, eadz4351 (2026).
13. Alber, S. et al. Cellvoyager: AI CompBio agent generates new insights by autonomously analyzing biological data. *Nat. Methods* **23**, 749–759 (2026).
14. Qu, Y. et al. CRISPR-GPT for agentic automation of gene-editing experiments. *Nat. Biomed. Eng.* **10**, 245–258 (2026).
15. Seo, M., Baek, J., Lee, S. & Hwang, S. J. Paper2Code: automating code generation from scientific papers in machine learning. In *Proc. 14th International Conference on Learning Representations* 38867–38932 (ICLR, 2026).
16. Movassaghi, C. S., Momenzadeh, A. & Meyer, J. G. From articles to code: on-demand generation of core algorithms from scientific publications. *Bioinformatics* **42**, btag015 (2026).
17. Hou, X., Zhao, Y. & Wang, H. Model context protocol (MCP): landscape, security threats, and future research directions. *ACM Trans. Softw. Eng. Methodol*. https://doi.org/10.1145/3796519 (2025).
18. Nowakowski, P. et al. The collage authoring environment. *Procedia Comput. Sci.* **4**, 608–617 (2011).
19. Rule, A. et al. Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks. *PLoS Comput. Biol.* **15**, e1007007 (2019).
20. Stojnic, R. & Taylor, R. Papers with Code is joining Facebook AI. *Medium* https://medium.com/paperswithcode/papers-with-code-is-joining-facebook-ai-90b51055f694 (2019).
21. Ragan-Kelley, B. et al. Binder 2.0-reproducible, interactive, sharable environments for science at scale. In *Proc. 17th Python in Science Conference* 113–120 (SciPy, 2018).
22. Staubitz, T., Klement, H., Teusner, R., Renz, J. & Meinel, C. CodeOcean-a versatile platform for practical programming excercises in online environments. In *2016 IEEE Global Engineering Education Conference (EDUCON)* 314–323 (IEEE, 2016).
23. Anthropic. Claude Code: deep coding at terminal velocity. *Anthropic* https://www.anthropic.com/claude-code (2025).
24. Kjolby, M., Nielsen, M. S. & Petersen, C. M. Sortilin, encoded by the cardiovascular risk gene *SORT1*, and its suggested functions in cardiovascular disease. *Curr. Atheroscler. Rep.* **17**, 18 (2015).
25. The GTEx Consortium. The GTEx Consortium atlas of genetic regulatory effects across human tissues. *Science* **369**, 1318–1330 (2020).
26. Wainberg, M. et al. Opportunities and challenges for transcriptome-wide association studies. *Nat. Genet.* **51**, 592–599 (2019).
27. Mostafavi, H., Spence, J. P., Naqvi, S. & Pritchard, J. K. Systematic differences in discovery of genetic effects on gene expression and complex traits. *Nat. Genet.* **55**, 1866–1875 (2023).
28. Wager, S. & Athey, S. Estimation and inference of heterogeneous treatment effects using random forests. *J. Am. Stat. Assoc.* **113**, 1228–1242 (2018).
29. Bloom, J., Tigges, C., Duong, A. & Chanin, D. SAELens. *GitHub* https://github.com/decoderesearch/SAELens (2024).
30. Hans, A. et al. Spotting LLMs with binoculars: zero-shot detection of machine-generated text. In *Proc. 41st International Conference on Machine Learning* 17519–17537 (PMLR, 2024).
31. Hollmann, N. et al. Accurate predictions on small data with a tabular foundation model. *Nature* **637**, 319–326 (2025).
32. Ravi, N. et al. Sam 2: segment anything in images and videos. In *Proc. 13th International Conference on Learning Representations* 28085–28128 (ICLR, 2025).
33. Chernozhukov, V., Demirer, M., Duflo, E. & Fernández-Val, I. Fisher–Schultz lecture: generic machine learning inference on heterogeneous treatment effects in randomized experiments, with an application to immunization in India. *Econometrica* **93**, 1121–1164 (2025).
34. Brodersen, K. H., Gallusser, F., Koehler, J., Remy, N. & Scott, S. L. Inferring causal impact using Bayesian structural time-series models. *Ann. Appl. Stat.* **9**, 247–274 (2015).
35. Knight, V. & Campbell, J. Nashpy: a Python library for the computation of Nash equilibria. *J. Open Source Softw.* **3**, 904 (2018).
36. Foreman-Mackey, D., Hogg, D. W., Lang, D. & Goodman, J. emcee: the MCMC hammer. *Publ. Astron. Soc. Pac.* **125**, 306–312 (2013).
37. Jin, Y. & Candès, E. J. Model-free selective inference under covariate shift via weighted conformal *p*-values. *Biometrika* **113**, asaf066 (2026).
38. Spence, J. P. et al. Specificity, length and luck drive gene rankings in association studies. *Nature* **649**, 918–925 (2026).
39. Ho, C.-H. et al. Genetic and epigenetic screens in primary human T cells link candidate causal autoimmune variants to T cell networks. *Nat. Genet.* **57**, 2536–2545 (2025).
40. Zhu, R. et al. Genome-scale perturb-seq in primary human CD4+ T cells maps context-specific regulators of T cell programs and human immune traits. *Cell* https://doi.org/10.1016/j.cell.2026.08.002 (2026).
41. Soskic, B. et al. Immune disease risk variants regulate gene expression dynamics during $CD4^+$ T cell activation. *Nat. Genet.* **54**, 817–826 (2022).
42. Rendon, A. & Schäkel, K. Psoriasis pathogenesis and treatment. *Int. J. Mol. Sci.* **20**, 1475 (2019).
43. Van der Laan, C. M. et al. Genome-wide association meta-analysis of childhood ADHD symptoms and diagnosis identifies new loci and potential effector genes. *Nat. Genet.* **57**, 2427–2435 (2025).

Methods

Details on implementing Paper2Agent

Paper2Agent converts a research paper and its public codebase into a production-ready MCP server and then exposes that server to an AI agent interface. We implemented this as a multi-agent system using Claude Code's agent SDK, where a central orchestrator agent coordinates specialized sub-agents through a six-step pipeline. Each sub-agent is defined by a structured prompt that specifies its role, permitted tools (for example, file read/write, shell execution and web access) and expected output schema. The orchestrator dispatches sub-agents sequentially across steps and in parallel within steps when multiple tutorials are processed concurrently.

The pipeline proceeds through six steps, with data flowing between steps via standardized JSON reports and file conventions:

1. Locate and download the codebase. Paper2Agent first attempts to automatically identify the associated code repository from the manuscript text, references, or supplementary materials. If automatic identification fails, if it returns multiple candidates, or if the user prefers to specify a particular repository, the repository URL can be provided directly. Once identified, the codebase is cloned or downloaded, along with associated resources such as supplementary data or configuration files. The outputs for this step are the cloned repository and detected language.
2. Environment setup. The environment manager sub-agent provides a clean, isolated virtual environment for the repository. The input is the cloned repository, and the outputs are an isolated virtual environment and test configuration files.
3. Tutorial discovery. The tutorial scanner sub-agent scans the repository to locate useful reference and educational materials and produces an index of candidate tutorials for tooling. The inputs are the cloned repository and an optional tutorial filter. The output is a JSON file representing a classified file index.
4. Tutorial execution and audit. The tutorial executor sub-agent runs the selected tutorials end-to-end with their example data, captures inputs, outputs, figures and runtime constraints, and records any implicit assumptions that must be made explicit. The inputs are tutorial source files, activated virtual environment and scanner report. The outputs are executed notebooks and per-tutorial execution reports.
5. Tool extraction, testing and refinement. This step involves two sub-agents operating in sequence. First, the tutorial tool extractor–implementor converts each executed tutorial into a standalone Python module containing reusable functions. It identifies generalizable analysis steps, parameterizes hard-coded values (file paths, thresholds, column names), enforces file-based inputs and outputs, and decorates each function as an MCP tool. Second, the test verifier–improver creates per-function test files using the tutorial's own example data as ground truth. Tests verify that expected output files are generated; functions that repeatedly fail have their MCP tool decorators removed and are excluded from the final server. The inputs are executed notebooks, virtual environment and scanner report. The outputs are tool modules, per-function test files, test logs and summaries.
6. MCP server assembly. The orchestrator integrates all validated tool modules into a unified MCP server with a manifest, versioning and basic security defaults, ready to be used by an orchestrator or co-scientist agent.

Each sub-agent is instantiated as an independent LLM session (Claude) with a role-specific system prompt and a defined set of permitted tools (file read/write, shell execution, code search).

- Environment manager: a specialized agent responsible for creating clean, reproducible environments for research codebases. It analyses project setup requirements, provisions an isolated workspace, installs all necessary dependencies and ensures the code runs without conflicts. Standardizing environment setup enables reliable execution and reproducibility across different systems.
- Tutorial scanner: a specialized agent for reviewing the public codebases to identify and organize educational resources. It systematically scans available materials, distinguishes genuine tutorials from other files and highlights those most useful for reuse. The agent then produces clear summaries and reports, providing a structured view of which resources are worth keeping and which can be set aside.
- Tutorial executor: executes approved tutorials end-to-end to generate gold-standard outputs and reference data for downstream tool extraction. The agent systematically resolves execution errors, preserves all generated outputs (numerical results, figures, tables) and records execution metadata. The resulting executed notebooks, extracted figures and generated data files serve as authoritative reference material for test creation and validation.
- Tutorial tool extractor–implementor: a specialized agent that converts tutorials into reusable tools. It reviews selected tutorials, identifies tasks that generalize beyond the example data and implements each as a clean, single-purpose function with clear inputs, outputs, and defaults. The agent parameterizes hard-coded values, enforces file-based inputs, saves essential results and figures, and returns a standardized summary of produced artefacts. Its goal is to create a practical function library that reproduces tutorial results on the original data while remaining ready to run on new datasets.
- Test verifier–improver: a specialized agent that creates, runs and refines tests for tutorial implementations. It uses only the tutorial's own examples to ensure complete coverage and faithful reproduction of numerical and visualization results. A test passes when expected files are generated, numerical results match tutorial outputs exactly (with a 3% tolerance for floating-point values) and generated figures match reference visualizations (verified via perceptual hashing with Hamming distance < 20). The agent runs in a loop of generating tests, executing them, diagnosing failures and applying fixes, with a maximum of six attempts per function. If functions repeatedly fail, their MCP decorators are removed, a failure comment is added and they will not be included in the MCP server. All results and logs are recorded for transparency.

The orchestrator agent invokes sub-agents as needed at different stages of the process. As the Paper2Agent workflow progresses, the results are automatically recorded for each step for traceability and reproducibility. The detailed setup and prompt are available in the Paper2Agent GitHub repository (https://github.com/jmiao24/Paper2Agent).

Generation and analysis of AlphaGenome agent

We applied the Paper2Agent framework to the AlphaGenome paper to generate an AlphaGenome MCP and connected the MCP with Claude Code to create the AlphaGenome agent. The generated AlphaGenome MCP server is remotely hosted on Hugging Face Spaces (Code availability). To verify reproducibility, the AlphaGenome agent was evaluated using 15 original tutorial-based and 15 novel queries. We prompted the agent with the queries and compared the agent's response with the ground truth answer. The prompt used to query the AlphaGenome agent on interpreting LDL genetic associations is: 'Use AlphaGenome to interpret why chr1:109274968:G>T associates with LDL cholesterol. Identify the causal genes and assess regulatory effects across modalities in liver. Generate a publication-ready report with figures. My AlphaGenome API key is: < API_KEY > . Reason step by step'. The detailed benchmark queries are available in the Paper2Agent repository (https://github.com/jmiao24/Paper2Agent).

Benchmarking the AlphaGenome agent against Claude + Repo and Biomni

For both the tutorial-based and novel benchmarks described above, we followed these general evaluation steps:

1. Generate ground truth answers for each query using manually curated and executed code.
2. Generate and capture the agent's response to the query, as well as performance metrics like runtime and cost.
3. Manually review and grade the agent's response relative to the ground truth.
4. Summarize the agent's performance across all queries for the benchmark dataset.

Unless otherwise specified, the primary evaluations used claude-sonnet-4-20250514 as the underlying LLM. All evaluations were run locally on a MacBook Air (M2 chip, 8-core CPU, 8-core GPU, 8 GB unified memory), using model APIs as needed. Each query was run in non-interactive mode from the command line, and all output was captured in JSON format—for example,

bash$ claude --model "claude-sonnet-4-20250514" --print --output-format "json" <prompt>

For the 30 open-ended AlphaGenome queries, each question was designed to require the agent to (1) independently formulate an analysis plan; (2) compose multiple tool calls (for example, comparing variant effect predictions across tissues, integrating motif and QTL evidence, or evaluating multiple candidate variants); and (3) synthesize results into a biological conclusion. These queries were scored by two domain experts using a predefined rubric based on key entity matching (for example, correct gene, variant, tissue, or biological conclusion).

For the AlphaGenome agent generated by Paper2Agent, each benchmark query was wrapped in a system prompt instructing the agent to use the available AlphaGenome MCP tools, return a structured JSON response containing both the final answer and step-by-step reasoning, and retrieve the API key from the project environment. The agent received no additional context beyond the MCP tool definitions and the query itself. For the Claude + Repo baseline, we used Claude Code with access to a local clone of the AlphaGenome repository. The system prompt instructed the agent to write and execute Python code using the AlphaGenome library to answer each query, explicitly prohibiting the agent from copying answers from tutorial notebooks or documentation. The agent was required to return a structured JSON response with the final answer and the executed code. We utilized the API-based version of Biomni. The system prompt directed the agent to the AlphaGenome repository and API key, and required a structured JSON response. The full prompt templates are provided in Supplementary Note. Our benchmarking tools and analysis are available in the Paper2Agent repository (https://github.com/jmiao24/Paper2Agent).

Generation and analysis of TISSUE agent

We applied the Paper2Agent framework to the TISSUE paper to generate a TISSUE MCP and connect it with Claude Code to create the TISSUE agent. To assess reproducibility, we compared the TISSUE agent's outputs against those generated by human researchers using identical mouse somatosensory cortex spatial transcriptomics data[44]. Human researchers performed the analysis based on the tutorial in the TISSUE GitHub repository.

Generation and analysis of Scanpy agent

The Paper2Agent framework was applied to the Scanpy software package to generate a Scanpy agent. This agent was restricted to the preprocessing and clustering workflows within Scanpy, providing a focused and reproducible pipeline for single-cell RNA-seq analysis. The resulting MCP server was deployed and integrated with Claude Code, creating a Scanpy agent.

To construct the workflow in MCP prompts, we prompted Paper2Agent with 'Based on the tools you have, construct an MCP prompt to replicate the tutorial in the correct order. Always inspect the data first, and only deviate from the default settings if adhering to them would yield incorrect results'. This ensured that the generated MCP prompts encoded the standard Scanpy preprocessing and clustering pipeline in a reproducible and interpretable manner.

Reproducibility was evaluated by comparing the agent's outputs with results obtained by human researchers following the official Scanpy reference tutorials. Three publicly available 10x Genomics PBMC single-cell RNA-seq datasets and four additional datasets were together used for benchmarking (Data availability). Across these datasets, the agent faithfully reproduced key workflow steps—including gene filtering, normalization, principal component analysis, neighbourhood graph construction and clustering—and produced results consistent with human-executed analyses.

Large-scale evaluation of Paper2Agent

To evaluate the generalizability, scalability and robustness of Paper2Agent, we conducted a systematic evaluation across three corpora, all processed end-to-end by Paper2Agent without manual cleanup, code modification or intervention. (1) One-hundred computational biology papers retrospectively sampled from the bioinformatics category of *bioRxiv* by iterating backward chronologically from December 2025, without filtering for documentation quality, repository maintenance status, or code completeness, ensuring the sample reflects the natural heterogeneity of research code in practice. (2) Twenty-six data- and discovery-focused papers (13 *bioRxiv* and 13 *Nature*, year 2025) reporting experimental results, datasets or discoveries with accompanying supplementary materials, used to evaluate Paper2Agent's structured resource layer over manuscript text, supplementary files and metadata. (3) Ten non-biology computational papers spanning diverse programming paradigms and scientific domains: grf (causal inference and econometrics), SAELens (mechanistic interpretability), Binoculars (natural language processing), SAM2 (computer vision), TabPFN (tabular machine learning), GenericML (heterogeneous treatment effects), CausalImpact (Bayesian structural time-series inference), Nashpy (computational game theory), emcee (affine-invariant Markov chain Monte Carlo) and conformal-selection (FDR-controlled selective inference). Successful agentification was defined as the generated MCP server completing tool extraction, execution, and automated validation end-to-end without human intervention.

For the 74 successfully agentified computational biology papers, we derived 300 tutorial-based benchmark questions, with ground truth answers obtained by executing the original code and verifying against the tutorial outputs. For the 26 data- and discovery-focused papers, we curated 100 synthesis-based questions requiring integration across main text and supplementary materials, including reinterpretation tasks (for example, 'The paper reports results using Pearson correlation; reanalyse the conclusions using Spearman correlation') and cross-referencing across tables, figures, and narrative text. For the 10 non-biology computational papers, we constructed 42 execution-based benchmark questions. Detailed prompts for all evaluations are provided in the Supplementary Note.

For benchmarking on computational biology papers, the primary baseline was Claude Code with direct repository access (Claude + Repo). In this setup, the agent was provided with the full code repository and paper but without any MCP tools, structured resources, or prompts, and was given the same queries to answer by writing and executing code. We did not include Biomni in this benchmark owing to its high cost. Primary evaluations used Sonnet 4 (claude-sonnet-4-20250514); an additional Claude + Repo baseline used Sonnet 4.6 on the same 300 questions. For benchmarking on data- and discovery-focused papers, the baseline was Claude with browser-use capabilities and direct access to the paper URL, representing a strong human-assisted LLM setup in which the agent can browse and read the paper directly.

We report mean accuracy ± s.e.m. across questions using a bootstrap procedure. To compare Paper2Agent with baselines, we used paired *t*-tests on per-run accuracy for tutorial-based benchmarks and bootstrap hypothesis tests (10,000 resamples) for the large-scale 100-paper

evaluation, reporting 95% confidence intervals for the accuracy differences. API costs were tracked by logging all LLM API calls during both MCP construction and downstream query answering. For MCP construction, we report total cost (the sum of all API calls across sub-agents) and time from pipeline initiation to MCP server creation. For query-time evaluation, we report per-query cost and latency, measured as time from query submission to final answer.

To evaluate false positive behaviour, we constructed an adversarial out-of-scope benchmark by randomly permuting paper–question pairs across the 26 data- and discovery-focused papers, ensuring each question was paired with a paper that does not contain the relevant information. We evaluated Paper2Agent under two conditions: (1) with an explicit rejection instruction ("If the question is not related to the files, say 'I don't know'"); and (2) without any explicit rejection instruction. In both conditions, we measured the correct rejection rate, defined as the fraction of out-of-scope queries for which the agent declined to answer rather than producing a hallucinated response.

To probe for potential tutorial-specific memorization, we performed a targeted inspection of generated MCP server implementations across the 100 computational biology papers. We examined whether extracted tools contained hard-coded tutorial constants, fixed file paths, cached outputs, or dataset-dependent heuristics. In addition, we implemented an automated reviewer agent that scans generated MCP server Python files to flag potential hard-coded values, fixed dataset paths, cached outputs or tutorial-specific logic, providing a systematic check for implementation-level shortcut learning beyond manual inspection.

We conducted five ablated variants of the full system using AlphaGenome and evaluated their performance using 30 tutorial and novel benchmark questions. In the monolithic agent setting, all sub-tasks, including environment setup, tutorial scanning, tool extraction and testing, were executed within a single agent using a single 200,000-token context window, without decomposition into specialized sub-agents. In the non-parallel multi-agent setting, the same four sub-agents were used but were forced to run sequentially rather than in parallel, isolating the contribution of parallel orchestration to runtime efficiency. In the no test verifier–improver setting, the test verifier–improver sub-agent was removed, and extracted tools were deployed without iterative validation against tutorial outputs, testing whether automated verification is essential for tool correctness. In the Markdown skill files variant, MCP tools were replaced with Markdown skill files using Claude Code's Skills feature, which encode tool usage instructions as structured text rather than executable tools. Finally, in the alternative scaffolding (OpenCode) variant, the Claude Code backend was replaced with OpenCode, testing whether MCP construction quality depends on the specific agent scaffolding. All evaluations used claude-sonnet-4-20250514 as the base model.

To evaluate robustness to upstream code defects, we constructed adversarial variants of three representative repositories: AlphaGenome (Python notebook-based), POP-TOOLS (Python command line-based)[45] and mlearner (R command line-based)[46]. For each repository, we injected four categories of execution-level errors. These included missing dependencies, in which required packages were removed from environment specification files such as requirements.txt, DESCRIPTION, or README installation instructions; broken file paths, where input paths were modified to reference nonexistent directories or filenames; typographical errors, introduced by misspelling function names, package names, or variable names in executable code cells or scripts; and deprecated API calls, where valid function calls were replaced with deprecated or incompatible alternatives. Each error type was injected independently into each repository, yielding 12 adversarial configurations in total. All modifications were applied prior to running Paper2Agent and were not disclosed to the agent. Paper2Agent was tasked with performing the standard agentification pipeline on each adversarial repository. We recorded whether the agent detected the injected errors, the repair strategies employed, and whether the final MCP server passed all validation tests. Detailed examples of injected errors and observed agent behaviour are provided in Supplementary Note. To evaluate whether executable tutorials are required, we removed all executable tutorials from POP-TOOLS while retaining the README and source code, then ran the standard Paper2Agent pipeline. We assessed whether the resulting MCP server exposed functional tools and reproduced human-executed POP-TOOLS CLI outputs across five analysis tasks; full details are provided in Supplementary Note. All the benchmarking questions and GitHub repositories are provided in the Paper2Agent repository (https://github.com/jmiao24/Paper2Agent).

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

We aim to prioritize and validate candidate causal genes associated with the psoriasis risk variant rs887314 using three Paper2Agent-generated agents: an AlphaGenome agent for computational variant effect prediction, an MPRA-coupled scCRISPRi agent that provides CRE perturbation effects on gene expression in primary human $CD4^+$ T cells and a $CD4^+$ T cell Perturb-seq agent that provides transcriptome-wide gene expression profiles following individual gene knockdowns under three culture conditions (Rest, Stim8hr, Stim48hr). In the source $CD4^+$ T cell Perturb-seq dataset, 'Rest' cells were maintained without restimulation and harvested after 8 h, whereas 'Stim8hr' and 'Stim48hr' cells were restimulated with 12.5 µl ml^{-1} ImmunoCult human CD3/CD28/CD2 T cell activator and harvested after 8 h and 48 h, respectively. All agents exposed their underlying datasets, metadata and supplementary tables as structured, queryable resources. The AlphaGenome agent was prompted to score rs887314 across all available prediction modalities, with analyses restricted to $CD4^+$ T cells. For each gene within the local genomic window surrounding rs887314, the agent returned a predicted expression impact score, and genes were ranked according to these predicted effects. Then, we instructed the AI co-scientist with access to the MPRA scCRISPRi and $CD4^+$ T cells Perturb-seq paper and data under human-in-the-loop supervision. The full prompts and documentation of human interventions are provided in Supplementary Note.

Agent availability

The Paper2Agent-generated AlphaGenome agent is publicly available at https://huggingface.co/spaces/Paper2Agent/alphagenome_agent.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

This paper utilized publicly available data for analysis: 10x Genomics single-cell RNA-seq datasets: http://cf.10xgenomics.com/samples/cell-exp/3.0.0/pbmc_1k_v2/pbmc_1k_v2_filtered_feature_bc_matrix.h5; http://cf.10xgenomics.com/samples/cell-exp/3.0.0/pbmc_1k_v3/pbmc_1k_v3_filtered_feature_bc_matrix.h5; http://cf.10xgenomics.com/samples/cell-exp/3.0.0/pbmc_1k_protein_v3/pbmc_1k_protein_v3_filtered_feature_bc_matrix.h5; http://cf.10xgenomics.com/samples/cell-exp/6.0.0/Brain_Tumor_3p_LT/Brain_Tumor_3p_LT_filtered_feature_bc_matrix.h5; http://cf.10xgenomics.com/samples/cell-exp/3.0.0/neuron_1k_v3/neuron_1k_v3_filtered_feature_bc_matrix.h5; http://cf.10xgenomics.com/samples/cell-exp/3.0.0/neuron_10k_v3/neuron_10k_v3_filtered_feature_bc_matrix.h5; and http://cf.10xgenomics.com/samples/cell-exp/3.0.0/heart_1k_v3/heart_1k_v3_filtered_feature_bc_matrix.h5. Mouse somatosensory cortex spatial transcriptomics data (Dataset 15): https://doi.org/10.5281/zenodo.8259942. ADHD GWAS summary statistics: https://www.ebi.ac.uk/gwas/studies/GCST90568440 and https://www.ebi.ac.uk/gwas/studies/GCST90568441. GTEx portal: https://gtexportal.org/home/snp/chr1_109274968_G_T_b38. $CD4^+$ T cell Perturb-seq data: https://virtualcellmodels.cziscience.com/dataset/

genome-scale-tcell-perturb-seq. MPRA-coupled scCRISPRi data for CRE perturbation: https://static-content.springer.com/esm/art%3A10.1038%2Fs41588-025-02301-3/MediaObjects/41588_2025_2301_MOESM4_ESM.xlsx.

Code availability

Paper2Agent is publicly available at https://github.com/jmiao24/Paper2Agent. AlphaGenome MCP server: https://huggingface.co/spaces/Paper2Agent/alphagenome_mcp. Scanpy MCP server: https://huggingface.co/spaces/Paper2Agent/scanpy_mcp. TISSUE MCP server: https://huggingface.co/spaces/Paper2Agent/tissue_mcp.

44. Sun, E. Single-cell spatial transcriptomics data with paired RNAseq for TISSUE spatial gene expression prediction. *Zenodo* https://doi.org/10.5281/zenodo.8259942 (2024).
45. Miao, J. et al. Valid inference for machine learning-assisted genome-wide association studies. *Nat. Genet.* **56**, 2361–2369 (2024).
46. Miao, J. et al. Polygenic prediction of treatment efficacy with causal transfer learning. Preprint at *medRxiv* https://doi.org/10.1101/2025.10.15.25338051 (2025).

Acknowledgements We thank A. Abid, E. Sun, E. Dann, members of the Zou laboratory and the Pritchard laboratory for helpful feedback during the project.

Author contributions J.M. and J.Z. conceived the study. J.M. designed and developed the framework. J.M., J.R.D. and Y.Z. performed the benchmarking. J.M. performed the case studies. J.K.P. and J.Z. supervised the project. J.M. and J.Z. drafted the manuscript. All authors discussed the results and reviewed and edited the manuscript.

Funding This work was supported by the US National Institutes of Health (grant R01HG014005). J.Z. is supported by funding from the Chan-Zuckerberg Biohub and the Stanford Center for Digital Health.

Competing interests The authors declare no competing interests.

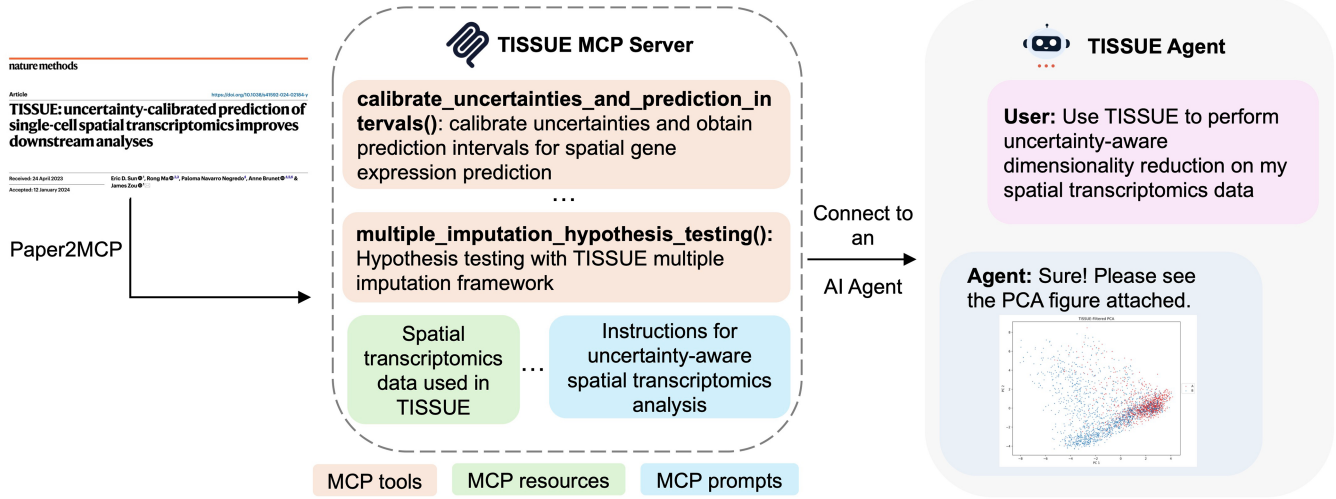
Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41586-026-11044-y.

Correspondence and requests for materials should be addressed to Jiacheng Miao or James Zou.

Peer review information *Nature* thanks Olivier Elemento and Su-In Lee, who co-reviewed with Soham Gadgil, and the other, anonymous, reviewers for their contribution to the peer review of this work. Peer reviewer reports are available.

A Paper2Agent automatically generates TISSUE MCP server and agent



B TISSUE agent offers Q&A support that guides researchers in applying TISSUE effectively

User: Based on the TISSUE MCP server, what can TISSUE do and what are the required input and expected output?

TISSUE Capabilities

TISSUE is a computational framework for spatial gene expression analysis that combines spatial and single-cell RNA-seq data. It provides 6 main functions:

1. Predict Spatial Gene Expression

- **Input:** Spatial count matrix, spatial locations, scRNA-seq count matrix
- **Output:** Prediction visualization and results
- **Methods:** SpaGE, Tangram, or Harmony

2. Calibrate Uncertainties and Prediction Intervals

- **Input:** Processed AnnData from prediction step
- **Output:** Uncertainty calibration and interval visualization
- **Purpose:** Quantify prediction confidence

3. Multiple Imputation Hypothesis Testing

- **Input:** Calibrated AnnData with condition labels
- **Output:** Statistical test results and condition visualization
- **Tests:** t-test, SpatialDE, Wilcoxon tests

4. Cell Filtering for Supervised Learning

- **Input:** Calibrated AnnData with conditions
- **Output:** Filtered results and classifier performance metrics
- **Purpose:** Improve ML classifier performance

5. Cell Filtering for PCA

- **Input:** Calibrated AnnData with conditions
- **Output:** PCA visualization and clustering results
- **Purpose:** Enhance dimensionality reduction

6. Weighted PCA (TISSUE-WPCA)

- **Input:** Calibrated AnnData with conditions
- **Output:** Weighted PCA visualization using uncertainty-based weights
- **Purpose:** Uncertainty-aware dimensionality reduction

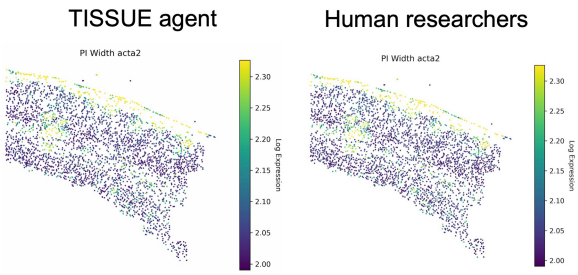
Required File Formats

All input files should be **tab-delimited text format** with:
- Count matrices: Header with gene names, rows as cells
- Location files: x and y coordinates for each cell

The workflow typically proceeds: Prediction → Uncertainty Calibration → Downstream Analysis (testing/filtering/visualization).

C TISSUE agent produces identical results to those of human researchers

User: Use TISSUE to generate the prediction interval for gene *Acta2*. Below are my data:
- Spatial count matrix: Spatial_count.txt
- Spatial locations: Location.txt
- scRNA-seq count matrix: scRNA_count.txt



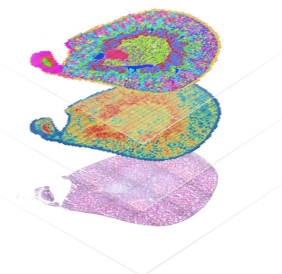
D TISSUE MCP resources provide a structured format for spatial transcriptomics data used in TISSUE

```python
# ---- Canonical dataset registry (idempotent, read-only) ----
DATASETS: List[Dict] = [
    {
        "id": "zenodo_master",
        "label": "Processed pairings + annotated adult mouse SVZ MERFISH",
        "species": ["mouse"],
        "tissue": ["SVZ", "multiple"],
        "spatial": {"modality": "MERFISH", "url": "https://doi.org/10.5281/zenodo.8259942"},
        "rna": {"modality": None, "url": None},
        "notes": "All processed pairings deposited; raw/large intermediates on request.",
        "type": "deposit"
    },
    {
        "id": "mouse_hippocampus_seqFISH_10x",
        "label": "Mouse hippocampus (seqFISH + 10x Chromium)",
        "species": ["mouse"],
        "tissue": ["hippocampus"],
        "spatial": {"modality": "seqFISH", "url": "https://content.cruk.cam.ac.uk/jmlab/SpatialMouseAtlas2020/"},
        "rna": {"modality": "10x Chromium", "url": "https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE158450"},
        "notes": "Use HIPP_sc_Rep1_10X sample",
        "type": "pairing"
    },
    ...
]

# ---- Resources ----

@mcp.resource("datasets://by/species/{name}")
def by_species(name: str) -> Dict[str, List[Dict]]:
    """Filter datasets by species (case-insensitive)."""
    n = name.lower()
    hits = [d for d in DATASETS if any(s.lower() == n for s in d["species"])]
    return {"species": name, "datasets": hits}
```

Filter data by species

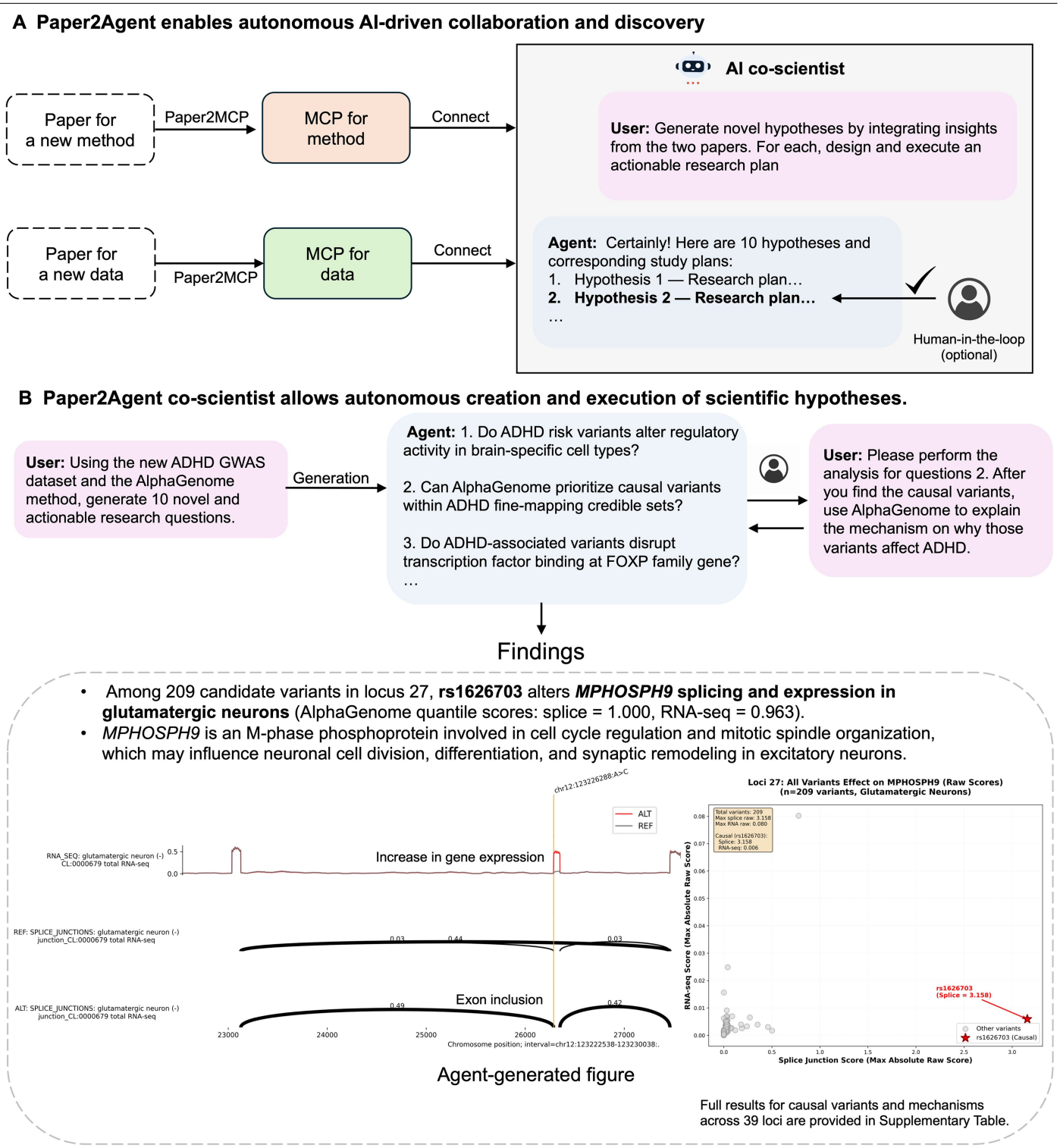
User: Download the spatial transcriptomics data used in TISSUE paper

TISSUE agent calls Zenodo REST API



Automatically downloaded spatial transcriptomics data

Extended Data Fig. 1 | Overview of the Paper2Agent-generated TISSUE agent. (A) Construction of the TISSUE MCP server and agent. (B) Q&A support for uncertainty-aware spatial transcriptomics analysis. (C) Reproducibility confirmed by matching human researcher results. (D) Structured MCP resources enable standardized dataset access and automated downloads.



Extended Data Fig. 2 | Paper2Agent enables autonomous AI-driven collaboration and genomic discovery. (A) Paper2Agent transforms scientific papers into Model Context Protocol (MCP) resources for both methods and data, allowing an AI co-scientist to integrate them and autonomously generate novel hypotheses and actionable research plans. (B) Using the ADHD GWAS dataset and the AlphaGenome method MCPs, the agent autonomously generates and tests scientific hypotheses, prioritizes causal variants, and interprets molecular mechanisms. The agent prioritized rs1626703 as a likely causal variant among 209 candidate variants. The agent then showed computationally using AlphaGenome that rs1626703 may alter splicing of *MPHOSPH9* and increase its expression in glutamatergic neurons, generating a candidate mechanistic hypothesis for ADHD risk that remains to be validated experimentally.

nature portfolio

Corresponding author(s): James Zou

Last updated by author(s): Jul 20, 2026

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed
☐	☒ The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☒	☐ A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒ The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐ A description of all covariates tested
☒	☐ A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒ A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☐	☒ For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐ For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐ For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐ Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	No software was used for data collection in this study. All data analysed were publicly available from previously published sources, including GTEx, 10x Genomics, Zenodo, the ADHD GWAS dataset, the MPRA-coupled scCRISPRi data for cis-regulatory element perturbation, and CD4+ T-cells perturb-seq data
Data analysis	Data analysis was performed using Python v3.12.1, Paper2Agent (17 September 2025; commit 37ecd69), AlphaGenome v0.2.0, TISSUE (10 March 2024; commit ffc3599), and Scanpy v1.11.4. All software is open source and publicly available through the corresponding GitHub repositories.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our policy

No new datasets were generated in this study. All datasets analysed are publicly available from the following sources:
10x Genomics single-cell RNA-seq datasets:
http://cf.10xgenomics.com/samples/cell-exp/3.0.0/pbmc_1k_v2/pbmc_1k_v2_filtered_feature_bc_matrix.h5
http://cf.10xgenomics.com/samples/cell-exp/3.0.0/pbmc_1k_v3/pbmc_1k_v3_filtered_feature_bc_matrix.h5
http://cf.10xgenomics.com/samples/cell-exp/3.0.0/pbmc_1k_protein_v3/pbmc_1k_protein_v3_filtered_feature_bc_matrix.h5
http://cf.10xgenomics.com/samples/cell-exp/6.0.0/Brain_Tumor_3p_LT/Brain_Tumor_3p_LT_filtered_feature_bc_matrix.h5
http://cf.10xgenomics.com/samples/cell-exp/3.0.0/neuron_1k_v3/neuron_1k_v3_filtered_feature_bc_matrix.h5
http://cf.10xgenomics.com/samples/cell-exp/3.0.0/neuron_10k_v3/neuron_10k_v3_filtered_feature_bc_matrix.h5
http://cf.10xgenomics.com/samples/cell-exp/3.0.0/heart_1k_v3/heart_1k_v3_filtered_feature_bc_matrix.h5
Mouse somatosensory-cortex spatial transcriptomics data (Dataset 15):
https://zenodo.org/records/8259942
GTEx eQTL data for chr1:109274968:G>T:
https://gtexportal.org/home/snp/chr1_109274968_G_T_b38
ADHD GWAS summary statistics:
https://www.ebi.ac.uk/gwas/studies/GCST90568440
https://www.ebi.ac.uk/gwas/studies/GCST90568441
CD4+ T-cell Perturb-seq data:
https://virtualcellmodels.cziscience.com/dataset/genome-scale-tcell-perturb-seq
MPRA-coupled scCRISPRi data for cis-regulatory-element perturbation:
https://static-content.springer.com/esm/art%3A10.1038%2Fs41588-025-02301-3/MediaObjects/41588_2025_2301_MOESM4_ESM.xlsx

Research involving human participants, their data, or biological material

Policy information about studies with human participants or human data. See also policy information about sex, gender (identity/presentation), and sexual orientation and race, ethnicity and racism.

Reporting on sex and gender	This study did not involve the collection or analysis of individual-level human participant data. Therefore, sex and gender variables were not applicable.
Reporting on race, ethnicity, or other socially relevant groupings	No human participant or socially categorized data were used in this study. All analyses were performed on publicly available, de-identified genomic and transcriptomic datasets (e.g., GTEx, 10x Genomics, and ADHD GWAS summary statistics).
Population characteristics	Not applicable. The study analyzed only publicly available, aggregated datasets without individual-level identifiers or demographic variables.
Recruitment	Not applicable. No participants were recruited; all data were obtained from previously published, publicly accessible sources.
Ethics oversight	Not applicable. The study used only publicly available, de-identified datasets and did not involve any new human data collection. Ethics approvals for the original datasets (e.g., GTEx, 10x Genomics, ADHD GWAS) are described in their respective publications.

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	The number of datasets was not determined by a statistical power calculation because this study evaluated software functionality rather than

Sample size	biological effect sizes. Public datasets were selected to cover each prespecified case study and provide diverse data modalities, tissues, dataset sizes and reference outputs. They were considered sufficient because they enabled evaluation of every intended workflow and comparison with human-executed or published reference results. No datasets were excluded based on the observed outcomes.
Data exclusions	No data were excluded. All publicly available datasets were used in their entirety as provided by their original sources.
Replication	Stochastic agent evaluations were performed in five independent runs per method. The AlphaGenome benchmark, TISSUE reproducibility analysis, non-biology benchmark and repaired-server evaluations were each repeated five times. The Scanpy workflow was evaluated once on each of 11 independent datasets (four primary and seven additional datasets) against human-executed references. The monolithic-agent ablation was repeated three times. Deterministic POP-TOOLS validation was performed once for each of five analysis tasks because the outputs were byte-for-byte identical to the human-executed references.
Randomization	Not applicable. This study did not involve experimental groups or participant assignment. All analyses were computational and based on existing datasets
Blinding	Blinding was not applicable to dataset selection or automated computational analyses because no participants, treatment groups or experimental allocations were involved. Benchmark responses were assessed against predefined ground-truth answers and scoring rubrics. Two human experts graded the responses independently; formal blinding to system identity was not feasible because the response formats and execution traces could reveal the evaluated system. Grading discrepancies were resolved by consensus.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☒	☐ Animals and other organisms
☒	☐ Clinical data
☒	☐ Dual use research of concern
☒	☐ Plants

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging

Plants

Seed stocks	Not applicable. This study did not involve the use of any plant materials or seed stocks. All analyses were computational and performed on existing genomic and transcriptomic datasets.
Novel plant genotypes	Not applicable. No novel plant genotypes were generated in this study. The work focused entirely on developing and evaluating the Paper2Agent computational framework.
Authentication	Not applicable. No plant genotypes or biological materials were used. All results are based on publicly available datasets analyzed using the Paper2Agent system.