Paper2Agent – study brief

Miao, Davis, Zhang, Pritchard & Zou, "Reimagining research papers as interactive and reliable AI agents", *Nature* (2026), doi:10.1038/s41586-026-11044-y · Prepared 21 Sep 2026 for the meeting on 22 Sep.

PAPER stated or shown in the paper **MY READING** interpretation, not a claim by the authors

OUTSIDE background from outside the paper

Diagram colours: LLM reasoning deterministic code resources prompts human

Your 3-hour plan

TIME	DO THIS
0:00–0:10	Part 0 and A1 — the short version, and your own summary checked.
0:10–0:55	A2–A8 — architecture, MCP, build pipeline, testing, the query loop. **The most important 45 minutes.**
0:55–1:40	Part B with the paper open beside you. Read each figure using the guides.
1:40–2:05	Part C — what is shown vs claimed, a statistician's notes, the think-questions.
2:05–2:25	Part D — read the ten research ideas; pick two you could talk about.
2:25–2:50	Part E — say your intro and 2-minute summary out loud twice. Choose which questions you'll ask.
2:50–3:00	Write the one-page cheat sheet (E9) by hand. Then take a break.

0 · SHORT VERSION

A · THE SYSTEM
A1 Your summary, checked
A2 Problem and idea (p. 1)
A3 Architecture (p. 2)
A4 MCP, properly
A5 Figure 1, part by part
A6 The build pipeline (Methods)
A7 Testing
A8 The query loop

B · THE EVIDENCE
B1 Biology decoder
B2 AlphaGenome (Fig. 2, SORT1)
B3 Scanpy (Fig. 3)
B4 TISSUE (ED Fig. 1)
B5 Agents working together (Fig. 4)
B6 Spearman correlation
B7 ADHD example (ED Fig. 2)
B8 Large-scale evaluation

C · THINK CRITICALLY
C1 Shown / suggested / unsolved
C2 Limitations
C3 A statistician's notes
C4 Your 15 questions, answered
C5 Questions to make you think

D · RESEARCH IDEAS
D0 How to use them
D1–D10 Ten ideas

E · THE MEETING
E1 What today is
E2 60-second intro
E3 2-minute summary
E4 Conversation flow
E5 Programmes fact sheet
E6 Career questions
E7 A next step to offer
E8 Follow-up email
E9 Cheat sheet

F · FINAL MENTAL MODEL

0 · The short version

Paper2Agent is a group of Claude Code agents that reads a paper and its code, **runs the paper's own tutorials**, turns the tutorial steps into general functions, and **keeps only the functions that reproduce the tutorial outputs**. It packages those functions — with the paper text, data links and workflow recipes — as an **MCP server**. Any MCP-compatible AI agent can connect to that server, so a user can run the method by asking in plain English.

> **Keep two sets of agents apart.** This is the most common confusion.
>
> - **Builder agents** (“Paper2MCP”): run once per paper. They write code, run tests, and produce the server.
> - **The paper agent**: a normal chat agent (Claude Code in the paper) with that paper's server connected. This is what the user talks to.

A1 · Your summary, checked

Your summary is mostly right. Four refinements:

Your interpretation is slightly off:

1. **"Tests those tools against reference results."** The reference is the *output of the original tutorials*, which Paper2Agent runs itself — not the numbers printed in the paper. A pass means: "this new function reproduces what the original code produced on the tutorial example."
2. **Only tools are tested that way.** Resources (paper text, data links) and prompts (workflow recipes) are packaged, not execution-tested.
3. **"Connects that MCP server to an AI agent."** The server isn't tied to one agent. Any MCP-compatible agent can connect. The paper uses Claude Code running Claude Sonnet 4. Several paper servers can be connected to one agent at the same time — that is how Figure 4 works.
4. **It still produces something when code can't be turned into tools.** Paper2Agent then builds a resource-only server (paper, supplements, metadata). That version still answered 89% of questions correctly on 26 data-focused papers.

A2 · Page 1 – the problem and the idea

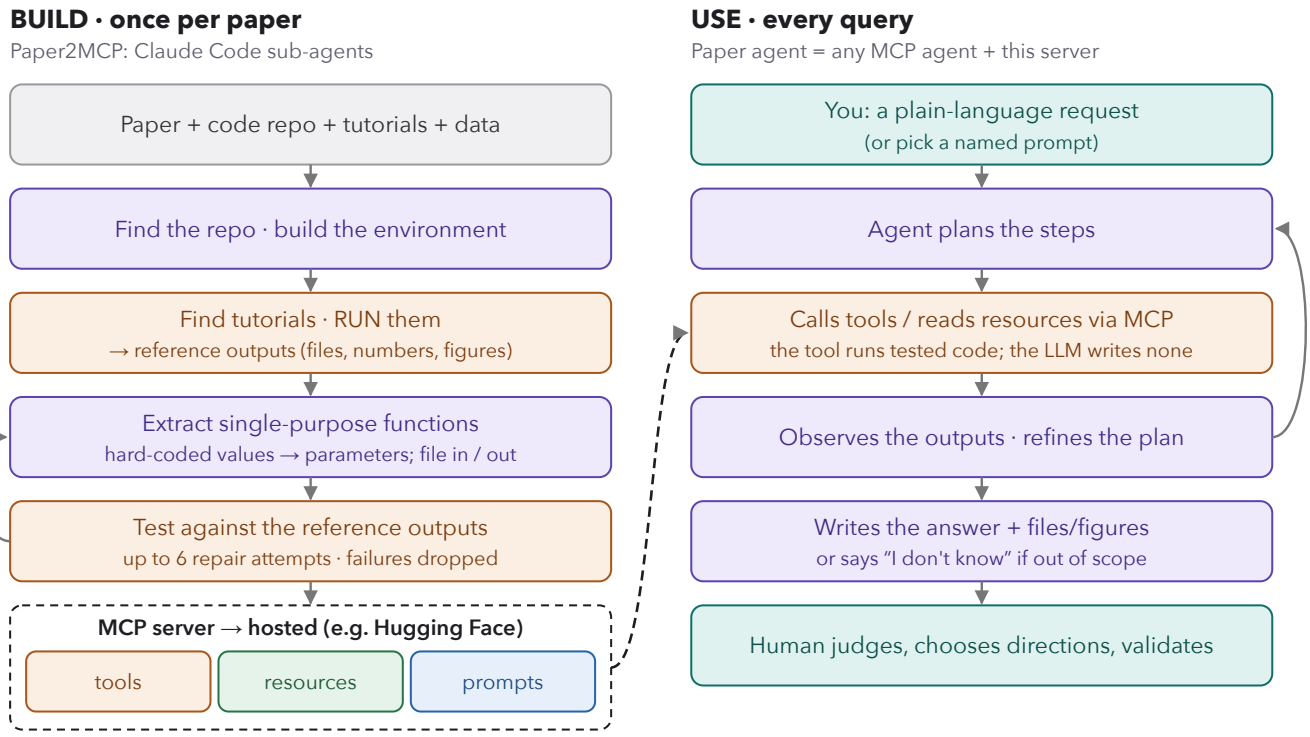
ITEM	IN PLAIN WORDS
The problem PAPER	A paper is passive. To use a computational method, you have to find the repository, install dependencies, configure an environment, and learn the right inputs and outputs. The authors' example: using AlphaGenome means setting up an environment, creating a client with an API key, building variant objects correctly and choosing output types.
What changes	The paper becomes something you can ask to explain, demonstrate or apply its method, in plain language — and it runs the real code.
The idea	Represent each paper as an MCP server. Fill it with **tested tools** (the method as functions), **resources** (paper, code link, data) and **prompts** (workflow recipes). Connect any LLM agent to it. The agent then plans and calls those tools to answer questions or analyse new data.
"Virtual corresponding author"	The corresponding author is the person you email with questions about a paper ("how do I run this on my data?"). The paper agent plays that role — it explains the method, tells you the inputs, and runs it for you — without waiting for a reply. It differs from a chatbot over the PDF (RAG) because it can *execute* the method, not just retrieve text.

Think. What can a real corresponding author do that this agent can't? (Judgement on edge cases, unpublished know-how, saying "don't use our method for that".)

A3 · Page 2 – the architecture

QUESTION	ANSWER
What goes in?	The paper, its code repository (found automatically or given by the user), tutorials/notebooks, example data, supplements.
What does Paper2Agent analyse?	The repository structure; which files are genuine tutorials; what each tutorial does; which steps generalise beyond the example; what the dependencies are.
What gets generated?	A configured software environment; tool modules (Python functions); tests and logs; the MCP server file that bundles tools, resources and prompts.

QUESTION	ANSWER
Where does MCP fit?	It is the standard interface between the paper's server and any agent. The server "speaks MCP"; the agent's host program is an MCP client.
Where does the LLM fit?	**Twice.** At build time, Claude Code sub-agents set up the environment, write wrappers and tests, and fix failures. At query time, the chat agent plans, picks tools and parameters, and interprets results.
What does the user touch?	Only the chat agent, in natural language (and optionally a named prompt, such as a slash command).
LLM reasoning	Build: choosing tutorials, writing wrapper code, writing tests, diagnosing failures. Query: planning, choosing tools, filling parameters, interpreting, writing the report.
Deterministic code	Running the original library inside tools; running tests; numeric and figure comparisons. Same inputs → same outputs (apart from randomness built into the method itself).
Where is testing?	**Build time only** (step 5). Nothing is re-tested when a user later runs a tool on new data.



Dashed arrow: the agent connects to the finished server. Build runs once (~45 min, ~US$14 for AlphaGenome); use runs on every question.

Figure A · Two phases. Colours: purple = LLM reasoning, orange = deterministic code, teal = human.

Your example project, conceptually

YOUR FILE	WHAT PAPER2AGENT WOULD DO	BECOMES
`load_data.py`	Wrap loading in a function whose file path is a parameter, not hard-coded.	tool `load_dataset(path)`
`clean_data.py`	Turn fixed thresholds into parameters, keeping the old values as defaults.	tool `clean_data(path, max_missing=0.2)`
`model.py`	Expose fitting as a function that writes metrics and predictions to files.	tool `fit_model(path, target, features)`
`visualise.py`	Save figures to files and return their paths.	tool `plot_results(...)`
`workflow.py`	This is the *order* of the steps. Encoded as a recipe that names the tools in sequence.	prompt "run the standard analysis on {data_path}"

YOUR FILE	WHAT PAPER2AGENT WOULD DO	BECOMES
README, paper, sample data, data dictionary	Stored as read-only reference material.	resources
A tutorial notebook, if you have one	Run end-to-end *first*; its outputs become the reference answers for the tests.	test references

PAPER Tools are single-purpose functions with file-based inputs and outputs; hard-coded paths, thresholds and column names become parameters. MY READING `workflow.py` could also become one "run everything" tool, but the paper's pattern is small tools plus a prompt that orders them — so the agent can still adapt between steps.

A4 · MCP, properly

For each idea: what it is → an analogy → where it is in the paper → why it matters.

MCP (Model Context Protocol)

What: an open standard for how AI applications connect to outside tools and data. OUTSIDE Introduced by Anthropic in November 2024; now widely supported.

Analogy: a USB-C port. Any device with the plug works with any laptop that has the port. No custom cable per device.

In the paper: any paper's server works with any MCP-compatible agent without custom integration. Several paper servers can be plugged into one agent.

Why: build a paper's server once; reuse it from many agents; combine papers.

MCP server (in this paper)

What: a program — here `<Paper>_mcp.py` — that offers the paper's tools, resources and prompts. It runs inside the environment Paper2Agent configured and is hosted remotely (Hugging Face Spaces), so users install nothing.

Analogy: a lab's service counter: a menu of analyses the lab can run for you, a shelf of documents, and procedure cards.

Why: the method and its setup travel together. The environment problem is solved once by the builder, not by every user.

MCP tool

What: a function with a name, a description, typed inputs and outputs, that the agent can call. The paper describes tools as executable functions that package a paper's methodological contribution.

Data example: a paper proposes a new anomaly-detection method. The tool `detect_anomalies(csv_path, column, window=30, threshold=3.0)` runs the paper's actual code and writes the flagged rows and a plot. The agent decides *when* to call it and *what values* to pass, but cannot change what happens inside.

In the paper: `score_variant_effect()` takes a genetic variant and returns predicted effects across measurement types and tissues.

Why: the LLM doesn't write analysis code at query time — it calls pre-tested code. This is how the paper reduces "code hallucination".

MCP resource

What: read-only material the agent can look up by address (a URI): manuscript, code link, supplementary tables, datasets, figures, training-data links.

Why "resource", not "tool": resources don't compute or change anything; they are context. **Tools act; resources inform.** (ED Fig. 1D shows a resource that filters a dataset list by species — still a lookup, not an analysis.)

Analogy: the appendix and data folder attached to a report, versus the calculator.

Why: grounds answers in the paper's real content — and it is the part that still works when the code can't run.

MCP prompt

What: a named, reusable instruction template stored on the server, with parameters (e.g. `data_path`). In Claude Code the user picks it like a slash command (Fig. 3b shows `/scanpy:preprocess_and_cluster_scanpy`).

How it differs from typing an instruction yourself: (1) written once, from the paper and code, not re-typed by each user; (2) it names the exact tools on this server, in the right order, with sensible parameter ranges; (3) it ships and is versioned with the tools, so everyone runs the same workflow — that is the reproducibility gain; (4) the user doesn't need to know the method to get a good result.

Analogy: a laminated standard operating procedure next to a machine, versus each new intern improvising from memory.

Still only text: the agent reads and follows it. It is not code, so the agent can adapt — the Scanpy prompt tells it to inspect the data first and change defaults only with good reason.

OUTSIDE MCP's design gives each part a different "controller": **tools** — the model decides when to call them; **resources** — read into context by the app or agent; **prompts** — chosen by the user.

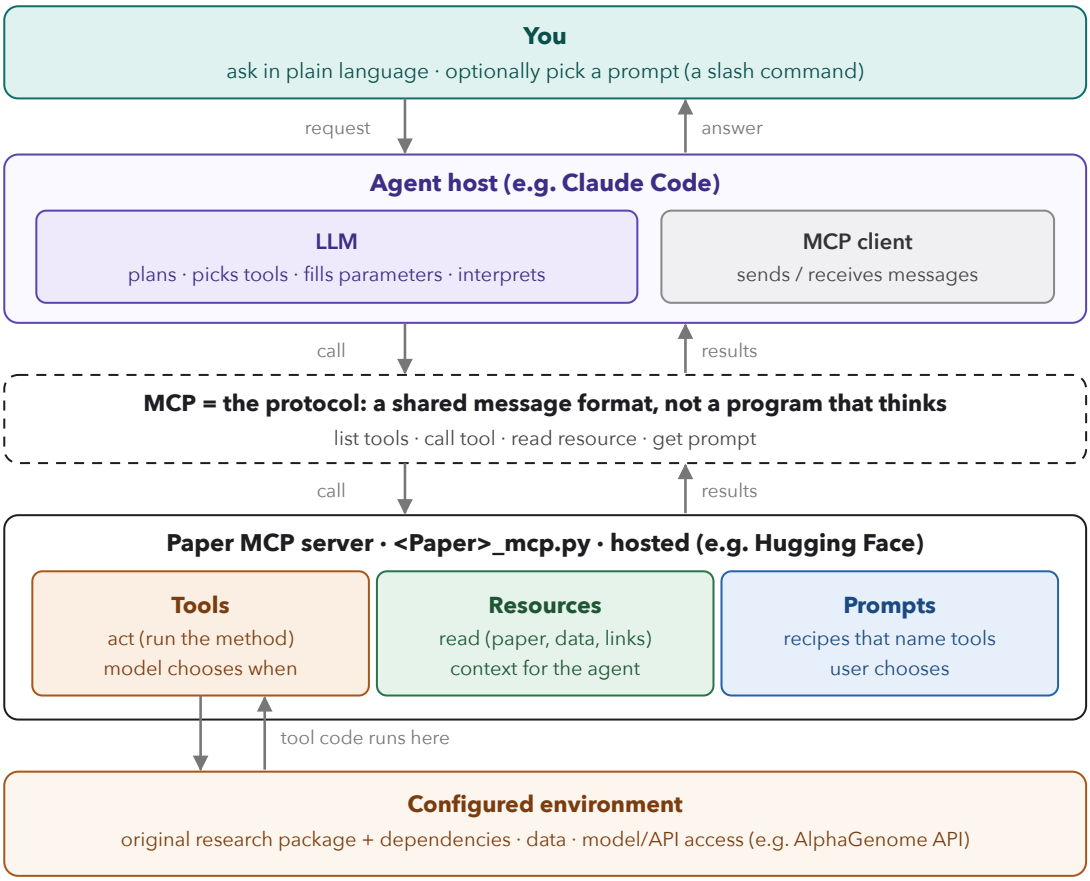


Figure B · Your layer diagram, corrected. Results travel back up the same path; that returned data is what the LLM reasons over.

Your interpretation is slightly off: your diagram is close, but: (1) MCP isn't a layer that does work — it is the agreed message format between the agent's MCP client and the paper's server; (2) the LLM sits inside an agent host (Claude Code), which also contains the MCP client; (3) information flows both ways — the results that come back are what the agent reasons about; (4) prompts don't sit "above" the code — they are recipes that point to tools.

A5 · Figure 1, part by part (p. 3)

Figure 1a

COMPONENT	WHAT IT IS	WHAT IT DOES	WHY IT EXISTS
Input: papers	The paper plus linked code, supplements, data	Source material	The method lives in code, not only in text
Paper2MCP	The builder: a multi-agent system on Claude Code	Reads paper and repo; builds the server	Automates setup work that takes a skilled person days
`<Paper>_mcp.py`	The server file for one paper	Registers tools, resources, prompts	One standard, deployable unit per paper
Tool 1 … Tool K	K validated functions (22 for AlphaGenome, 7 for Scanpy)	Run the method	Tested units the agent can call
MCP resources (green)	Manuscript, code-repository link, supplements and data	Looked up for context	Grounding; useful even with no tools
MCP prompts (blue)	"Instructions for scientific task 1, task 2, reproducing a figure"	Guide multi-step workflows	Correct order without expert prompting
Stacked sheets	One server per paper	—	Many papers → many servers
Remote server → Hugging Face	Hosting (Hugging Face Spaces)	Runs the server online	Users don't install dependencies
"Connect to any agent or LLM without setup"	The MCP connection	The agent discovers the tools automatically	Standard plug
<Paper> Agent	Chat agent + this server	Answers queries, runs tools	What the user actually talks to
User query	e.g. "Apply this paper's method to my dataset"	Starts planning and tool calls	Plain-language use is the goal

The chart inside the agent's reply is a GWAS "Manhattan plot" — just an example output; you don't need to read it.

The `.py` question

PAPER Yes — the server is packaged as an MCP Python file. In the Methods, step 5 turns each executed tutorial into a standalone Python module of reusable functions and marks each function as an MCP tool; step 6 merges all validated modules into one server with a manifest, versioning and basic security defaults. You can see the Python definitions in the figures: Fig. 3b defines a prompt with `@clustering_mcp.prompt`; ED Fig. 1D defines a resource with `@mcp.resource(...)`.

MY READING The file does not replace the original repository. The tool functions follow the tutorial code, and the tutorial code calls the original package, which is installed in the configured environment. **PAPER** Each tool also carries a link to the original source code it came from, for traceability.

Tool names differ slightly across the paper (`score_variant`, `score_variant_effect()`, `score_variant_batch()`; `quality_control()` vs `quality_control_basic_filtering()`). Don't get stuck on this.

Figure 1b – the workflow, mapped to the Methods

FIGURE 1B LABEL	WHAT HAPPENS	METHODS STEP
Paper → identify the codebase	Find the repository from the paper, its references or supplements; clone it.	1
Environment agent → configured environment	Isolated environment; dependencies installed until the code runs.	2

FIGURE 1B LABEL	WHAT HAPPENS	METHODS STEP
Extraction agent → implemented tools	Find tutorials, run them, turn general steps into functions.	3, 4, 5a
Testing agent ↔ Refine	Test each function; diagnose failures; fix code *and* environment; repeat.	5b
MCP server Python file	Package validated tools into one server.	6
Remote server → Hugging Face	Deploy online.	—
Connect with AI agent → Paper agent	A chat agent connects; users start asking.	—

A6 · The build pipeline (Methods, pp. 10–12)

1 Locate code → 2 Environment → 3 Find tutorials → 4 Run tutorials → 5a Extract tools → 5b Test + fix → 6 Assemble server

PAPER An **orchestrator** agent coordinates sub-agents. Each sub-agent is a separate Claude session with its own role prompt and a set of allowed actions (read/write files, run shell commands, search code). Steps run in order; within a step, several tutorials can be handled in parallel. Steps hand work to each other through JSON reports and file conventions.

STEP	INPUT	WHAT HAPPENS	OUTPUT	WHY IT MATTERS
1 · Locate and download	The paper (or a URL from the user)	Find the repo in the text, references or supplements; clone it with supplementary data and config files.	Cloned repo; detected language	Nothing runs without the code; the wrong repo gives the wrong tools.
2 · Environment setup	Cloned repo	Environment manager creates a clean, isolated environment and installs dependencies until the code runs without conflicts.	Working environment; test config	Dependency failures are a main reason papers fail; the environment ships with the tools.
3 · Tutorial discovery	Repo (+ optional filter)	Tutorial scanner separates real tutorials and examples from other files and ranks usefulness.	JSON index of candidate tutorials	Tutorials show intended use and come with example data.
4 · Execution and audit	Tutorials, environment, index	Tutorial executor runs them end to end, fixes execution errors, saves every output, notes hidden assumptions.	Executed notebooks; execution reports	Produces the reference ("gold-standard") outputs and proves the original code works.
5 · Extract, test, refine	Executed notebooks, environment, index	Extractor–implementor writes single-purpose functions (parameters, file-based I/O) and marks them as MCP tools. Test verifier–improver writes per-function tests from the tutorial data, runs, fixes, drops failures.	Tool modules; tests; logs; summaries	This is where reliability comes from.

STEP	INPUT	WHAT HAPPENS	OUTPUT	WHY IT MATTERS
6 · Server assembly	Validated modules	Orchestrator combines them into one server with a manifest, versioning and basic security defaults.	Deployable MCP server	One standard package any agent can use.

The specialised agents, in one line each

Orchestrator	Dispatches the sub-agents step by step, runs work in parallel where possible, records each step for traceability. A project manager.
Environment manager	Solves “works on my machine”: reads setup requirements, builds an isolated workspace, installs everything, checks the code runs.
Tutorial scanner	Looks for genuine tutorials and worked examples (not tests, configs or docs) and reports which are worth turning into tools.
Tutorial executor	Runs the chosen tutorials, fixes execution errors, keeps every output. *Why run instead of read?* Reading tells you what the code should do; running tells you what it actually does — and produces the reference answers.
Extractor–implementor	Finds tutorial steps that generalise beyond the example data; writes each as a clean function with clear inputs, outputs and defaults; replaces hard-coded paths, thresholds and column names with parameters; saves results and figures to files.
Test verifier–improver	Writes tests using only the tutorial's own examples, runs them, diagnoses failures, applies fixes; after six failed attempts, removes the function from the server.

> **Extracting vs validating.** **Extracting** = writing a function that should reproduce a tutorial step. It is a *claim*.
> **Validating** = running that function on the tutorial's data and checking it gives the tutorial's actual outputs. It is *evidence*.
> Analogy: rewriting a colleague's spreadsheet into a cleaner model (extracting), versus checking the new model gives the same totals as the old one on last month's data (validating).

A7 · Testing – why tutorial outputs are the reference

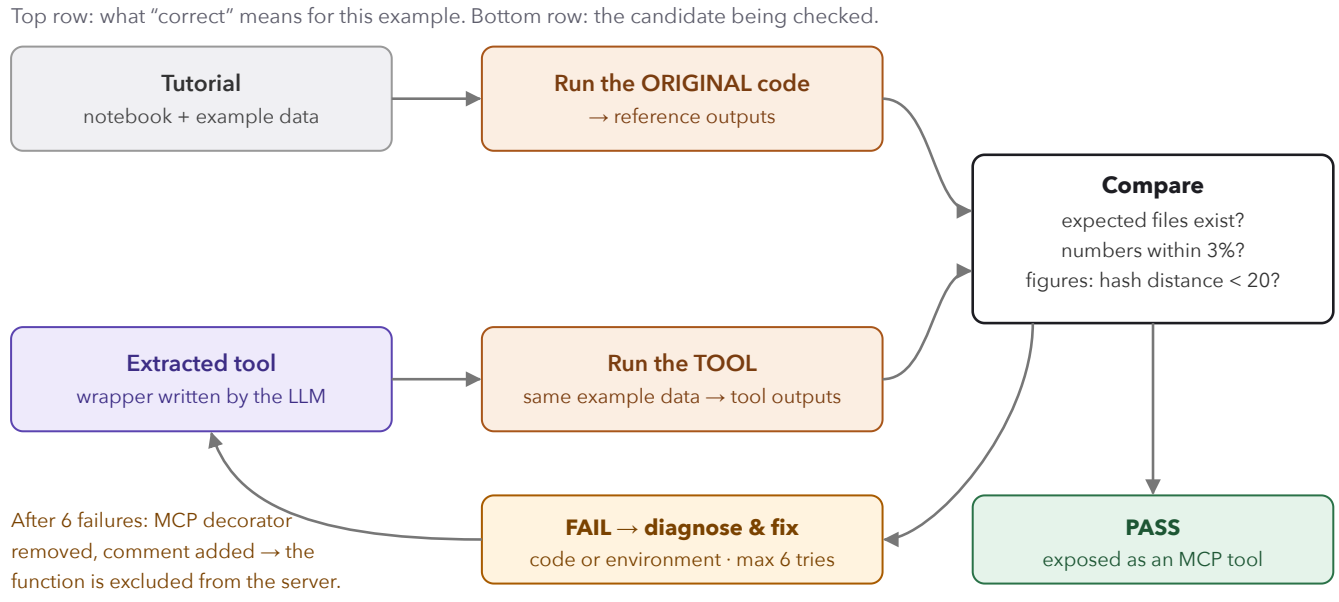


Figure C · The test loop inside step 5. The comparison is deterministic; writing and fixing the wrapper is LLM work.

- **Why run the original first:** the original code on the example data defines what "correct" means for this test — not the LLM's opinion. An LLM-written wrapper can silently change a default, skip a normalisation step or use the wrong column; comparing with the original outputs exposes that.
- **Expected files exist:** did the function produce its outputs at all?
- **Numbers within 3%:** allows tiny floating-point differences (different hardware, library versions, order of operations) but flags real ones.
- **Perceptual hashing, Hamming distance < 20:** each figure is reduced to a short "fingerprint" of its visual structure. The Hamming distance counts how many positions differ between two fingerprints. Under 20 → treated as the same figure, even if exact pixels differ (fonts, anti-aliasing).
- **Six attempts, then exclusion:** a tool that can't reproduce a known answer can't be trusted on an unknown one. The agent can't call what isn't exposed. A smaller toolset you can trust beats a larger one you can't.
- **Why this beats "does this code look correct?":** looking right isn't behaving right. An LLM reviewer checks plausibility and shares the writer's blind spots. Execution checks behaviour against real outputs.

Think. What does a pass *not* tell you? (How the tool behaves on inputs unlike the tutorial; whether the tutorial itself was right; whether 3% is tight enough for your use.) The paper also doesn't say how long the image fingerprint is, so "< 20" is hard to interpret on its own.

A8 · Query time – Planning → Action → Observation → Finding (Fig. 2d)

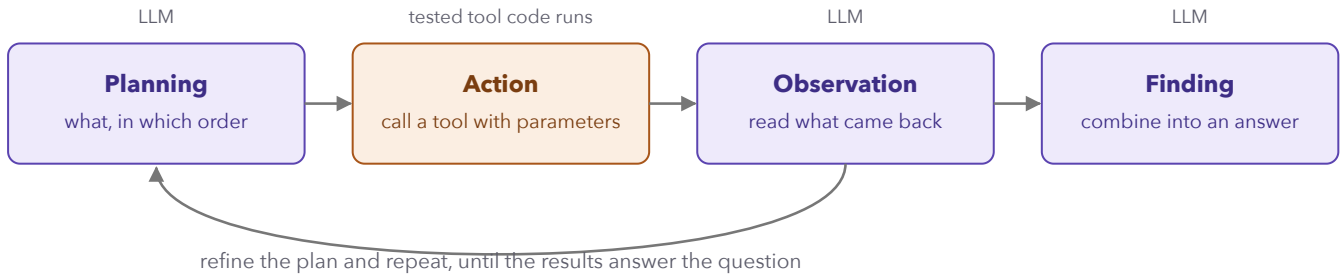


Fig. 2d: plan (score variants, visualise in liver, …, write report) → score_variant_batch(), visualize_variant_effects() → tables + tracks → SORT1 report.

Figure D · The loop. This is the standard reason-then-act agent pattern (ReAct, which the paper cites).

STAGE	WHAT HAPPENS	AGENT PRODUCES	TOOL CALLED?	EFFECT ON NEXT STEP
Planning	Reads the request and the list of available tools; writes a step list.	A to-do list	No (may read resources)	Decides the first action
Action	Calls a tool with chosen parameters. The tool returns tables, numbers, file paths, figures.	A tool call	Yes	Produces data to look at
Observation	Reads what came back: did it work, what stands out, what's missing?	Notes, interim conclusions	No	Triggers refinement: filter tissues, change window, call another tool
Finding	Combines the observations.	A report with figures	No	Ends the loop — or the user asks a follow-up

A non-biology example

```
User: In survey.csv, which variables are associated with customer satisfaction?
Planning:    1) inspect columns and missing values  2) clean  3) fit a model  4) check robustness
5) report
Action:      inspect_data("survey.csv")
Observation: satisfaction is a 1–5 rating; wait_time has 12% missing; region is categorical
Action:      clean_data(max_missing=0.2, impute="median") → fit_model(target="satisfaction",
model="ordinal")
Observation: wait_time and first-contact resolution have large, stable effects; region does not
Finding:     longer waits and unresolved first contacts go with lower satisfaction (association, not
causation)
Next action: does the wait_time effect differ by channel? → fit_model(...,
interaction="wait_time:channel")
```

Why the loop helps: the agent reacts to what the data actually shows (missing values, variable types, errors) instead of running a fixed script blindly, and problems become visible mid-way.

PART B · THE EVIDENCE, FIGURE BY FIGURE — KEEP THE PAPER OPEN

B1 · Biology decoder (just enough)

> **One analogy.** DNA is a huge instruction manual (about 3 billion letters: A, C, G, T). Every cell carries the same manual but reads different chapters. A **genetic variant** is a one-letter difference between people's copies. **AlphaGenome** is a model that reads a stretch of the manual and predicts what lab measurements would show — how much each chapter is read, which pages are open — *with and without the changed letter*. The difference is the variant's predicted effect.

TERM	PLAIN MEANING	IN THE ANALOGY
DNA	The genetic code	The manual
Gene	A stretch of DNA that is the recipe for a product (usually a protein)	A chapter
Genetic variant	A position where people's DNA differs	An edit in one copy
Mutation	A DNA change; often used for new or rare changes. “Variant” is the neutral word.	—
REF / ALT	The letter in the standard reference genome / the alternative letter	Original vs edited letter
Chromosome	One of 23 DNA packages	A volume
Genomic position	`chr1:109274968:G>T` = chromosome 1, position 109,274,968, G changed to T	Volume, page, letter
Regulatory effect	A change in how much, when or where a gene is used — not in the recipe itself	Editing the margin note “read this chapter more”
Gene expression	How actively a gene is being used (how many copies of its message a cell makes)	How often a chapter is read
Chromatin accessibility	Whether a DNA region is open (readable) or packed away	Page open or glued shut
RNA-seq	Lab measurement of gene expression	Counts how often each chapter is read
ATAC-seq (and DNase)	Lab measurement of which regions are open	Which pages are open

TERM	PLAIN MEANING	IN THE ANALOGY
ChIP-seq	Lab measurement of where specific proteins or chemical marks sit on DNA (e.g. histone marks)	Sticky notes on pages
Tissue / cell type	Liver cell, T cell, neuron … same DNA, different usage	Different readers
GWAS	A study that tests millions of variants across many people for statistical association with a trait	A very large association screen
Causal gene	The gene through which the variant actually affects the trait	The chapter whose change really matters
eQTL / GTEx	A variant associated with a gene's expression level; GTEx is a public atlas of these across tissues	"When this letter changes, this chapter is read more or less"

Why GWAS leaves a puzzle: nearby variants are inherited together, so a GWAS finds a region and a set of correlated suspects. It can't say which variant — or which nearby gene — is causal. AlphaGenome's predictions are one way to narrow it down.

B2 · AlphaGenome agent (pp. 3-5, Fig. 2)

Figure 2a – construction

PART	MEANING
`score_variant` tool	Compares predicted readouts for the REF vs ALT sequence across measurement types (expression, splicing, accessibility …) and tissues; returns a table of scores.
`visualize_variant_effects` tool	Plots predicted REF vs ALT signal along the genome. Options include the window length around the variant and which measurement types to show (Supplementary Fig. 1).
All 22 tools	Single and batch scoring, sequence prediction, tissue lookup, visualisation. All passed validation. About 45 minutes, US$14, on a laptop, no human intervention.
Links to training data resource	Lets the agent answer "what was the model trained on?"
Chain of tools for interpreting GWAS loci prompt	Sits in the blue prompts box — see below.
AlphaGenome MCP → agent	Server + Claude Code. Example in the figure: the user gives a designed DNA sequence; the agent reports very low predicted RNA-seq signal, i.e. little promoter/enhancer activity.

> **What "chain of tools" means here.** PAPER It is drawn in the MCP prompts box, so it is an **MCP prompt**: a written recipe describing the order of tool calls for a task. In Fig. 2d the agent then executes the tools itself, step by step (generate inputs → score variants → filter to trait-relevant tissues → visualise → write report), refining as it observes results.
>
> So the answer to "Tool A → Tool B → Tool C?" is **both**: A → B → C is the order the recipe suggests, and the agent makes each call itself, one at a time, and can adjust. It is not a hard-coded pipeline. Analogy: the prompt is the recipe, the agent is the cook, the tools are the appliances.

The benchmark – what it is actually asking

"Given the same questions, does an agent with pre-built, validated tools get the right answer more often, faster and more consistently than (a) the same LLM handed the raw repository, and (b) a general biomedical agent — on known task types and new inputs?"

ITEM	MEANING
Tutorial-derived queries (15)	The same kind of task the tutorials cover, e.g. score a given variant with ATAC-seq predictions in motor neurons and report its quantile score.

ITEM	MEANING
Novel queries (15)	Same form, new inputs (another variant, cell type, assay). MY READING "Novel" means new inputs, not new kinds of task.
Open-ended queries (30)	Need a plan, several tool calls and a biological conclusion (e.g. likely mechanism and causal gene for a bone-density variant).
Ground truth	Humans wrote and ran the code themselves. For open-ended questions: a predefined rubric of key items (right gene, variant, tissue, conclusion).
Claude + Repo	Claude Code with a local copy of the AlphaGenome repository. It must write and run its own code and is told not to copy answers from the tutorials.
Biomni	A general-purpose biomedical AI agent from Stanford (API version), pointed at the repository and the API key.
Paper2Agent	Only the MCP tool definitions and the query — no manuscript, no repository.
Human grading	Two independent experts with predefined rubrics. They gave the same grade 96.7% of the time.
Independent runs	Each query was run five times, because LLM outputs vary from run to run.

Results in words. On exact tasks, Paper2Agent was right 98.7% (tutorial) and 100% (novel) of the time, against 82.7% / 78.7% for Claude + Repo and 37.3% / 56.0% for Biomni. On open-ended questions: 82.7% vs 56.7% (Claude + Repo) vs 72.2% (Biomni). The gains held under reworded prompts and against a newer model for the Claude + Repo baseline (Supplementary).

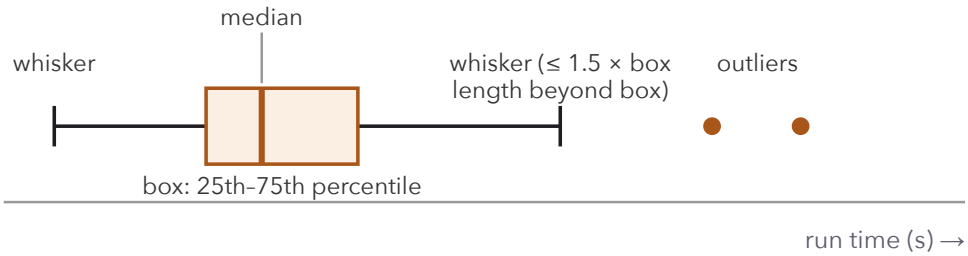
What it tells you: pre-built, tested tools remove the step where a general agent writes — and gets wrong — its own code. Notice that Biomni does much better on open-ended synthesis (72%) than on exact numbers (37–56%): general agents can reason, but exact execution is where they slip.

Why two graders: open-ended answers need judgement; independent agreement shows the grades aren't one person's opinion. 96.7% means the two gave the same grade on 96.7% of items (C3 explains why that number needs context).

Figure 2b – how to read the bar chart

- **x-axis:** the method, grouped by benchmark (tutorial-based on the left, novel on the right). **y-axis:** % of queries answered correctly.
- **Bar** = mean over 5 runs. **Each dot** = one run's accuracy. **Error bar** = ±1 standard error of the mean across the 5 runs.
- **Why five runs:** the same query can get a different answer on a different run; five runs show consistency.
- **"98.7 ± 1.3%":** 15 queries × 5 runs = 75 attempts. If four runs got 15/15 and one got 14/15, the mean is 98.7% and the s.e.m. is 1.3 — exactly the reported numbers. So: **one miss in 75 attempts**. "100.0 ± 0.0" = 75 of 75.
- Left panel: ground truth −0.0203067882, agent answer −0.0203067882 — identical to 10 digits. It ran the same code; it did not estimate.
- **What it says about reliability:** on these task types, the agent is accurate and consistent across runs. What it doesn't say: how it does on other kinds of task — and the ± reflects run-to-run noise only, not which 15 questions happened to be chosen (C3).

Figure 2c – the box plots (runtime)



- **Each dot in the paper's figure** = one query run (75 per box: 15 queries × 5 runs).
- **Median** (centre line): half the runs were faster, half slower. **Box**: the 25th to 75th percentile — the middle half of runs. **Whiskers**: the furthest runs within 1.5 × the box length from the box. **Dots beyond**: outliers.
- **The ratios are ratios of medians.** On tutorial queries, Claude + Repo's median time is 1.9× Paper2Agent's and Biomni's is 3.1×; on novel queries, 2.9× and 3.8×. Read roughly off the figure: about 50 s vs 100 s vs 170 s on tutorial queries.

Your interpretation is slightly off: Paper2Agent still has outliers. On tutorial queries a few runs took roughly 250–370 s — several times its own median, and as slow as a typical Biomni run. What clearly differs is the lower median and the much narrower box. Also, outliers are defined relative to each method's own box: a tight box flags a 150 s run as an outlier for Paper2Agent, while the same time would sit inside Biomni's box. The paper only makes claims about medians.

Why runtime matters: agents are used interactively and many times; every extra minute is also extra LLM calls, i.e. cost. It's faster because it calls a ready tool instead of reading the repository, writing code and debugging it. In the large-scale test: US$0.20 and 1.6 min per query, vs US$0.38 and 4.3 min.

Think. The runtime comparison leaves out the one-time build (~45 min, ~US$14). When does building pay off? (Roughly: at the large-scale saving of ~US$0.18 per query, after about 80 queries — ignoring the accuracy difference, which matters more.)

Figure 2d and the SORT1 result (pp. 4–5)

The question: why is the variant `chr1:109274968:G>T` associated with LDL ("bad") cholesterol? The agent plans, runs batch scoring and visualisation, observes liver tracks (CAGE, RNA-seq, splice sites; the genes CELSR2, PSRC1 and MYBPHL are in view), and reports SORT1 as the causal gene.

QUESTION	ANSWER
What is SORT1?	A gene near the variant. Its protein (sortilin) is involved in how the liver handles and releases cholesterol-carrying particles (LDL/VLDL).
CELSR2 and PSRC1?	Two neighbouring genes in the same stretch of DNA.
"Causal gene" here	The gene whose change in liver activity actually drives the LDL association.
Why the agent chose SORT1 PAPER	(1) Quantile score 0.99983 for SORT1 expression in liver — an extreme predicted effect. (2) SORT1's known role in LDL/VLDL secretion.
Why the original paper emphasised CELSR2 and PSRC1	PAPER Their AlphaGenome quantile scores are even higher: 0.99998 each.
Why it's hard to be sure PAPER	The authors checked GTEx: in liver, the variant is a significant eQTL for *all three* genes (SORT1 P = 1.1×10^{-65}; CELSR2 P = 4.7×10^{-46}; PSRC1 P = 8.5×10^{-50}). One variant moving several neighbouring genes together means statistics alone can't separate them.

Quantile score, simply: where this variant's predicted effect ranks against a large background of other variants. 0.99983 ≈ more extreme than 99.98% of them. It is *relative*: a small absolute change can still rank as extreme, because most variants do almost nothing. It can be negative (−0.0203 in Fig. 2b): read it as signed — near 0 is ordinary, near ±1 is extreme.

Your interpretation is slightly off: you wrote that the agent used the paper's method to produce a different interpretation, which was then checked against more evidence.

1. By the method's own numbers, SORT1 was **not** the top gene — CELSR2 and PSRC1 scored higher. The agent's preference came from combining a high score with background biological knowledge (sortilin's known role). The difference comes from the agent's *interpretation*, not from the method.
2. The extra check (GTEx) was done by the **authors**, not the agent — and it didn't settle the question: it supports all three genes.

A more accurate version: the agent re-opened a published interpretation, and the authors show the question is genuinely ambiguous. The paper presents this as a strength (re-checking published conclusions with one prompt). MY READING It also shows the LLM's own prior knowledge entering a "tool-based" answer in a way that is hard to audit.

Think. In Fig. 2d the findings list log2FC = 0.058 for SORT1. log2FC is normally ALT vs REF, so +0.058 is a small *increase* (about 4%) — yet the text says "strong expression downregulation" and "reduced SORT1 expression". Either the sign convention is different, or the narrative doesn't match its own number. The paper doesn't discuss this. Raise it as a question, not an accusation.

OUTSIDE – CHECK BEFORE SAYING IT ALOUD This variant is widely known as rs12740374, and earlier lab work (Musunuru et al., *Nature* 2010) linked it to SORT1 in liver. If so, the LLM has very likely seen that literature — which makes "the agent found SORT1" less independent than it looks.

B3 · Scanpy (pp. 5-6, Fig. 3)

The name: you're right to question it. **Scanpy** ("Single-Cell Analysis in Python") is a Python package, used as `import scanpy as sc`. "scan.py" is not its name. It is a standard toolkit for single-cell gene-expression data: quality control, normalisation, dimensionality reduction, clustering, UMAP plots, marker genes. The paper agentified its most common workflow: preprocessing and clustering.

TERM	JUST ENOUGH
Single-cell data	Measurements for each individual cell, not an average over a tissue sample.
Single-cell RNA-seq	For each cell, counts of each gene's messages.
Cell	The unit. Different cell types (T cells, B cells, monocytes …) use different genes.
Gene expression	The count per gene per cell.
Preprocessing	Quality control (drop broken or dying cells, "doublets" where two cells were captured as one, and near-empty genes) → normalise for sequencing depth → log-transform → keep the most informative genes → PCA.
Clustering	Grouping similar cells.
Leiden clustering	Link each cell to its most similar cells (a graph), then find tightly connected groups. The "resolution" setting controls how many groups.
UMAP	A 2-D picture of the cells: nearby points are similar cells. Distances between far-apart groups mean little.
Cell-type annotation	Naming each cluster from its marker genes (e.g. MS4A1 and CD79A high → B cells).

A data analogy

Rows = cells, like thousands of customers. **Columns** = ~20,000 genes, like products. **Values** = counts, like how many of each product a customer bought.

Clustering = customer segments. **Annotation** = naming each segment from its signature products — that's the marker-gene dot plot.

Why preprocessing: drop empty or bot accounts (low-quality cells) and merged duplicate accounts (doublets); drop products nobody buys; convert to shares so heavy buyers don't dominate (normalisation); keep products whose buying varies most (highly variable genes); compress (PCA). Skip this and your "segments" form around who buys more overall, or around junk accounts.

QC → normalise → select features → PCA → neighbour graph → Leiden clusters → annotate cell types

Figure 3a – the Scanpy MCP

- `quality_control_basic_filtering()`: calculates quality metrics (genes per cell, total counts, % of mitochondrial reads — a high % suggests dying cells), applies filters, and writes the filtered data plus the violin plots shown. The agent reports 17,041 cells and 23,424 genes kept.
- `clustering_analysis()`: runs Leiden clustering at several resolutions (coarse to fine) and saves labels and plots.
- Both execute Scanpy's own functions inside the configured environment. MY READING i.e. Scanpy's standard calls for QC metrics, filtering and Leiden — the paper doesn't list the calls.
- Resource: a link to the Scanpy documentation. Prompt: instructions for preprocessing and clustering. Scanpy agent = Claude Code + this server. 7 tools, all passed; about 45 minutes; US$13.

Figure 3b – the chain of tools, checked against the figure

PAPER The left panel is a Python function in the server, marked as an MCP prompt (`@clustering_mcp.prompt`, named `preprocess_and_cluster_scanpy`, taking `data_path`). It returns a text recipe:

- First inspect the data (size, organism, batch columns, quality). Change defaults only with a strong reason.
- Then run, in order: `quality_control` → `normalize_data` → `select_features` → `reduce_dimensionality` → `build_neighborhood_graph` → `cluster_cells` → `annotate_cell_types`.
- Each step comes with parameter guidance (e.g. 2,000–3,000 variable genes; 10–30 neighbours; resolution 0.1–0.4 for broad or 0.6–1.5 for fine clusters), plus "validate each step".

On the right, the user runs `/scanpy:preprocess_and_cluster_scanpy` in Claude Code and gives `data.h5ad`; the agent executes the pipeline and summarises the results.

> **Which of your four options? Option 4 — a combination.**
>
> - **(2)** The prompt fixes the order and gives guidance.
> - **(3)** The agent makes each tool call itself and picks parameters within the guidance after inspecting the data.
> - The tools themselves are fixed, tested code.
> - **Not (1):** there is no hard-coded function that runs all seven steps.

Your interpretation is slightly off: it is not a Markdown file telling the agent which `.md` files to use. It is a prompt template served by the MCP server and defined in Python. Its text is formatted like Markdown (numbered steps, bold tool names), which may be why it looks like one — and it points to *executable tools*, not to other documents. Keep this separate from the ablation "Markdown skill files instead of MCP tools", which replaced the tools themselves with text instructions.

PAPER How it was made: the authors asked Paper2Agent, in one sentence, to build a prompt that replicates the tutorial in the correct order, inspects the data first and only departs from defaults when needed. The steps themselves were inferred from the code and tutorial, not written by hand.

Figure 3c – agent vs human researcher

ROW OF THE FIGURE	WHAT IS BEING COMPARED
Highly variable genes (scatter plots)	Which genes were selected as informative — feature selection.
UMAP coloured by Leiden cluster and by batch	The cluster structure, and whether the three samples ("batches") mix — clustering.
Dot plot of marker genes per cluster	Which genes mark which cluster — the basis for naming cell types.

Left: the agent, given only a file path. Right: a human following the official Scanpy tutorial on the same blood-cell (PBMC) data. PAPER "Reproduces human results" means: the same numbers of cells and genes after quality control, and equivalent top marker genes per cluster, with matched parameters, on public datasets not included in the Scanpy codebase.

> **Discovery or workflow?** **Workflow.** The biology here (blood cell types) is well known. What is shown is that the agent picks the right steps, order and parameters and executes them correctly with only a file path. MY READING UMAP pictures looking alike is weak evidence on its own; matched cell counts and marker genes are the real test.
>
> On seven datasets the agent adjusted parameters to the data (Supplementary Table 2). Who decides whether an adjustment was sensible?

B4 · TISSUE (p. 5; Extended Data Fig. 1, p. 14)

TERM	PLAIN MEANING
Spatial transcriptomics	Measuring gene activity while keeping where each cell sits in the tissue.
Single-cell spatial data	Per-cell measurements plus location — but only for a limited panel of genes. The rest are *predicted* from a separate single-cell dataset that measures all genes but loses location.
TISSUE	A method (Sun et al., *Nature Methods* 2024 — from the Paper2Agent senior author's lab) that attaches calibrated uncertainty (prediction intervals) to those predicted values, and uses it in later analysis: hypothesis testing with multiple imputation, filtering unreliable cells, weighted PCA.
Uncertainty estimation	Giving each prediction a range, and checking the ranges are honest (true values fall inside a 95% interval about 95% of the time).

Analogy: you have complete purchase histories for a sample of customers but not their locations; for every store you know sales of only 50 products. You predict the other products' sales per store from similar customers. TISSUE attaches an error bar to every predicted number and makes later comparisons respect those error bars, so shaky predictions aren't treated as facts.

What Paper2Agent did: built a TISSUE MCP — tools such as `calibrate_uncertainties_and_prediction_intervals()` and `multiple_imputation_hypothesis_testing()`; resources = the spatial datasets used in TISSUE; prompts = instructions for uncertainty-aware analysis — connected it to Claude Code, and compared its outputs with those of human researchers who followed the TISSUE GitHub tutorial on the same mouse brain dataset (somatosensory cortex).

How to read Extended Data Fig. 1

PANEL	WHAT TO LOOK FOR
A	Construction, same pattern as the others. The example asks for uncertainty-aware dimensionality reduction; the agent returns a PCA figure.

PANEL	WHAT TO LOOK FOR
B	Q&A: asked what TISSUE can do, the agent lists six functions with their inputs, outputs and file formats. This is the "virtual corresponding author" role — explaining how to use the method.
C	Reproducibility: a map of prediction-interval width for one gene (Acta2), from the agent and from humans — same pattern, same colour scale. *Reproducible here* = same data + same method → same output as humans running the official tutorial.
D	Structured resources: a dataset registry (ID, species, tissue, URLs) and a resource that filters it by species. Asked to download the paper's data, the agent calls the Zenodo API and fetches it. **Why it matters:** finding and downloading the exact data a paper used is one of the most common blockers to reproducing it.

B5 · Paper agents working together (pp. 7–8, Fig. 4)

Agent 1 · AlphaGenome

Evidence: a prediction. For the psoriasis-linked variant rs887314, predicts which nearby gene's activity in CD4+ T cells (an immune cell) is most affected → **GPR137** (quantile score 0.997, top of the region).

Agent 2 · MPRA-coupled scCRISPRi

Evidence: an experiment. Researchers switched off the DNA region containing the variant (a "CRE" — a control switch) in T cells and measured how ~20 downstream genes changed. → the region's **fingerprint**.

Agent 3 · Perturb-seq (CD4+ T cells)

Evidence: another experiment. Researchers knocked down genes one at a time and measured how other genes changed, under three conditions. → each candidate gene's **fingerprint**.

How they combine: if the region works through gene X, switching off the region should look like knocking down X. So correlate the region's fingerprint with each candidate's fingerprint. Only GPR137 matched — and only when the cells were stimulated.

Analogy: a feature flag seems to hurt a metric, and you suspect it acts through one of five features. Log 1: what happens to 20 metrics when the flag is off. Log 2: what happens to the same 20 metrics when each feature is disabled. The feature whose pattern matches the flag's pattern is the likely route.

Psoriasis in plain English: an immune-driven skin disease in which activated CD4+ T cells are involved. A GWAS linked rs887314 to psoriasis risk; the question is which gene it acts through.

Who did what

HUMAN	Asks: what is the causal gene and mechanism for rs887314 in CD4+ T cells?
AGENT	Uses AlphaGenome tools to rank nearby genes → GPR137 on top.
HUMAN	Instructs: verify using the scCRISPRi and Perturb-seq data.
AGENT	Reads both papers, their supplementary tables and summary statistics; proposes **ten** validation strategies.
HUMAN	Selects one: signature correlation.
AGENT	Computes Spearman correlations for the five top candidates that have knockdown data, in three conditions; reports that GPR137 matches.
AUTHORS	Interpret: GPR137 is the *probable* causal gene, and its role is activation-dependent.

PAPER The signature-correlation idea was not in either source paper; the agent proposed it (as one of ten), and a human chose it. Note the language gap: the agent's reply in Fig. 4a says the result *confirms* GPR137 as the causal gene; the authors' text says the evidence *supports* it as the *probable* causal gene. MY READING That gap is exactly why the human stays in the loop.

B6 · Spearman correlation

One sentence: Spearman's ρ measures whether two lists rise and fall in the same order — +1 same order, −1 opposite order, 0 no relation.

Tiny example: five students are ranked in maths as 1, 2, 3, 4, 5 and in physics as 1, 3, 2, 4, 5. Only two swap places, so ρ = 0.9. Because it uses ranks rather than raw values, one extreme value can't dominate the way it can with ordinary (Pearson) correlation.

In the paper: each dot is one downstream gene. A high positive ρ (0.61 at Stim8hr, 0.63 at Stim48hr, for GPR137) means genes that drop more when the region is switched off also drop more when GPR137 is knocked down — the fingerprints match. ρ = 0.29 at rest (P = 0.21) is weak and not distinguishable from chance. For BAD: −0.12, 0.09, 0.05 — no match.

P value: how often a correlation at least this strong would appear by chance if there were no real relationship. With ~20 genes, ρ ≈ 0.3 happens easily by chance; ρ ≈ 0.6 rarely (P ≈ 0.004). **Why they care:** 5 candidates × 3 conditions = 15 tests, so some would look good by luck. They used Benjamini–Hochberg correction; orange labels mean the false discovery rate is below 5%.

Figure 4 – how to read it

ELEMENT	MEANING
x-axis	How much each downstream gene changed (log2FC) when the rs887314 region was switched off.
y-axis	How much the same gene changed when the candidate was knocked down — GPR137 (top row) or BAD (bottom row).
Each dot	One downstream gene (n = 20, 21, 19 across the three columns). The knocked-down gene itself is excluded.
log2FC	log2 fold change: 0 = no change, −1 = halved, +1 = doubled.
KD	Knockdown: reducing a gene's activity (here with CRISPRi).
Rest / Stim8hr / Stim48hr	T cells not re-stimulated (collected at 8 h) / activated with a CD3/CD28/CD2 activator and collected at 8 h / 48 h. Think “immune system idle vs switched on”.
Dashed line	Straight-line fit after removing outliers — a visual guide only.
Orange text	Significant after multiple-testing correction.
Why GPR137	Two independent lines agree: AlphaGenome ranked it top, and it is the only candidate whose knockdown fingerprint matches the region's.

Think. In every panel most dots sit near (0, 0) and one gene sits far to the left. With ~20 points, how much would ρ change without that gene? What would you want to see before calling this “confirmed”?

Found vs selected: the agent ranked genes, read the papers, proposed strategies and ran the analyses. Humans chose the question, chose the validation strategy, and interpreted the “stimulated only” pattern as activation-dependent. The paper says researchers remain responsible for choosing directions and judging evidence. No new experiment was run: the “experimental support” is a re-analysis of existing published data.

B7 · ADHD example (Extended Data Fig. 2, p. 15)

ADHD GWAS data (data MCP) + AlphaGenome (method MCP) → AI co-scientist → candidate variants → mechanistic hypothesis

PART	WHAT IT CONTRIBUTES
GWAS dataset	Regions linked to ADHD and, for each, many correlated candidate variants (locus 27 has 209). It says *where*, not *which* or *how*.
AlphaGenome	The predicted effect of each candidate on splicing and gene activity in relevant brain cells (glutamatergic neurons).
AI co-scientist	Claude Code connected to both servers. It proposed 10 research questions; a human picked one (can AlphaGenome prioritise causal variants among fine-mapped candidates?) and asked for the mechanism. The agent scored all candidates, ranked them, and made the figures.
"Prioritise a causal variant"	Rank the correlated suspects by how likely each is to change biology. Here rs1626703 stands out (splice quantile 1.000, RNA-seq 0.963): it may change how MPHOSPH9 is spliced and raise its activity in these neurons. Results for 39 loci are in the supplement.
Still to validate PAPER	The paper says so explicitly. For example: edit the variant in neurons, measure MPHOSPH9 splicing and activity, then link that to relevant traits.

> **Hypothesis vs discovery.** A **computational hypothesis** is a ranked, plausible guess from a predictive model (which has its own error), plus biology text written by an LLM. A **validated discovery** needs independent evidence that the variant causes the change and that the change matters for the trait. Extended Data Fig. 2 is the first. Fig. 4 sits in between: existing experimental data, re-analysed, with no new experiment. In ED Fig. 2A, the human in the loop is marked "optional".

B8 · Large-scale evaluation (pp. 5, 7, 11–12)

Procedure PAPER: (1) produce ground-truth answers; (2) run the agent non-interactively, capturing the answer, runtime and cost; (3) human graders compare the answer with the ground truth; (4) summarise — mean ± s.e.m., paired t-tests on per-run accuracy, and bootstrap tests with 10,000 resamples for the 100-paper set.

"Ground truth" in this paper = answers obtained by running the original code (by humans for AlphaGenome; by executing the original code and checking against tutorial outputs for the 300 questions), plus rubric items for open-ended questions.

GROUP	WHAT IT IS	WHAT WAS TESTED	RESULT
100 computational biology papers	Sampled backwards from Dec 2025 on bioRxiv (bioinformatics), with no filtering for code quality — a realistic mix	Can it build a validated server with no human help? Do the tools answer 300 tutorial-derived questions correctly?	74/100 built; 593 of 599 proposed tools passed; 91.2% vs 80.3% (Claude + Repo, Sonnet 4) and 86.3% (Sonnet 4.6); cheaper and faster per query
26 data/discovery papers	13 bioRxiv + 13 *Nature* papers from 2025; mostly results and data, little runnable code	Does the resource layer alone help? 100 synthesis questions (e.g. redo an analysis with Spearman instead of Pearson; cross-reference tables and text)	89.0% vs 82.0% for Claude with browser access ($P = 0.03$); 34× cheaper, 15× faster
10 non-biology papers	grf, SAELens, Binoculars, SAM2, TabPFN, GenericML, CausalImpact, Nashpy, emcee, conformal-selection	Does it work beyond biology? 42 execution tasks	98.1% ± 0.8 over 5 runs

Why the third group matters for you: it includes causal inference, Bayesian time series, game theory, MCMC and selective inference — statistics and economics tools. MY READING Caveat: these are popular, well-maintained packages, and 42 tasks is a small number.

How each case study was evaluated

AGENT	EVALUATION	COUNTS AS SUCCESS WHEN
AlphaGenome	15 tutorial + 15 novel + 30 open-ended queries; 5 runs each; vs Claude + Repo and Biomni; human-graded	The answer matches human-computed ground truth / rubric
Scanpy	Agent vs humans following the official tutorial, on public datasets not in the codebase; seven datasets (blood, brain tumour, neurons, heart; 1k–10k cells) to see whether it adapts parameters	Same cell and gene counts after QC; same top marker genes
TISSUE	Agent vs humans following the TISSUE GitHub tutorial, same mouse brain data	Same prediction intervals and figures

"Human researcher results" = outputs produced by people running the official tutorial steps themselves on the same data. The paper also reports, in examples, that agents can support model training and adaptive hyperparameter tuning (Supplementary Note).

What makes a codebase "agentifiable"

PAPER Failures came from missing executable code, missing data or model files, environment/dependency failures, and scripts that don't generalise. Turned around:

- The code is public and complete (not "available on request").
- Models, weights and data can be downloaded.
- The environment can be rebuilt (dependencies resolvable; no dead or licence-only packages).
- There are runnable examples with example data — these become the tests.
- The logic is general (functions with parameters), not one-off scripts hard-wired to one dataset.
- Outputs are checkable (files, numbers, figures), and randomness is controlled.

The authors suggest that how easily a paper can be agentified could itself be a practical measure of its reproducibility.

> **Think.** 593/599 tools passing (99%) sounds high — but 26 papers produced no server at all, and tools the extractor never attempted aren't counted. Which number would you report?

Out-of-scope questions

Why: to measure false positives — does the agent make up answers to questions its paper can't answer? **How** PAPER: questions for the 26 data papers were randomly re-paired with the wrong paper, and the agent was run with and without an explicit instruction to say "I don't know". A **correct rejection** = declining instead of answering. Reported: 100% correct rejection.

Why "I don't know" can be better: in research, a confident wrong answer flows into later work; an honest refusal costs little. **Useful agent:** answers in-scope questions correctly. **Hallucinating agent:** answers everything confidently regardless of evidence. **Correctly refusing agent:** recognises "my paper doesn't cover this".

> **Think (stats framing).** An agent that refuses everything also scores 100% here. You need both numbers — in-scope accuracy (like sensitivity) and out-of-scope refusal (like specificity). And random mismatches are easy to spot; a harder test would pair a question with a similar paper from the same field.

Shortcut learning (memorising the tutorial)

The worry: tools are tested against tutorial outputs, so an LLM could "pass" by hard-coding the tutorial's numbers, fixed file paths, cached outputs, or logic that only works on the tutorial data — and then fail on new data. Like a student memorising the practice exam.

The check PAPER: manual inspection of generated servers across the 100 papers, plus an automated reviewer agent that scans server files for those patterns. No systematic evidence found.

What it tells you: the tools appear genuinely parameterised. MY READING Limits: the reviewer is itself an LLM; "no systematic evidence" isn't "none"; the stronger evidence is correct results on new inputs (100% on novel AlphaGenome queries; Scanpy on new datasets).

Ablations

Ablation study: remove or swap one component, keep everything else the same, and see what changes — it tells you what each part contributes. Run on AlphaGenome, 30 questions, Sonnet 4.

VARIANT	WHAT CHANGES	QUESTION IT ANSWERS
Monolithic agent	One agent does everything in one 200,000-token context	Does splitting work across specialised sub-agents matter?
Non-parallel multi-agent	Same sub-agents, forced to run one after another	How much does parallel work save (mainly time)?
No test verifier–improver	Tools deployed without the test-and-fix loop	Is validation what makes the tools correct?
Markdown skill files instead of MCP tools	Executable tools replaced by text instructions; the agent writes code each time	Is it the locked, executable tool that matters, or just good instructions?
OpenCode instead of Claude Code	A different agent framework	Does the method depend on one vendor's harness?

The main text says only that automated validation and the multi-agent design contribute to performance; the numbers are in the Supplementary Note, which is not in your PDF.

Repository-drift tests

Why: real repositories decay — packages update, paths change, functions get deprecated. The authors injected missing dependencies, broken file paths, typos and deprecated API calls into three repositories (AlphaGenome notebooks, POP-TOOLS as a Python command-line tool, mlearner as an R command-line tool) — 12 set-ups, one error type at a time, not revealed to the agent.

If it recovers: the execute–diagnose–repair loop can fix errors that make code crash, across Python and R, notebooks and command-line tools.

What it does NOT prove: that it catches *silent* errors (code runs but computes the wrong thing) — the authors themselves say the loop fixes bugs that show up as execution failures; that a "fix" keeps the original scientific intent (it could swap in a similar-but-different function); robustness to several errors at once or to larger repositories; that the tutorial outputs themselves were right.

PAPER Also: with every executable tutorial removed from POP-TOOLS (README and source code only), it still built a working server that matched human-run outputs on five tasks. Tutorials help but aren't strictly required.

C1 · Shown / suggested / not yet solved

Demonstrated by the paper

- Automatic building of validated servers: AlphaGenome (22 tools) and Scanpy (7 tools), each in about 45 min for US$13–14; TISSUE also built.
- On its benchmarks: more accurate and faster than Claude + Repo and Biomni (AlphaGenome), and than Claude + Repo on 300 questions.
- Reproduction of human-run analyses (Scanpy, TISSUE).
- Scale: 74/100 biology papers; 593/599 tools passed; 98.1% on 42 non-biology tasks; resource layer at 89%.
- 100% correct rejection on permuted questions; recovery in 12 injected-error set-ups.
- A multi-agent analysis that produced a candidate gene (GPR137), supported by existing data, with human steering.

Suggested (authors' vision)

- Journals adding an “agent availability” section.
- Paper agents as standard research outputs, maintained like code.
- Agentifiability as a measure of reproducibility.
- Re-checking published conclusions systematically, at scale.
- Communities of paper agents linking methods, data and fields.

Not yet solved

- About a quarter of papers can't be agentified.
- Maintenance as code and dependencies change.
- Security, intellectual property, attribution.
- Evaluating open-ended answers where several are defensible.
- Whether scientific conclusions are valid — needs humans and often new experiments.
- Correctness beyond what the tutorials cover.

C2 · Limitations – your list, checked

YOUR LIMITATION	IN THE PAPER?	WHERE / HOW
Not every paper can be agentified	Yes	26 of 100 failed; Discussion.
Code / dependency quality matters	Yes	The failure modes: missing code, data, environments.
Upstream code can change	Yes	Discussion; the repository-drift tests.
Agents need maintenance	Yes	Discussion: treated as part of publishing executable research.
Security / IP / attribution	Yes	Discussion; details in the Supplementary Note.
Open-ended reasoning stays human-in-the-loop	Yes	Stated explicitly in the Discussion.
Benchmark agreement ≠ scientific validity	Yes	Discussion: agreement with one reference measures faithful execution, not analytical validity.

Add these, all grounded in the paper:

- **Validation only covers the tutorial's examples** — the tests use only the tutorial's own examples.
- **The repair loop targets crash-type bugs**, not silent ones.
- **Small benchmarks** — 15 + 15 + 30 AlphaGenome queries, several of them tutorial-derived.
- **Human choices shaped the discovery case studies** (question, strategy, interpretation).
- **One model family** (Claude Sonnet 4) was used for building and for answering.

C3 · A statistician's notes MY READING

These are not claims made by the paper. **Pick two to raise, as questions.**

1. **Small n.** 15 out of 15 correct is consistent with a true success rate as low as about 78% (exact 95% binomial interval ≈ 78–100%). “100%” on 15 questions is encouraging, not conclusive.
2. **What the ± measures.** In Fig. 2b the s.e.m. is across 5 runs of the *same* 15 questions. It captures LLM randomness, not uncertainty about which questions were chosen — the second is probably larger.
3. **Circularity.** Tools were validated on tutorial outputs, then benchmarked on tutorial-derived questions. High scores there are expected; the “novel” queries are the fairer test — and they are new inputs to the same task templates.
4. **Agreement without chance correction.** When most answers are correct, two graders agree often by chance. Hypothetically, if each marks ~85% correct, chance agreement ≈ $0.85^2 + 0.15^2 \approx 0.75$, and Cohen's kappa ≈ $(0.967 - 0.75) / (1 - 0.75) \approx 0.87$ — still good, but more informative than 96.7%.
5. **Conditioning.** The 91.2% is accuracy on the 74 papers that succeeded; it says nothing about the other 26.
6. **Fixed tolerances vs randomness.** A 3% band and a hash cut-off suit deterministic outputs. For random methods (MCMC such as emcee, bootstrap, simulation), identical inputs give different outputs, so a fixed band can fail correct tools or pass wrong ones. The natural test compares distributions.
7. **Wording vs evidence.** The agent says “confirms”; the authors say “probable”. Agent language may not match the strength of the evidence.
8. **Fig. 2d sign** (+0.058 vs “downregulation”) — see B2.
9. **Familiar repositories.** Some test repositories come from the authors' own groups (TISSUE; POP-TOOLS and mlearner). Not wrong, but worth noticing.

C4 · Your 15 questions, answered short

QUESTION	ANSWER
1. What problem?	Using a paper's method means finding code, installing it, configuring it and understanding its inputs — a barrier for most readers.
2. Paper vs paper agent?	A paper describes a method; a paper agent can explain, run and apply it on request, using the paper's own tested code.
3. What is MCP doing?	It is the standard interface that lets any compatible agent discover and call a paper's tools, read its resources and use its prompts.
4. MCP tool?	A tested, callable function that runs part of the paper's method with parameters.
5. MCP resource?	Read-only material — paper text, code link, supplements, datasets, figures — used for context.
6. MCP prompt?	A stored, parameterised workflow recipe that tells the agent which tools to use, in what order.
7. How are tools found?	Scan the repository for tutorials, run them, and turn the generalisable steps into functions with parameters.
8. Why execute tutorials?	To confirm the original code works and to capture reference outputs for testing.
9. How are tools validated?	Per-function tests on tutorial data: files produced, numbers within 3%, figures similar by perceptual hash; up to six repairs; persistent failures excluded.
10. What happens at query time?	The agent plans, calls tools or reads resources, observes the outputs, refines, and reports — without writing new analysis code.
11. Why the loop?	The agent adapts to what the results show (errors, odd data) instead of running a fixed script blindly.

QUESTION	ANSWER
12. Why multi-agent collaboration?	Combining a method paper with data papers gives independent lines of evidence (a prediction plus experiments) that no single paper has.
13. What is agentifiable?	Complete public code, available data and models, a rebuildable environment, runnable examples, general functions, checkable outputs.
14. How was reproducibility evaluated?	Agent outputs compared with ground truth from human-run code, or with humans following official tutorials, over repeated runs, graded with predefined rubrics.
15. Main limitations?	Many papers fail; validation is limited to tutorial examples; maintenance; security and IP; open-ended validity needs human judgement.

C5 · Questions to make you think

A. Five questions for yourself (don't answer them yet)

1. If a tool passes all its tests, what exactly do you know about its behaviour on *your* data — and what don't you know?
2. Why might Claude with the full repository do worse than an agent that can't even see the repository?
3. In the SORT1 answer, which parts came from AlphaGenome and which from the LLM? How would you check?
4. What would a benchmark look like that measures whether a conclusion is *valid*, not just whether it matches a reference?
5. In Fig. 4, if the human had chosen a different one of the ten strategies, could the conclusion have changed? Who gets the credit — and who is responsible if it's wrong?

B. Five possible research questions (not ranked)

1. **Validating random methods.** For methods with random outputs (MCMC, bootstrap, simulation), what test should decide that an extracted tool is equivalent to the original, and how should its tolerance balance rejecting good tools against accepting bad ones?
2. **Benchmark design.** How many questions and repeated runs does an agent benchmark need to separate two agents reliably, and how should variation from question sampling, run-to-run randomness and grader disagreement be modelled together?
3. **Human oversight as a decision.** If expert time is limited, which agent outputs should a human check, and when should an agent stop and ask rather than continue — framed with a cost of checking and a cost of error?
4. **Combining agents' evidence.** When several paper agents supply evidence that may be correlated or biased (a prediction plus two screens), how should it be combined, and when does adding another agent actually add information?
5. **Incentives.** If journals expected an “agent availability” section, how would authors' incentives change — effort on agent quality, favourable framing of their own method, who pays for maintenance — and could agentifiability be gamed as a reproducibility signal?

C. Three limitations worth discussing with the professor

1. **Validation is anchored to tutorial examples.** A pass means “reproduces the original code on the example”, not “correct on new data”. If a tutorial is narrow or wrong, the tools pass anyway.
2. **The evaluation is small and partly circular.** 15 + 15 + 30 AlphaGenome queries, many tutorial-derived; the ± reflects run-to-run noise only; grader agreement is reported without chance correction.
3. **Conclusions have mixed sources.** Final answers blend tool outputs with the LLM's own knowledge and wording (the SORT1 choice, “confirms”), which makes it hard to audit which claims come from the method.

D. Three directions to investigate (ideas, not recommendations)

1. Agentify a set of statistics / operations research / analytics codebases and measure build success, tool pass rate and accuracy on genuinely new tasks, with proper intervals. Does the biology result carry over?
2. For methods whose correctness can be certified — optimisation (feasibility, optimality gap), interval methods (coverage) — test whether certificate-based checks catch injected silent bugs that tutorial matching misses.
3. Study reliance: when people see “validated tools”, do they check less, and do they catch planted wrong answers? How should an agent communicate uncertainty?

E. Five questions you could ask Prof. Keppo

1. “Their tests accept a tool if outputs are within 3% of the tutorial's. For random methods like MCMC or simulation that seems ill-defined. How would you think about validating a stochastic tool?”
2. “They say agreement with one reference measures faithful execution, not analytical validity. How would you evaluate an agent on open-ended questions where several answers are defensible?”
3. “In the psoriasis example the agent proposed ten strategies and a human chose one. Could the split between what the agent decides and what the human decides be studied formally — as a decision under uncertainty?”
4. “If journals required an ‘agent availability’ section, how do you think authors' incentives would change? Could agentifiability become something people optimise for instead of reproducibility?”
5. “What made you choose this paper for me? Is there an angle on it — from operations or decision-making — that you think is under-explored?”

D0 · How to use these

Built from your background — statistics, revenue analytics, conversational AI, MCP and agents — and from Paper2Agent's open questions.

- **They are not ranked.** Each is sized so you could start it alone in 4–6 weeks with a laptop, Claude Code and public data.
- In the meeting, mention **two at most**, as questions: “Would something like this be interesting?” Let his reaction guide you, not your pitch.
- Each idea has a **decision or statistics core** (uncertainty, incentives, evaluation, aggregation). That is what makes it research rather than a build.
- Two are pricing-adjacent (8 and 9) because that is real experience you have; the rest are not.

01 When should an AI agent stop and ask a human?

Decision under uncertainty · human oversight

THE QUESTION

An agent working through a task can act on its own or escalate to a person. Asking costs expert time; acting wrongly costs more. What is the best escalation rule, and how does it change when the agent's confidence is miscalibrated?

WHY IT'S INTERESTING

Paper2Agent keeps a human in the loop, but only informally. This turns “human in the loop” into a stopping or threshold problem with a clear value-of-information trade-off. A biased confidence signal changes the optimal rule, which is a nice, testable prediction.

A FIRST STEP

Take your complaint-triage prototype. Get the LLM's probability for each label and measure calibration on a labelled sample. Then compare three rules on accuracy against the number of escalations:

- a fixed confidence threshold;
- a cost-based threshold;
- “ask when the top two labels are close”.

YOUR EDGE

You have built human-in-the-loop validation (the SM-104 knowledge assistant) and a triage prototype.

Bayesian decision theory | optimal stopping | calibration

02 Testing tools whose answers are random

Statistics · a direct Paper2Agent follow-up

THE QUESTION

Paper2Agent accepts a tool if its numbers are within 3% of the tutorial's. For MCMC, bootstrap or simulation tools, two correct runs can differ by more than that. What test should decide that an extracted tool is “the same” as the original?

WHY IT'S INTERESTING

It is a clean statistics problem inside a current AI system. The core is equivalence testing for distributions: you trade off rejecting good tools against accepting subtly wrong ones, and you can measure both.

A FIRST STEP

Run Paper2Agent on emcee (MCMC) or a bootstrap package, then inject small silent bugs, such as a wrong prior or a shortened burn-in. Compare the 3% rule with a distribution-level equivalence test across many random seeds. For each test, count the bugs it catches and the correct tools it rejects.

YOUR EDGE

Statistics training plus hands-on Claude Code and MCP.

equivalence testing (TOST) | two-sample tests | simulation

03 How many questions make a trustworthy agent benchmark?

Evaluation · measurement

THE QUESTION

Agent papers report accuracy on 15–300 questions with a few runs each. How much of the gap between two agents comes from question sampling, run-to-run randomness and grader disagreement? And how many questions and runs do you need to rank two agents reliably?

WHY IT'S INTERESTING

Evaluation is the bottleneck of agent research, and uncertainty is rarely reported properly. A variance decomposition plus a power analysis would be a small, useful and publishable contribution.

A FIRST STEP

Use Paper2Agent's public benchmark questions from its GitHub repository. Run two agents (with and without the MCP) 5–10 times each. Fit a mixed-effects model of correct/incorrect with question and run effects, and report the sample sizes needed.

YOUR EDGE

Regression background, and cheap to run.

mixed-effects models | bootstrap | power analysis | Cohen's kappa

04 LLM crowds: diversity, correlation and herding

Information aggregation · forecasting

THE QUESTION

Ask five AI agents and you don't get five independent opinions, because they share training data and blind spots. If agents see each other's answers first, they may also herd. How should their answers be combined? And does mixing different model families beat sampling one model many times?

WHY IT'S INTERESTING

These are classic questions about crowds and aggregating information, applied to a new kind of crowd whose errors are correlated by construction. The answer matters for any multi-agent system, including Paper2Agent's co-scientists.

A FIRST STEP

Take about 100 forecasting questions that have already resolved (public forecasting platforms publish them). Collect probability forecasts from several models, first independently and then in sequence with each model seeing the earlier answers. Measure error correlation, herding, and the Brier score of different aggregation rules.

YOUR EDGE

You use several model families daily (Claude, Gemini Gems, NotebookLM) and have forecasting coursework.

Brier score | aggregation rules | correlated errors

05 Competing for an AI agent's attention

Strategic information · manipulation · MCP markets

THE QUESTION

When agents choose tools (MCP servers) or products on a user's behalf, providers have an incentive to write descriptions that steer the agent. How easily are agents swayed? Does it hurt the user? What disclosure or ranking rules would reduce it?

WHY IT'S INTERESTING

This is strategic communication in a new market where the reader is an algorithm. It links incentives, information design and AI reliability, and it will matter more as marketplaces of MCP servers grow. **OUTSIDE** Security researchers have already shown that malicious tool descriptions can steer agents ("tool poisoning").

A FIRST STEP

Build a small MCP marketplace: 5–10 tools that do the same job, with honest or "persuasive" descriptions, and some of lower quality. Measure how often the agent picks each one and the resulting task quality. Then repeat with product listings and an LLM shopping agent.

YOUR EDGE

You build with MCP. Your lifecycle-marketing work means you know how copy is optimised for engagement. You have also used multinomial logit choice models (the Netflix case study).

controlled experiments | choice models (MNL) | simple game theory

06 Does "validated" make people stop checking?

Human–AI reliance · experiment

THE QUESTION

Paper2Agent labels its tools as validated. Do such labels make users check less and catch fewer planted errors? Does showing uncertainty (intervals, or "I might be wrong") bring the checking back?

WHY IT'S INTERESTING

Over-reliance is the practical risk of agents that *look* reliable, and it can be measured with a simple randomised design. The results speak directly to how agents should present evidence.

A FIRST STEP

Run an online experiment with classmates. Show 12 agent answers to data questions, 3 of which contain planted errors. Randomise the presentation: no label, a "validated tools" label, or the label plus an uncertainty statement. Measure error detection and time spent. You will need ethics approval, so check NUS's IRB process.

YOUR EDGE

Conversational-UX work at Yellow.ai, and access to a student pool.

randomised experiment | logistic regression | information design

07 Complaint triage as a capacity problem

Operations · prediction meets prioritisation

THE QUESTION

A team can investigate only *k* complaints a day, and an LLM gives each complaint a probability of being severe or a regulatory risk. Which should be handled first? When is it worth waiting for more information, such as a follow-up complaint?

WHY IT'S INTERESTING

It combines prediction with a queueing and prioritisation decision. Ranking by "probability severe" alone ignores costs, deadlines and how uncertain each prediction is.

A FIRST STEP

Use the US CFPB Consumer Complaint Database, which is public and includes narratives and timely-response flags. Build an LLM classifier and calibrate it. Then simulate three policies under capacity limits: rank by probability, rank by expected cost, and an index policy.

YOUR EDGE

Your airline complaints prototype (3,000+ complaints, severity, regulatory flags) is already a pilot.

calibrated classification | queueing and priority policies | simulation

08 When algorithmic revenue managers compete

Pricing-adjacent · algorithms in markets

THE QUESTION

Airlines increasingly price with learning systems, like the one you tuned. Suppose the pricers on a route learn from each other's visible fares. Do fares drift up or down, or oscillate? What changes if one side is an LLM-based agent instead of a classic bid-price rule?

WHY IT'S INTERESTING

Whether algorithms coordinate, and how fragile that coordination is, is an open policy question. A realistic revenue-management simulation, with capacity, stochastic demand and booking curves, is rarer than the usual toy models.

A FIRST STEP

Build a two-airline route simulator with stochastic demand and fixed capacity. Pit bid-price rules against learning pricers and an LLM agent. Vary fare visibility and demand noise, then measure fares, load factors, and whether any coordination survives demand shocks.

YOUR EDGE

Hands-on revenue management at Air India Express: bid-price tuning and competitive response.

simulation | bandits and reinforcement learning | game theory

09 When should a human override the AI?

Pricing-adjacent · human judgement with private information

THE QUESTION

Analysts often override AI recommendations; you tuned bid prices by hand. When do overrides add value, and when do they just add noise? Can you learn which analyst–situation pairs to trust?

WHY IT'S INTERESTING

The human knows things the model doesn't, such as events or competitor moves, but also has biases. That makes this a clean signal-plus-bias problem, with implications for how human–AI teams should be designed.

A FIRST STEP

Write a small model: the algorithm sees data, and the human sees a private signal plus a bias. Derive when overriding improves the expected outcome. Then test it with a short exercise: forecasting or allocation tasks with an AI suggestion and a private cue, varying the cue's quality.

YOUR EDGE

You have been the human in this loop.

Bayesian signal model | experiment | regression

10 Paper agents for operations research: certificates as tests

Optimisation · verification

THE QUESTION

Optimisation has something biology lacks: answers you can verify, through feasibility checks and an optimality gap. Can these certificates replace tutorial matching as the test for OR tools? And do agents with validated OR tools formulate and solve problems more reliably?

WHY IT'S INTERESTING

It extends Paper2Agent into operations research and brings in a stronger notion of correctness than “matches the tutorial”. It also asks whether agents fail at formulating a problem or at solving it.

A FIRST STEP

Agentify one or two OR codebases, such as the OR-Tools examples or a Pyomo model library. Write 20 business problems (scheduling, allocation, inventory). Compare Claude with the raw repository against Claude with validated MCP tools, on feasibility, optimality gap and runtime.

YOUR EDGE

Optimisation case work (Solver), plus Claude Code and MCP.

linear and integer programming | duality | benchmark design

E1 · What today is

- He has said there are no RA positions now. He still made time and sent you a paper: he wants to see how you read and think. Treat it as an exploratory conversation, not an interview for a post.
- **Your goals, in order:** (1) show you understood the system and can think critically about it; (2) find out why he chose this paper and what interests him about it; (3) get his honest advice on PhD vs industry routes; (4) leave with **one concrete next step** you can deliver.
- **What tends to work with a quantitative professor:** precise understanding; separating what was shown from what was claimed; one or two sharp statistical questions; honesty about your goals; following through on what you promise.
- **Avoid:** reciting numbers; attacking the paper; big startup talk; pretending you want an academic career.

E2 · Your 60-second introduction (draft – put it in your own words)

"Thank you for making the time, Professor. Briefly: my undergraduate degree was in computer science, maths and statistics. I then worked as a revenue analyst at Air India Express, where a lot of my work was tuning and monitoring an AI-driven revenue-management system, and before that I interned at two AI companies, Yellow.ai and SuperAGI. I'm now in the MSc in Management here, focusing on business analytics. Outside class I build with Claude Code and MCP, so this paper connected directly to things I've been doing. I'm seriously considering a PhD, but I'm still working out whether it's the right route for me, and I'd really value your advice."

E3 · Your 2-minute summary of the paper (say it out loud)

"The problem is that a paper's method is hard to use: you have to find the code, install it and work out the inputs. Paper2Agent automates that. A group of Claude Code agents finds the repository, builds an environment, finds the tutorials and actually runs them. Then they turn the tutorial steps into general functions and test each one against what the original tutorial produced — numbers within 3%, figures by image similarity. Functions that keep failing are dropped. What survives is packaged as an MCP server with three parts: tools that run the method, resources like the paper and data, and prompts that encode workflows. Any agent can connect to it.

They show it on AlphaGenome, Scanpy and TISSUE. On AlphaGenome, the agent got essentially all the exact queries right, against about 80% for Claude working from the raw repository, and it was two to four times faster. At scale, 74 of 100 biology papers could be agentified, and it also worked on statistics and machine-learning packages. They also connect three paper agents to prioritise a psoriasis gene, with a human choosing the validation strategy.

What I found most interesting is their own caveat: agreement with a reference measures faithful execution, not whether the scientific conclusion is right."

E4 · A simple flow for the conversation (about 30–45 min)

1. **Thanks + intro** (1 min).
2. **The paper:** 2-minute summary → "There were two things I wanted your view on …" (pick two from C3 or C5-E).
3. **Listen:** "What made you pick this paper for me?" Let him talk; follow his thread.
4. **Your situation, honestly:** considering a PhD; leaning towards industry or a startup; want work that combines AI with decision-making; want to stay and work in Singapore.
5. **Routes:** the career questions (E6).

6. **Ideas:** if the conversation opens up, float one or two ideas from Part D — as questions.
7. **Offer:** a small piece of work (E7). “Could I send you something in three or four weeks?”
8. **Close:** thank him; ask whether there is anyone else he'd suggest you talk to.

E5 · Programmes – fact sheet

Checked 21 Sep 2026 from the sources at the end. Rules change — confirm on official pages before you rely on a detail.

PROGRAMME	KEY RULES	MONEY	NOTES FOR YOU
NUS Business School PhD (Analytics & Operations)	Full-time. Applications 1 May – 15 Dec 2026 for the August 2027 intake. GRE/GMAT optional.	NUS Research Scholarship ≈ S$2,700/month before the qualifying exam, up to ~S$500 more after (NUS-wide figure from secondary sources — confirm).	Designed mainly to train researchers. Ask where graduates go outside academia, and whether co-supervision with SoC happens.
AISG PhD Fellowship (AI Singapore)	You need admission to (or be ≤ 2 years into) an **AI-related PhD** at NUS SoC, NUS CDE (Computer Engineering), NTU CCDS, SMU SCIS or SUTD ISTD; **other faculties are assessed case by case**. The university nominates you. Supervisor(s) must have published actively in the last 3 years at top AI venues (e.g. AAAI, ACL, CVPR, EMNLP, ICCV, ICLR, ICML, IJCAI, KDD, NeurIPS). Jan 2027 intake: nominations close 22 Oct 2026; NUS SoC's August-intake deadline is 15 Dec.	S$6,300/month for international students + full tuition + up to S$8,000 travel per top-tier paper; up to 4 years.	You must publish ≥ 4 first-author papers at top venues; no other paid work. International fellows have a **2-year service obligation** (work in Singapore after graduating) — which fits your goal of staying. Singapore citizens/PRs and ASEAN nationals get priority.
SAP Labs Singapore — PhD Research Associate (Industry PhD Program), AI	Full-time SAP employee + NUS PhD; SAP mentor + NUS supervisor. Requires acceptance into an NUS PhD; degree in CS, maths, statistics or similar; ML frameworks, Python, LLM experience. SAP says it is prioritising candidates with full working rights in Singapore. Posted 9 Sep 2026.	Salaried (amount not stated).	Topics: LLM agents, robust LLM-based evaluation, multimodal RAG, document AI, LLMs for tables, synthetic data for anomaly detection — close to this paper. Part of an EDB-supported plan to train 9 researchers by 2030.
EDB Industrial Postgraduate Programme (general)	Company-employed PhD, ≥ 50% of time on a company project. Singapore citizens or PRs only (per SUTD's IPP page).	Salary, part reimbursed by EDB (cap S$4,000/month per SUTD).	Probably why SAP prioritises work rights. As an international student this route is uphill — worth asking about exceptions.
EP reality check	An Employment Pass needs ≥ 40 COMPASS points and ≥ S$5,600/month (2026, most sectors; higher with age). Qualifications give 20 points for a degree from a top-tier institution.	—	Your NUS MSc already earns the 20 qualification points; a PhD doesn't add more. What a PhD changes is which jobs you can get (research roles), not EP eligibility.

> **An honest reading** MY READING If your aim is industry and a good income in Singapore, the PhD options that pay close to a salary (AISG, an industry PhD) both lean towards computing departments and publishing at ML conferences. A low-risk way to test research fit before committing 4–5 years is one small project with a professor's feedback over the next 1–3 months.

E6 · Career questions to ask

1. “For someone who wants to end up in industry or a startup rather than academia, when does a PhD in Analytics & Operations make sense — and when would you advise against it?”
2. “What do your department's PhD graduates do when they don't take faculty jobs?”
3. “The AISG fellowship expects an AI-related PhD, usually in a computing school, and a supervisor with recent top-tier AI papers. Is co-supervision between the Business School and SoC common here?”
4. “SAP Labs has just posted an NUS industry PhD on LLM agents and evaluation. It prioritises people with Singapore work rights. Do you know whether such industry PhDs can be based in the Business School, or whether there are routes open to international students?”
5. “The deadline for August 2027 is 15 December. What would make an application from someone like me stronger in the next three months?”
6. “Would you be open to me doing a small independent project related to this paper and sharing it with you?”

To ask others rather than him: what PhD graduates earn versus MSc graduates in the roles you want; whether any part-time or employed PhD route exists; how long the programme usually takes in practice.

E7 · A next step you could offer (ideas, not a plan)

1. **Any one idea from Part D**, cut down to a 3–4 week pilot (ideas 2, 3 and 10 build directly on this paper).
2. **Agentify 3–5 statistics / OR / analytics codebases** with the open-source Paper2Agent (it now installs as a Claude Code skill; the paper reports about US$13–14 of API cost to build each server). Record whether it builds, how many tools pass, and accuracy on genuinely new tasks, with confidence intervals.
3. **Stress-test validation for one random method** (e.g. an MCMC or bootstrap package): show how the fixed 3% test behaves across random seeds, and try a distribution-level test instead.
4. **Write a two-page critique of the paper's evaluation design** (C3) with a proposal for a better benchmark.

Keep it small enough to finish in 3–4 weeks alongside your MSc. Offer it; don't over-promise.

E8 · Follow-up email (send within 24 hours)

```
Subject: Thank you — Paper2Agent discussion

Dear Professor Keppo,

Thank you for your time today and for sending me the Paper2Agent paper.
I especially appreciated your point about [one specific thing he said].

As we discussed, I will [next step, e.g. agentify three statistics packages with
Paper2Agent and test how its validation handles random outputs], and I'll share a
short write-up by [date].

[If he suggested someone or something to read: I'll also look into / reach out to …]

Best regards,
Aditya
```

E9 · The one-page cheat sheet – write this by hand

1. **Mental model:** the two-phase flow in Part F (build once / use every query).
2. **MCP:** tools = act (model chooses) · resources = read (context) · prompts = recipe (user chooses).
3. **Build:** locate → environment → find tutorials → RUN → extract → test (files · 3% · hash < 20 · ≤ 6 tries · drop failures) → assemble → host.
4. **Numbers:** AlphaGenome 22 tools, ~45 min, ~US$14 · 98.7 / 100% vs 82.7 / 78.7% (Claude + Repo) vs 37.3 / 56.0% (Biomni) · open-ended 82.7 vs 56.7 vs 72.2 · 1.9–3.8× faster (medians) · 74/100 papers, 593/599 tools · 91.2% vs 80.3% · non-biology 98.1% · resources 89% · out-of-scope 100%.
5. **Two critical points** you will raise (from C3).
6. **Two research ideas** you might float (Part D).
7. **Two questions for him** (C5-E) + your proposed next step (E7).
8. **Three career questions** (E6).

F · The whole system, corrected

```
BUILD — once per paper   (Paper2MCP: Claude Code sub-agents)
Research paper + code + tutorials + data
        ↓
find the repo → set up an isolated environment
        ↓
find tutorials → RUN them → reference outputs (files, numbers, figures)
        ↓
extract single-purpose functions (hard-coded values → parameters)
        ↓
test against the reference outputs   (≤ 6 repair attempts; failures dropped)
        ↓
MCP server = tested tools + resources + prompts   (resources only, if no tools)
        ↓
host remotely (e.g. Hugging Face)

USE — every query   (paper agent = any MCP agent + this server)
natural-language request  (optionally a named prompt)
        ↓
agent plans
        ↓
calls tools / reads resources → observes results → refines the plan  (repeat)
        ↓
answer + files/figures   (or "I don't know" if out of scope)
        ↓
human judges, chooses directions, validates
```

What changed from your version: build and use are split (the build runs once); "understand + extract" really means *run the tutorials first*, then extract; resources and prompts are packaged too; there is a refusal path; and a human sits at the end.

The paper in one paragraph

Paper2Agent treats a research paper as software that should be usable, not just readable. A team of AI coding agents finds a paper's code, gets it running, runs its tutorials, rewrites the useful steps as general functions, and keeps only the functions that reproduce the tutorials' outputs. Those tested tools, together with the paper's text and data (resources) and step-by-step workflows (prompts), are packaged as an MCP server that any AI agent can connect to. A user can then ask, in plain language, to apply the method; the agent plans, calls the tools, checks the results and reports. On the authors' benchmarks this is more accurate and faster than giving an AI agent the raw repository; it works on about three-quarters of real biology papers and on statistics and machine-learning packages; and several paper agents can be combined to build evidence for a hypothesis. What it demonstrates is faithful execution of published methods. Whether the scientific conclusions are right still needs human judgement and, often, new experiments.

Sources

- The paper: Miao et al., *Nature* (2026), doi:10.1038/s41586-026-11044-y · code: https://github.com/jmiao24/Paper2Agent
- AISG PhD Fellowship page: https://aisingapore.org/research/phd-fellowship-programme/
- AISG application guideline (Jan 2026 intake, version 18 Sep 2025): https://aisingapore.org/wp-content/uploads/2025/08/2.1-AISG-Research-PhD-Fellowship-Application-Guideline-Jan-2026-Intake_Applicant_v1_clean.pdf
- SAP job posting (req. 439621): https://jobs.sap.com/job/Singapore-PhD-Research-Associate-(Industry-PhD-Program)-Artificial-Intelligence,-SAP-Labs-Singapore-117440/1264048901/
- EDB release on SAP Labs + NUS (Mar 2025): https://www.edb.gov.sg/en/about-edb/media-releases-publications/sap-labs-singapore-boosts-ai-innovation-with-local-talent.html
- SUTD EDB-IPP page: https://www.sutd.edu.sg/admissions/graduate/edb-ipp/
- NUS Business School PhD admissions: https://bschool.nus.edu.sg/phd/admissions/ · NUS Graduate School scholarships: https://nusgs.nus.edu.sg/scholarships/
- COMPASS 2026 summary (Fragomen): https://www.fragomen.com/insights/singapore-compass-lists-of-educational-qualifications-and-awarding-institutions-updated-for-2026.html · check MOM: https://www.mom.gov.sg/passes-and-permits/employment-pass