How to Make Your Science Easier with AI Support

Tianyuan Liu

Date: October 3rd, 2025
Time: 10:00–12:00
Location: I2SysBio Seminar Room

by our PhD student: **Tianyuan Liu**

October 3rd, 2025

Genomics of Gene Expression Lab

Outline

1. Introduction to AI-Assisted Development
2. Context Engineering
3. Responsible AI & Production Best Practices
4. Conclusion

Quick Poll: Your AI Experience

Genomics of Gene Expression Lab

Let's see where everyone is!

1. **Who uses LLMs?**
   (ChatGPT, Gemini, Claude for conversations)
2. **Who uses coding agents?**
   (GitHub Copilot, Cursor, Claude Code, Gemini CLI)
3. **Who uses agents for everything?**
   (Slides, CVs, research papers, full workflows)

Raise your hand for each one!

Quick Start: Quick Install

```
# Using npx (no installation required)
npx https://github.com/google-gemini/gemini-cli

# Install globally with npm
npm install -g @google/gemini-cli

# Install globally with Homebrew (macOS/Linux)
brew install gemini-cli

# Requirements: Node.js v20+, macOS/Linux/Windows
```

Workshop Structure

What We'll Cover

1. **Introduction to AI-Assisted Development**
   ReAct, MCP, Tools

2. **Context Engineering**
   Guiding AI effectively

3. **Responsible AI & Production Best Practices**
   Best practices & quality

4. **Conclusion**
   Future directions

Live Demos

- **Demo 1: Research slides**
  LaTeX generation from papers

- **Demo 2: CV generation**
  Context engineering in action

- **Demo 3: React website**
  Unit testing & CI/CD

Hands-on + Theory for practical AI-assisted research

Let's Get Started: Clone the Repository

Clone the Workshop Materials

```
git clone
   https://github.com/TianYuan-Liu/ai-agent-workshop
```

This repository contains:

- All demo examples
- LaTeX templates
- Configuration files
- README with setup instructions

Please clone this now so you can follow along with the demos!

LLM vs. AI Agent

Large Language Model

- Cannot use tools

Example: ChatGPT (basic mode)

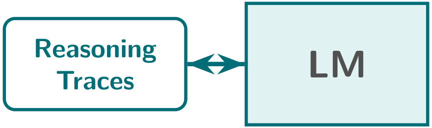
AI Agent

- Tools: Execute actions
- Memory: Context retention
- Planning: Multi-step tasks

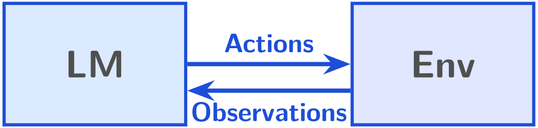
Example: Cursor, Gemini CLI

Agent = LLM + Tools + Memory + Planning[1]

Reason Only and Act Only

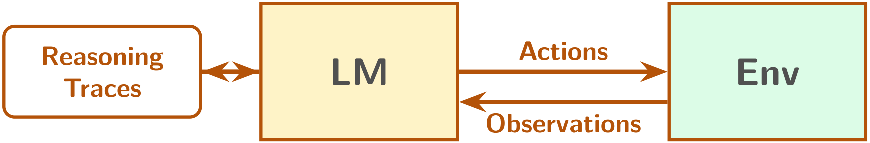


Reason Only



Act Only

ReAct (Reason + Act)



(Reason + Act)[2]

Why Do We Need MCP?

The Missing Link

ReAct combines reasoning and action, but lacks a standardized, reliable interface for taking actions.

Without MCP

- **Custom code for each AI agent**
- **Tools stay invisible to AI**

With MCP

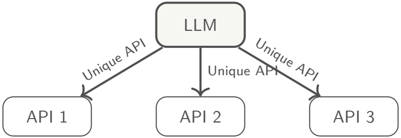
- **One protocol for all agents**
- **AI finds and uses tools itself**

MCP = The bridge enabling ReAct to work in practice[3]

API vs MCP: Before and After

Genomics of Gene Expression Lab

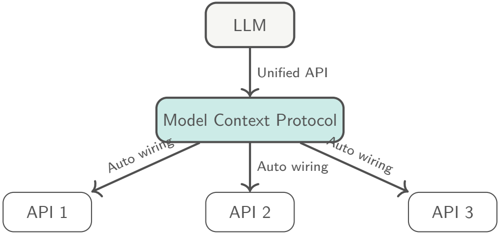
Before MCP



Traditional API

You write code

After MCP



MCP

AI uses tools

MCP = APIs that AIs can find and use by themselves

API vs. MCP: Concrete Example

Task: “What’s the weather in Valencia?”

Using Traditional API

You must code:

```python
import requests

def get_weather(city):
    url = "https://api.weather.com"
    params = {"city": city}
    response = requests.get(url, params)
    data = response.json()
    return data["temp"]

# Call it manually
temp = get_weather("Valencia")
print(f"Temperature:␣{temp}C")
```

Problem: You write & maintain all integration code

Using MCP

MCP Server provides:

```json
{
  "name": "get_weather",
  "description": "Get␣current␣weather",
  "parameters": {
    "location": {
      "type": "string",
      "description": "City␣name"
    }
  }
}
```

AI automatically:

- ✓ Reads the schema
- ✓ Knows when to call it
- ✓ Passes correct parameters
- ✓ Interprets the response

MCP Call Flow: How It Works

Genomics of Gene Expression Lab

Understanding Client-Server Architecture

The Players

Client: The AI agent
Server: Your MCP tool provider

Client asks, server does the work

The Call Flow

1. Client connects → `list_tools` (gets tool schemas)
2. AI chooses tool + arguments
3. Client → `call_tool(name, args)`
4. Server runs code → returns data (hits APIs, DB, shell, etc.)

Key: AI never sees your implementation, just the interface

Why MCP Is More Secure

Security Through Controlled Access

The Risk

Model has no direct power
Only calls tools you expose

But if model sends personal data:

- Client forwards it
- Server processes it
- Data could leak out

Warning: Privacy risk once data leaves

MCP Protection

1. **Schema constraints**
   Strict field validation
2. **Scoped servers**
   Only run what you trust
3. **Local by default**
   Data stays on your machine
4. **Auditing/logging**
   Track all tool calls

Bottom line: You control what tools exist and where data flows

How LLMs Learn Tool Use[4]

Genomics of Gene Expression Lab

Pretraining

- Massive web + code
- Predict next tokens
- Learn world priors

Instruction Tuning

- Prompt–response demos
- Tool schemas inline
- Structured formats

Alignment / RLHF

- Rewarded feedback
- Safety objectives
- Reliable tool calls

ReAct in Action: RNA-Seq Workflow

Genomics of Gene Expression Lab

Task: "Deliver QC-ready RNA-seq counts for today's batch."

1. **Thought**: "Need QC thresholds before planning."
2. **Action**: `read_file("GEMINI.md")`
3. **Observation**: "Standards: reads >10M, FRiP >0.2."
4. **Thought**: "Plan pipeline using those gates."
5. **Action**: `launch_rnaseq_qc(batch="2024-10-01", min_reads=1e7)`

Result: counts.tsv + qc_report.html ready for review.

Agent Stack for Scientific Workflows

Genomics of Gene Expression Lab

Agent Ingredients

- Foundation model reasoning
- ReAct planning loop
- MCP tool catalog

Human Inputs

- GEMINI.md + project context
- Quality gates & constraints
- Review & acceptance criteria

Bridge: This stack powers the upcoming demos for slides, CVs, and production code.

Coding Agents

Genomics of Gene Expression Lab

Definition

ReAct loops for planning, writing, testing & refining code autonomously[2].

Where They Shine

- **Automate repetitive QC or plotting**
- **Scaffold pipelines & tests**
- **Keep research notes in sync**

What You Still Own

- **Set objectives & constraints**
- **Provide project context**
- **Review & sign off outputs**

Next: Watch this collaboration build research slides.

Demo 1

Coding Agent for Research

Generating LaTeX Slides from Research Papers

Live Demo: Coding Agent for Research

Goal Turn the epigenome review into polished slides with a ReAct loop.

Setup

- Workspace: `examples/demo1/`
- Read `prompt.txt`
- Use Conesa theme

Loop

1. Think: ground in prompt + paper
2. Act: write slides, manage assets
3. Observe: compile, adjust

Checklist

Compile cleanly; figure rule respected; log key decisions.

Coding Agent Example: RNA-Seq

Scenario

Raw FASTQs $\rightarrow$ expression report by morning

ReAct Loop

1. **Think: Review QC goals**
2. **Act: FastQC + STAR**
3. **Observe: Flag weak samples**
4. **Deliver: Counts ready**

Responsible Use

- **Review flagged reads**
- **Track tool versions**
- **Log decisions**

What Is Context Engineering?

Genomics of Gene Expression Lab

Key Insight[5]

"Most agent failures are context failures, not model failures." — Tobi Lütke, Shopify

Core Components

Instructions
Project Context
Domain Knowledge
Examples

Context = Better AI

Specificity beats vagueness
Examples beat explanations
Constraints beat preferences

Guiding AI for Multi-Omics Analysis

Genomics of Gene Expression Lab

Simple Strategy

Tell AI your **data type** + **quality rules** + **ask for a plan**

RNA-seq Example

```
QC RNA-seq FASTQs, align to
GRCh38, generate counts.

Quality: $>$10M reads

Output: counts.tsv
```

ATAC-seq Example

```
Process ATAC reads, call peaks
with MACS2.

Quality: FRiP $>$ 0.2

Output: filtered_peaks.bed
```

Key: Specific data type + QC thresholds + reference to GEMINI.md

Anatomy of a Research GEMINI.md

```
# Project: Multi-omics Data Analysis

## Data (CRITICAL)
- IRB: #2024-456 | Data: RNA-seq, ATAC-seq

## Tech
- STAR (GRCh38), MACS2, RGmatch, Python 3.11

## Domain
- RNA-seq QC: $>$10M reads | ATAC-seq QC: FRiP $>$ 0.2

## Stats
- Correlation networks for omics integration | FDR: BH | Seed:
  42
```

Key Sections: Privacy • Tech • Domain • Stats

Context Engineering Best Practices

DON'T	DO
"Write good code"	**Functions $<$ 50 lines**
12 pages of docs	**Core concepts only**
Outdated specs	**Living document**

Remember: Examples $>$ Explanations

Demo 2

Context Engineering in Action

Building a LaTeX CV from Web Forms

The Workflow

Web form → Structured prompt → AI creates CV

Your Task

1. **Open** `examples/demo2/index.html`
2. **Fill in the form**
3. **See what context AI needs**

Understanding context = Better AI output

Key Insight

The form shows you exactly what information AI agents need:

- **Personal details**
- **Education & experience**
- **Skills & projects**
- **Formatting preferences**

What Is Vibe Coding?

Genomics of Gene Expression Lab

Definition[6]

"A new coding style where you embrace exponentials and forget the code even exists."
— Andrej Karpathy

Vibe Coding

Prompt → Generate → Run

No Review!

Responsible AI

Prompt → Generate → REVIEW

Test → Deploy

The Research Integrity Framework

Genomics of Gene Expression Lab

Code Type	Examples	AI Usage
Exploration	Visualizations, Conversions	✓Vibe OK
Production	Pipelines, Tools	AI + Review
Critical	Statistics, Publications	× Traditional

Real Risks for Researchers

Genomics of Gene Expression Lab

Security[7]

- 40% of AI code has flaws
- Prompt injection
- Hardcoded credentials

Reproducibility

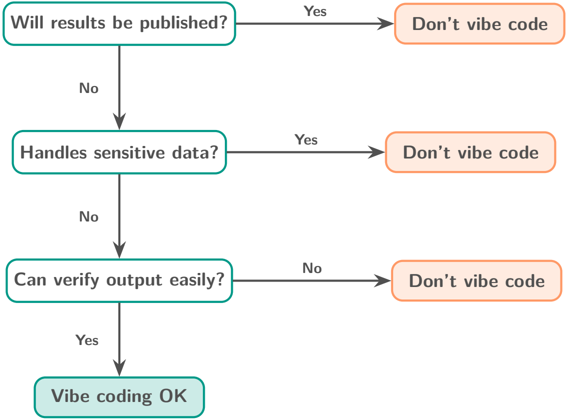
- Non-deterministic
- Can't verify
- Journals need disclosure

Surprising Finding[8]

- Experienced devs: 19% slower with AI on complex tasks

Best Practices Decision Tree

Genomics of Gene Expression Lab



Keeping Vibe Coding in Check

Genomics of Gene Expression Lab

1. **Ask for a step-by-step plan before work starts**

   Example: "Make a step-by-step plan for RNA-seq QC analysis"

   Agent outputs: 1. FastQC on FASTQs 2. MultiQC reports 3. Flag $<$10M reads 4. Summary stats

2. **Review and approve the plan before execution**

   ✓ Review agent's proposed steps and approve or modify

3. **Have the agent make a to-do list to track progress**

   Track completion: ✓ Done ▷ In progress Pending

Version Control: Essential for Agent Coding

Why GitHub Matters

AI agents iterate fast, mistakes happen — **version control is critical** for tracking, rollback, and collaboration.

Key Benefits

- **Track every change**
- **Easy rollback**
- **Safe collaboration**

Workflow

1. **Branch** $\rightarrow$ **Generate**
2. **Review** $\rightarrow$ **Commit**
3. **Pull request** $\rightarrow$ **Merge**

CI/CD & Unit Testing: Quality Automation

What is CI/CD?

Auto-test and deploy AI code on every commit.

Unit Tests

- Fast feedback
- Catch regressions

pytest, Jest, JUnit

CI/CD Flow

1. Push code
2. Auto-test & lint
3. Block if failing
4. Deploy if passing

GitHub Actions

Demo 3

Production-Ready AI Code

React Website with Unit Testing & CI/CD

Demo 3: Unit Testing & CI/CD in Practice

Genomics of Gene Expression Lab

Building a React CV Website with Automated Testing

What's Tested

- Component rendering
- Data validation
- User interactions
- Accessibility (a11y)

`npm test` — runs all tests

CI/CD Workflow

1. Push to GitHub
2. Tests run automatically
3. Lint & format check
4. Build production site
5. Deploy to GitHub Pages

Location: `examples/demo3/` — See `README.md` for full guide

Next Step: Fully Autonomous Agents

Why This Matters

The next evolution of AI agents is full computer control so they can execute end-to-end workflows without human clicks.

- **Example:** Agent-TARS plans and books trips autonomously.
- **Capability:** Interfaces with web apps, files, and APIs to manage travel logistics while you supervise outcomes.
- **See it:** `https://agent-tars.com/`

Takeaway: Prepare workflows and safeguards so agents can operate safely when they get full desktop control.

The Last Agent?

Embodied Intelligence brings AI into the physical world.

- **Unitree G1:** Anti-gravity mode trained via reinforcement learning. Watch: https://www.youtube.com/watch?v=bPSLMX_V38E
- **Figure AI:** Humanoid robot that can do your dishes autonomously. Watch: https://www.youtube.com/watch?v=8gfuUzDn4Q8

References I

Genomics of Gene Expression Lab

[1] Zhiheng Xi et al. "The Rise and Potential of Large Language Model Based Agents: A Survey". In: *arXiv preprint arXiv:2309.07864* (2023). Comprehensive survey of LLM-based autonomous agents. URL: https://arxiv.org/abs/2309.07864.

[2] Shunyu Yao et al. "ReAct: Synergizing Reasoning and Acting in Language Models". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. Introduces the ReAct framework combining reasoning traces and task-specific actions. 2023. URL: https://arxiv.org/abs/2210.03629.

[3] Anthropic. *Introducing the Model Context Protocol*. Anthropic Blog. Technical documentation for Model Context Protocol. Nov. 2024. URL: https://www.anthropic.com/news/model-context-protocol.

[4] Yujia Qin et al. "Tool Learning with Foundation Models". In: *arXiv preprint arXiv:2304.08354* (2023). Survey on how foundation models learn to use external tools. URL: https://arxiv.org/abs/2304.08354.

References II

[5] Tobi Lütke. *Context Engineering: The Key to Successful AI Agents.* Shopify Engineering Blog. Most agent failures are context failures, not model failures. 2024.

[6] Andrej Karpathy. *Vibe Coding: A New Paradigm in AI-Assisted Development.* Personal Blog. Accessed: 2025-10-01. Feb. 2025. URL: `https://karpathy.ai/`.

[7] Sarah Johnson, Wei Chen, and Luis Martinez. "Security Vulnerabilities in AI-Generated Code: A Large-Scale Analysis". In: *Journal of Cybersecurity Research* 12.3 (2025). 40% of AI-generated code contains security flaws, pp. 234–251. DOI: `10.1016/j.jcr.2025.03.015`.

[8] METR Research Team. *Evaluating Developer Productivity with AI Tools: A Comprehensive Study.* Technical Report. Found 19% decrease in productivity for complex tasks. Model Evaluation and Threat Research, 2025.