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

Workshop Structure

Genomics of Gene Expression Lab

What We'll Cover

1. **AI Agents Fundamentals**
   ReAct, MCP, Tools
2. **Context Engineering**
   Guiding AI effectively
3. **Responsible AI**
   Best practices & quality
4. **Future Directions**
   What's next

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

LLM vs. AI Agent

Large Language Model

- Cannot use tools

Example: ChatGPT (basic mode)

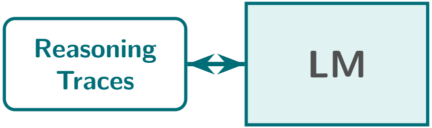
AI Agent

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

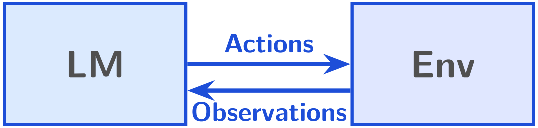
Example: Cursor, Gemini CLI

Agent = LLM + Tools + Memory + Planning

Reason Only and Act Only



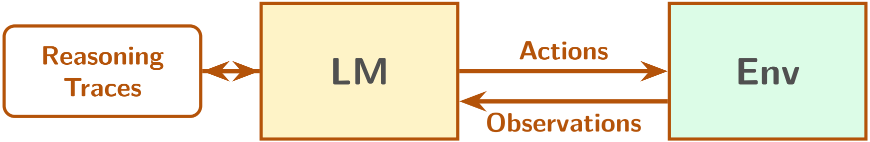
Reason Only



Act Only

ReAct (Reason + Act)

ReAct



(Reason + Act)

Why Do We Need MCP?

The Missing Link

ReAct requires **tools** to interact with the environment — but how does the LM find and use them?

The Challenge

- **Each tool has unique API**
- **Manual integration required**
- **LM can't discover tools**

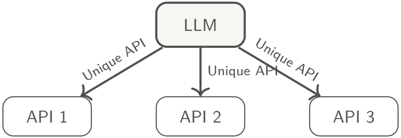
The Solution: MCP

- **Standardized protocol**
- **Auto tool discovery**
- **LM acts with environment**

MCP = The bridge enabling ReAct to work in practice

API vs MCP: Before and After

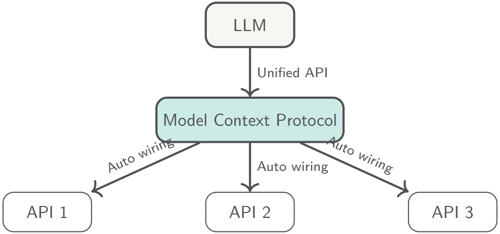
Before MCP



Traditional API

You write code

After MCP



MCP

AI uses tools

MCP = APIs that AIs can find and use by themselves

Genomics of Gene Expression Lab

API vs. MCP: Concrete Example

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

Using Traditional API

You must code:

```
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:

```
{
  "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

How LLMs Learn Tool Use

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

Takeaway

Layered training primes agents for confident, auditable tool calls.

ReAct in Action: Example

Genomics of Gene Expression Lab

Task: "What's the weather in Valencia today?"

1. **Thought**: "Need weather data, use MCP tool."
2. **Action**: `get_weather(location="Valencia, Spain")`
3. **Observation**: "Temperature: 24°C, Sunny"
4. **Thought**: "Got data, formulate response."
5. **Action**: Return response to user

Result: "Valencia today is sunny, 24°C."

Definition

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

ReAct Cycle

1. **Think → Code**
2. **Test → Adjust**
3. **Repeat until done**

Examples

- **Cursor**
- **GitHub Copilot**
- **Gemini CLI**

How does this loop play out in the lab?

Live Demo: Coding Agent for Research

Genomics of Gene Expression Lab

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

Genomics of Gene Expression Lab

Scenario

Raw FASTQs → polished expression report by morning.

ReAct Loop in Practice

1. **Think**: Review sample sheet & QC goals
2. **Act**: Run FastQC + STAR alignment
3. **Observe**: Check logs, flag weak samples
4. **Deliver**: Counts & summaries ready

Responsible Use

- Human review of flagged reads
- Track prompts & tool versions
- Log decisions for audit trail

What Is Context Engineering?

Genomics of Gene Expression Lab

Key Insight [1]

"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

What Is Vibe Coding?

Definition [2]

"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

Security [3]

40%

of AI code has flaws

Prompt injection
Hardcoded credentials

Reproducibility

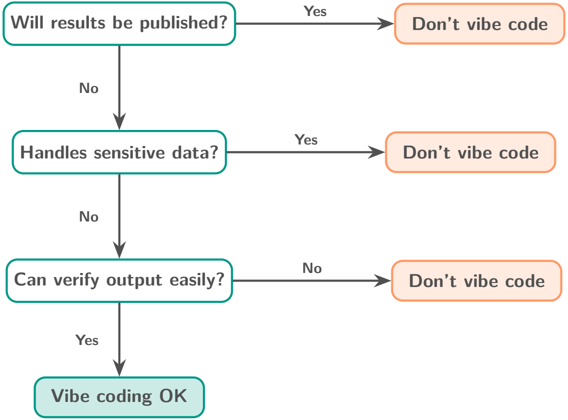
Non-deterministic

Can't verify
Journals need disclosure

Surprising Finding [4]

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

Best Practices Decision Tree



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

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

```
Make a step-by-step plan to:
1. QC my RNA-seq FASTQ files
2. Align with STAR (GRCh38)
3. Generate count matrix
4. Flag samples with $<$10M reads
5. Output: counts.tsv

Follow project standards in
GEMINI.md
```

ATAC-seq Example

```
Make a step-by-step plan to:
1. Process ATAC-seq reads
2. Call peaks with MACS2
3. Calculate FRiP scores
4. Filter peaks: FRiP $>$ 0.2
5. Output: filtered_peaks.bed

Follow project standards in
GEMINI.md
```

Key: Specific data type + QC thresholds + step-by-step request

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 for LaTeX CV Generation

The Workflow

Web form → Structured prompt → AI creates CV

Steps

1. **Gather information**
2. **Structure for AI**
3. **Generate output**

Key Insight

Better input

$\Downarrow$

Better output

See `examples/demo2/index.html`

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** → **Generate**
2. **Review** → **Commit**
3. **Pull request** → **Merge**

CI/CD & Unit Testing: Quality Automation

What is CI/CD?

Continuous **I**ntegration / **C**ontinuous **D**eployment — automatically test and deploy AI-generated code on every commit.

Unit Tests

- Fast feedback
- Catch regressions
- Document behavior

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: 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.

Beyond Agents: Humanoid Robots

Genomics of Gene Expression Lab

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

Learn More

Gemini CLI Documentation
Pandoc Manual
Software Carpentry [5]

Workshop Materials

Slides + Code on GitHub
Office Hours: Tuesdays 2-3pm

Thank you!

Questions? instructor@university.edu

References I

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

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

[3] 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`.

[4] 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.

[5] Software Carpentry Foundation. *Best Practices for Scientific Computing with AI Tools.* Online: Software Carpentry, 2024. URL: `https://software-carpentry.org/`.

Appendix: Security Vulnerability Example

Genomics of Gene Expression Lab

```python
# AI-generated code with vulnerabilities
def process_data(filename):
    # No error handling!
    f = open(filename)
    data = f.read()

    # SQL injection vulnerability
    query = "SELECT * FROM users WHERE name = '" + data + "'"

    # Hardcoded credential
    api_key = "sk-1234567890abcdef"

    return query

# Better version after review
def process_data_safe(filename):
    try:
        with open(filename, 'r') as f:
            data = f.read()
    except FileNotFoundError:
        return None

    # Use parameterized query
    query = "SELECT * FROM users WHERE name = ?"
    params = (data,)

    # Use environment variable
    api_key = os.environ.get('API_KEY')

    return query, params
```

Appendix: Sample GEMINI.md for Research

```
# Neuroimaging Analysis Project

## Compliance Requirements
- IRB Protocol: #2024-789
- HIPAA compliant data handling
- No cloud storage allowed
- De-identified data only

## Analysis Standards
- Software: FSL 6.0.5, SPM12, Python 3.10+
- Multiple comparisons: FWE correction p$<$0.05
- Effect sizes: Cohen’s␣d␣required
-␣Motion␣threshold:␣FD␣$>$␣0.5mm␣exclusion

##␣File␣Structure
data/␣␣␣␣␣␣␣␣␣␣␣␣#␣Raw␣DICOM␣files␣(never␣commit)
derivatives/␣␣␣␣␣#␣Preprocessed␣data
scripts/␣␣␣␣␣␣␣␣␣#␣Analysis␣code
results/␣␣␣␣␣␣␣␣␣#␣Statistical␣maps
logs/␣␣␣␣␣␣␣␣␣␣␣␣#␣Processing␣logs

##␣Quality␣Control
-␣Visual␣inspection␣of␣all␣registrations
-␣Motion␣parameters␣$<$␣3mm␣translation,␣3deg␣rotation
-␣tSNR␣$>$␣40␣for␣functional␣data
-␣Document␣all␣exclusions␣with␣reasons
␣␣
```