2026 年 6 月 16 日 @ 北京大学

AI 时代的研究范式：从模型能力到工作流设计

陶学臻

上海财经大学商学院



内容安排

一、AI 使用入门

二、Workflow 在研究中的应用

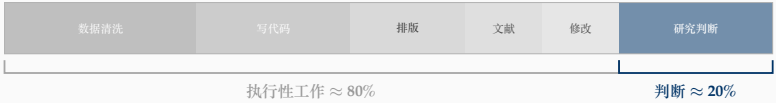
三、工作流的解剖与设计

四、风险、边界与研究者角色

五、补充：常用工具与现场练习

附录：给读者的延伸材料

研究者的一天：时间都花在哪了

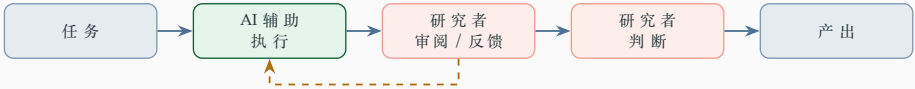


执行性工作

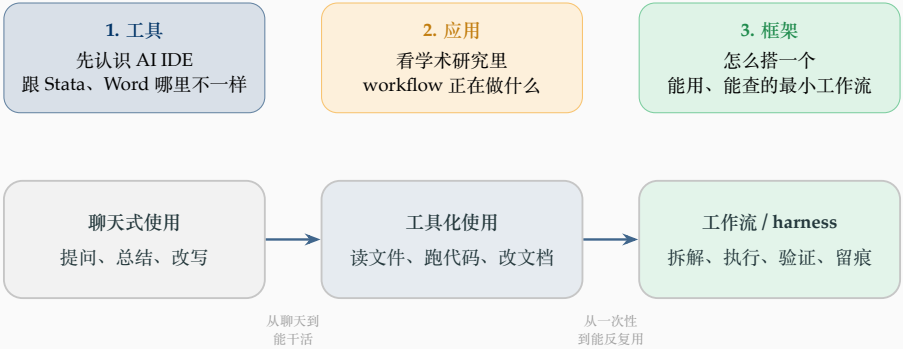
- 数据清洗、合并、缺失值处理
- 写脚本、跑回归、调试报错
- 表格、图、公式排版
- 文献检索、整理、引文核验
- 修稿、审稿回复、版本管理

研究判断

- 研究问题与对象的选择
- 识别策略、规格设定
- 结果的解释与边界
- 贡献的取舍、是否单发
- 数据伦理、隐私判断

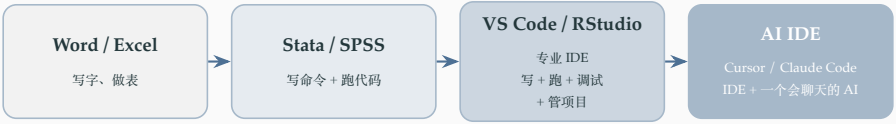


本讲内容与核心问题


1. 工具
先认识 AI IDE
跟 Stata、Word 哪里不一样
2. 应用
看学术研究里
workflow 正在做什么
3. 框架
怎么搭一个
能用、能查的最小工作流
聊天式使用
提问、总结、改写
工具化使用
读文件、跑代码、改文档
工作流 / harness
拆解、执行、验证、留痕
从聊天到
能干活
从一次性
到能反复用

一、AI 使用入门

工具谱系：从 Word / Stata 到 AI IDE



打开 AI IDE 看到什么：三个窗格



在哪（你的项目）　　改什么（代码与文稿）　　下指令（跟 AI 说）

典型任务：IDE 能做的三类工作

1. 文档处理

文字 · 文献 · 文档

- 读懂一个项目 / 一批资料
- 检索、整理、核验文献
- 起草、改写、润色文字
- 生成三线表 / 图 / Word / PPT

2. 代码处理

数据 · 脚本 · 运行

- 清洗、合并数据
- 写 / 改回归、检验等代码
- 跑代码，用大白话解释结果
- 调试报错

3. 搜集与多媒体

资料 · 网页 · 可视化

- 联网搜资料，整理成数据
- 建能查、能筛的网页
- 做可视化 / dashboard
- 采集后写脚本自检

三类活的开口方式一样：讲清要做什么＋给哪些材料＋要什么结果，AI 去执行，你来把关。

怎么跟 AI 开口：说清需求，它负责执行

1. 文档处理

读懂项目

「读一下这个项目，告诉我它研究什么、各文件都干嘛」

→ 扫文件树和 README，复述研究问题与文件分工

起草润色

「把这段结果写成一段引言，口语一点、别太 AI 腔」

→ 先出初稿，你改，再调措辞、去 AI 味

2. 代码处理

清洗数据

「这是原始数据，帮我清洗合并，标出缺失和异常」

→ 写清洗脚本，查缺失/异常，生成 codebook

写改代码

「给基线回归加平行趋势检验，改动先给我看再写入」

→ 读旧代码，给出 diff，你确认后写入

3. 搜集与多媒体

搜集资料

「把各省这类政策都搜来，整理成能查的数据表」

→ 联网采集，结构化成一张数据表

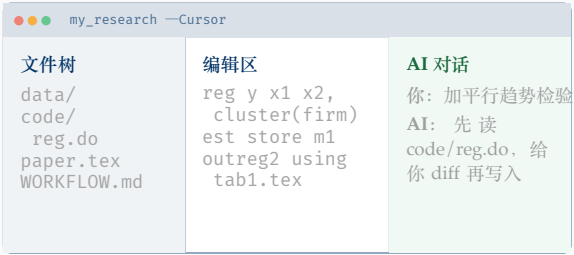
搭建网页

「用这份数据做一个能搜索、能筛选的网页」

→ 生成静态站，写脚本自检（断链/ID）

两种入口：GUI Agent 与 CLI Agent

GUI Agent（Cursor）



图形窗口，左中右三栏并排

- 文件树、代码、对话摆在一个窗口里，改哪儿、AI 在做什么都看得见
- 文件、表格、图点一下就打开看

CLI Agent（Claude Code）



黑色终端，只靠打字交流

- 只能看到它一句句的文字提示（在读什么、在改哪个文件）
- 看不到文件、表格、图本身，要另开编辑器才能看

AI 使用的几个层级


1. Prompt
提示词
单次问答
2. Context
上下文
给足背景资料
3. Rules
规则文件
项目专属约束
4. Skills
可复用模块
任务封装
5. Workflow
多步/检查
/可复现
本讲焦点

把意图说清楚：提示词 / 上下文 / 规则

1. 提示词工程——一次把任务、约束、输出格式写清楚

例：把“帮我做个回归”改成“panel data，对 firm 聚类，输出 outreg2 表，对照组用 EV 渗透率 $<5\%$ 的城市”

⇒ AI 直接给可跑代码，表头按论文格式排好

2. 上下文功能——用 @ 文件 / @ 文件夹 / 选中段落把材料丢进会话

例：把 `paper.tex` 的“Identification”段和 `reg.do` 一起 @ 进去，问“代码和这段设定一致吗”

⇒ AI 指出“你写的是双向 FE，代码里只有单向”

3. Rule Engineering —写 `WORKFLOW.md` / `.cursorrules`，规则常驻

例：规则里写死“所有回归 cluster=firm；所有图带 90% CI；所有表用 outreg2”

⇒ 之后每次跑回归都自动套，换台机器、换个 session 也守这套

示例：政策文件汇总成一个网页

研究问题

人工智能在高等教育发展中的相关政府政策与高校政策有哪些？政策导向如何，当下推进到什么阶段？

任务：广泛采集国家、教育部、各高校的 AI 教育 / 治理政策，精选深读，做一个能查、能比、带语义分析的政策图谱网站。

提示词（按 4 步写给 AI）

1. **搜集**：联网检索各主体的 AI 政策原文，记标题、时间、文号、URL，攒成全量证据库（**1629 份**）
2. **语义分析**：对全文做语义记录（**61 份**），抽概念、政策方向、层级关系
3. **建站 + 校验**：做 6 个标签页的图谱站，再写 `validate_site.mjs` 查数据加载、ID、断链

产物：6 个标签页

总览 · 顶层逻辑 · 高校实践 · Workflow 启发 · 语义分析 · 全量证据库 `ai_edu_policy_site/`

```
index.html   render.js
subject.html   detail.html
data.js   styles.css
collection/  1629 份
semantic_analysis/  61 份
uploads/   tools/validate_site.mjs
```

做一个网站：前端 + 后端，分别跟 AI 说什么

前端——用户看到的界面

管什么：页面布局、按钮 / 表格 / 图表、点击后的反应、手机端适配。

怎么说：

「做一个带搜索和筛选的列表页」
「蓝白配色，手机上也能正常显示」

后端——用户看不到的部分

管什么：存数据、处理请求、连数据库和外部接口、登录和权限。

怎么说：

「表单提交后存进数据库」
「加登录功能，每天定时抓一次数据」

AI 能干的

写代码、连数据库、调试报错、部署上线。

得你自己定的

做哪些功能、数据准不准、隐私和安全怎么把控、好不好用。

只放内容的静态网站一般只要前端；需要存数据、登录、实时更新才加后端。手机 App 同理——先做网页版把功能跑通。

网站上线之后怎么改

截图反馈

看到哪里不对，截个图发过去：「这个按钮位置偏了」「这里太挤了」。AI 根据截图找到对应代码改。

文字描述

说清在哪个页面、点了什么、出了什么毛病。AI 去找对应的文件和函数，列出改动，你看过了再写入。

加新功能

想加筛选、排序、弹窗、登录、图表——直接说就行，AI 写好代码，把前后端接上。

每改一次，存一版

用 git 存快照（见后面 git 那页）：改坏了一条命令退回去，也方便对比每次改了什么。

分工始终不变：你说哪里不对、要加什么；AI 去改代码、跑通、部署。

真要做 App：在网页基础上打包

从网页到 App，多哪些事

打包方式（网页代码直接复用）：

- 桌面：**Tauri / Electron** 打包成 Mac / Windows 应用
- 手机：**PWA**（网页直接装到桌面），或 **React Native / Flutter**（一套代码出 iOS + Android）
- 原生 Swift / Kotlin 最重，一般用不着

比网页多出来的：真机调试、应用商店账号和审核、推送 / 离线 / 权限等原生功能。

例：我做的 Research Planner

干什么用：研究项目耗时不确定，普通日历按截止日排，不合适。这个工具按进度排，随手记想法，慢慢整理；搁了很久重新打开，先告诉你上次做到哪、卡在哪。

怎么做的：前端 React + TypeScript，后端 Fastify + Prisma，用 **Tauri 打包成桌面 App**——就是左边说的路子。

两种视图：时间线（简化甘特图）和看板（分"在想"和"在做"）；支持多项目、本地搜索、标签、导出 Markdown / HTML。

什么是 Skill

Skill 是一个带触发器的本地文件夹。AI 平时只看见一张目录卡片（名字 + 触发时机），匹配上任务才把里面的内容按需读进来——所以一台机器装几十个 Skill 也不会撑爆上下文。

一个 Skill 文件夹

```
my-skill/
  SKILL.md          必需:YAML + 正文
  scripts/          可执行脚本
  references/       长文档
  assets/           模板 / 字体 / 图标
```

Skill 跟前面讲的 Rule 哪里不一样

Rule（`WORKFLOW.md` / `.cursorrules`）：一个项目里常驻，AI 每次都看。

Skill：装在机器上、跨项目复用；只有目录卡片常驻，正文按需触发。



几个热门 Skill 例子

例子	触发场景	里面装了什么	产出
PubMed 文献观点映射	上传 CSV + abstract TXT，要求整理一批医学文献	PMID / DOI 对齐脚本、观点字段 schema、局限性提取规则	回填后的文献观点表
frontend-design	从需求或草图生成前端页面、组件、视觉稿	组件示例库、字体 / 配色资源、视觉检查清单	可运行页面 / 设计稿
documents / pptx / xlsx	做 Word、PPT、Excel，而不是只要一段文字	`.docx`/`.pptx`/`.xlsx` 模板、渲染脚本、视觉自检流程	真实文件 + 渲染校验
skill-creator	做一个新的 Skill 或审查已有 Skill	YAML 写法、触发测试用例、目录规范、反例清单	可安装的 `SKILL.md`
Claude Autoresearch / ARIS	睡前启动，agent 连续跑实验、查新、pilot	循环策略、评估标准、失败后的续跑规则	实验日志 / 候选方案
guodegang	把一段讲座内容改成郭德纲风格	角色规则、表达 DNA、惯用句式表	风格化文本
Gans research advisor	给一个现象，按 Gans 的方法搭一个最小理论模型	95 篇语料地图、25 篇模型拆解、12 步建模 workflow	研究备忘 + 命题草稿
de-ai	给一段中文初稿去 AI 味	判定块清单、AI 句式骨架、高频词替换表	改写稿 + 修改对照

同一段内容，两种说法

输入：本讲主要内容（约 300 字）

这次讲座从研究者一天的工作出发，说明 AI 不只是聊天问答，而是可以进入文件、代码、数据、文献和写作流程的研究工具。讲座先区分 prompt、context、rules、skill 和 workflow：prompt 解决单次表达，context 提供材料，rules 固定项目约束，skill 封装可复用能力，workflow 则把任务拆成步骤，配置工具和角色，设置 Gate 与人工检查点，并保留日志、版本和记忆。目标不是让 AI 替代研究者，而是让它承担数据整理、代码执行、格式检查、资料归档等执行性工作。研究者把注意力留给选题、识别策略、结果解释、证据边界和伦理判断。好的工作流要能查、能退回，下次还能接着用。

输出：guodegang skill 转换后

我以郭德纲的视角和你聊，基于公开言论提炼，不是本人观点啊。这个事儿啊，别把 AI 当茶馆里陪您闲磕牙的，它是后台干活的伙计。您得先立规矩：哪句是开场白，哪包是材料，哪条是班规，哪门是手艺，哪一整套是流程。prompt 就是您吩咐那一句；context 是箱子里的道具；rules 是台上的尺寸；skill 是练熟的活儿；workflow 是从开场到返场的章法。查资料、跑代码、改格式、记日志，这些让它去奔。可选题值不值、识别靠不靠谱、话说到哪儿收住，还得您自己定。观众买不买账，不能让伙计替您担。好 skill 就是把规矩和分寸封进去，下回再使，还不走样。活儿可以交出去，主意不能交出去，这才叫自己说了算。

把 Joshua Gans 的研究方法做成一个 Skill

怎么造的

一份 PRD 分成 **8 个阶段**，每步写清读什么、交什么、做到哪算完；AI 跑了约一天：

1. 收集 95 篇 NBER 论文，**逐篇做笔记**
2. 挑 25 篇**逐篇拆细**，每篇拆成 **7 个文件**（研究问题 / 模型图 / 命题 / 推导 / 写作分析 / 小结 / 迁移）
3. 提炼出共用的做法：选题、建模、推导、写作
4. 整理成方法文档、模板、评测脚本
5. 做成一个带触发器的 `SKILL.md`

文件架构

```
gans-research-advisor/
  SKILL.md                    触发器 + 操作流程
  references/                 307 个文件
    paper_notes_95/           95 篇论文笔记
    deep_cases/               25 个案例，每个 7 文件
    *.md                      34 篇方法 / 模板 / 清单
  scripts/                    4 个：选案例 / 看案例 / 评测
  agents/openai.yaml
```

开源仓库 `https://github.com/xtao-sh/gans-research-advisor` · 300+ 文件 · 约 32 万字 · 全部由 AI 生成

skill 里每个文件装了什么（节选）

文件	内容（节选）
`SKILL.md` 触发器 + 方法骨架	YAML 里写 `name/description` 触发词；正文给方法链「现象 → 缺失的摩擦 → 最小模型 → 命题 → 含义」
`paper_notes_95/` 95 篇论文笔记	「# w13234 *Uncertain IP Rights* · Authors: Gans, Hsu, Stern · NBER 13234 · JEL L24」——每篇的元数据 + 要点
`deep_cases/` 25 篇拆细，每篇 7 个文件	`research_question`:「区块链降低了哪些经济成本？何时让去中心网络能与中心化平台竞争?」 `model_map`: 经济对象表（verification / networking cost、token…）
`*.md` 模式目录 如 common_friction_catalog	「Friction 1：预测变便宜、判断仍稀缺 → 决策结构改变」（标注用于 w24243 / w24626 / w26673）
`*.md` 模板 如 proposition_template	填空骨架：「Proposition X / Plain-English Claim:（待填） / Mechanism:（待填）」
`scripts/select_cases.py`	docstring 写「Rank Gans corpus cases for a user idea」——输入一句话，从 95 篇里挑出最相关的几篇
`agents/openai.yaml`	`default_prompt: "Use $gans-research-advisor to turn my idea into a Gans-style memo"`

每篇都拆成这 7 个文件：研究问题、模型图、推导、命题、写作分析、小结、迁移笔记。

用这个 Skill 分析一个新问题

怎么用

一句话触发：「用 Gans 风格看这个 idea」、「把这个想法变成一个模型」、「按 Gans 的逻辑改我的 introduction」。

它会从语料里找出 **3–6 个最接近的案例**，照这条链分析你的问题，给出修改建议。



例：“前沿大模型为什么免费开源”（Llama / DeepSeek / Qwen）

步骤	这个案例里填进去的内容
缺失的摩擦	租金主要落在**互补品**（云算力、部署、生态）和**事实标准**上，靠模型本身赚不到多少
最小模型	厂商选开源 / 闭源；闭源租金 π、互补 + 生态收益 v、外溢给对手 s
命题	$v - s \geq \pi$ **时开源**（净收益盖过闭源租金），否则闭源

de-ai Skill：删什么

决定什么压根别上幻灯片——slide 是显示用的，结论、过渡、舞台话都口头说。

检查点	AI 味写法	de-ai 改法
底部判定块	“核心问题”“为什么算 X”“这意味着”	默认删掉; 除非是全场必须带走的结论
抽象类别	“传统工具”“研究判断”“执行性工作”	换成具体名:Stata、Word、识别策略、解释、伦理
戏剧化容器名	“失败相册”“X 图鉴”“X 全家桶”	直接写“4 个常见失败和怎么修”
舞台辞令	“接下来我们要”“请台下哪位”“没有标准答案”	从 slide 上删掉, 现场口头说
元叙述前缀	“完整走查 W1”“关键概念 X”每帧标题背一遍	section 已经定位了, 帧标题只放本帧具体内容

de-ai Skill：改什么

决定留下来的句子，把 LLM 句式骨架、抽象词、营销语换成具体动作。

它会抓的句式 / 词	问题	改法
“从 X 到 Y”“越... 越...”	教科书概括，骨架太顺	换具体名:Stata、Word、`WORKFLOW.md`
“可 X、可 Y、可 Z”(可检查 / 可验证 / 可追溯)	抽象形容词排队	改动词: 能查、能验、能跑回去看
“X 是一种 Y”	定义句像百科	列具体文件:`SKILL.md` / scripts / references
“帮你”“替你”“一键”“直接”	像产品文案在卖好	去掉用户承诺，只说文件、脚本发生了什么
“形成闭环”“沉淀”“被组织起来”	抽象 + 被动，没说真动作	换动词: 跑通、整理成、用上、存进
“真正的”“本身”“本质上”	给普通句子加高潮	删掉；句子还成立再留
“一句说明 + 三条动作”	用结构模板代替内容	直接写具体改了什么

AI 应用的四个趋势



第 4 条：Gartner 统计，多智能体系统的企业咨询量从 2024Q1 到 2025Q2 涨了约 14 倍。

二、Workflow 在研究中的应用

The GitHub Landscape of Automated Research Workflows

Open-source AI tools for literature review, report generation, experiments, and autonomous "AI Scientist" systems

ⓘ Based on a synthesized report of leading GitHub repositories; star counts are approximate and change over time.

1 EXECUTIVE SNAPSHOT

Most-starred overall: karpathy/autoresearch (~83.9k)

Deep-research leaders: STORM (~28.2k) and gpt-researcher (~26.5k)

Autonomous AI Scientist frontier: Sakana AI-Scientist (~12.1k)

Core trend: from single-purpose paper assistants to multi-agent end-to-end research systems

2 THE 8 STRANDS OF THE ECOSYSTEM

1. Deep-Research Report Agents
Plan, search, synthesize long-form cited reports.
Examples: gpt-researcher, STORM, open_deep_research, DeerFlow.

2. Literature Review / Knowledge Curation
Survey perspectives, structure knowledge, and build outlines.
Examples: STORM, Co-STORM.

3. Autonomous AI Scientist Systems
Attempt idea → experiments → paper → review.
Examples: AI-Scientist, AI-Researcher, Agent Laboratory.

4. Scientific RAG / Paper Q&A
Grounded answers from scientific PDFs with citations.
Example: PaperQA2.

5. Paper Reading / Writing Assistants
Summarize, translate, polish, and review papers.
Examples: gpt_academic, ChatPaper.

6. ML Experiment Automation / R&D
Automate coding, debugging, benchmarking, and iterative improvement.
Examples: autoresearch, RD-Agent, AIDE.

7. Domain-Specific Science Agents
Specialized agents for biomedical or materials science workflows.
Examples: Biomni, SciAgents.

8. Reference Management / Paper Discovery
Citation workflows and paper recommendation.
Examples: zotero-better-bibtex, arxiv-sanity.

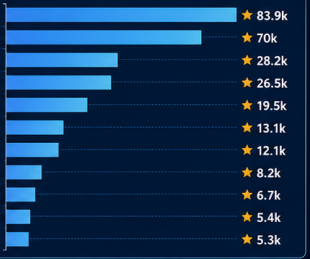
4 WHAT INNOVATIONS ARE CONVERGING?

- Agentic plan-and-execute loops
- Multi-agent role specialization
- Tree search over ideas and code

Evaluation design is increasingly the bottleneck.

- Retrieval + contextual summarization for trustworthy citations
- Fast, verifiable evaluation loops
- Benchmark-driven selection (Deep Research Bench, LitQA2, MLE-Bench)

3 LEADING REPOSITORIES BY APPROXIMATE STARS



5 PRACTICAL RECOMMENDATIONS AND CAVEATS

Choose by Task

- Report generation: gpt-researcher, STORM, open_deep_research
- Scientific literature Q&A: PaperQA2
- Paper polishing / translation: gpt_academic, ChatPaper
- Experiment automation: autoresearch, RD-Agent, AIDE
- End-to-end autonomy: AI-Scientist-v2, Agent Laboratory
- Reference workflows: zotero-better-bibtex

Key Caveats

- Star counts change quickly
- Some benchmark or performance claims are project-reported
- Autonomous code execution creates real security risk — use isolated sandboxes / Docker
- License, maintenance, and governance matter for production use
- High popularity does not always equal highest scientific novelty

Autonomous AI Research Systems

How 'AI Scientists' moved from research concept to real academic adoption, 2023–2026

Survey of end-to-end paper generators, hypothesis engines, algorithmic-discovery systems, institutional response, and key caveats.

1 EXECUTIVE SNAPSHOT

Adoption is real
2024–2026 saw AI systems generate papers, propose hypotheses, and support lab-validated discoveries.

Three main strands
(1) End-to-end ML paper generators; (2) Hypothesis-and-discovery engines; (3) Verifiable algorithmic-discovery systems.

Most credible evidence
Nature-linked wet-lab or computational validation: Robin, Virtual Lab, Coscientist, A-Lab, plus Google co-scientist claims.

Big constraint
Novelty, reproducibility, and governance remain the bottlenecks — not just generation capability.

★ *The field is early, real, and controversial.*

2 THREE ARCHITECTURES OF 'AI SCIENTIST' SYSTEMS

A. End-to-end ML research / paper-generation systems

Idea → experiments → code → figures → manuscript → AI review.

Examples:
Sakana AI Scientist v1/v2, Carl, Zochi, AIDE, Agent Laboratory.

Key traits
- Agentic tree search
- Automated peer-review loops
- Code execution and experiment management

B. Hypothesis-generation and discovery engines

Multi-agent systems paired with wet labs or computational pipelines.

Examples:
Google co-scientist, FutureHouse Robin, Stanford Virtual Lab, Coscientist, ChemCrow, A-Lab, SciAgents.

Key traits
- Literature synthesis
- Hypothesis ranking
- Lab / simulation validation

C. Verifiable algorithmic-discovery systems

Systems that produce mathematically or computationally checkable results.

Examples:
FunSearch, AlphaTensor, AlphaEvolve.

Key traits
- Formal verification
- Algorithm improvement
- Higher trust via objective validation

The more directly verifiable the output, the more credible the autonomy claim.

3 MAJOR MILESTONES, 2023–2026

Timeline of Key Milestones

- **2023** — **CMU Coscientist**: autonomous chemistry execution (Nature 2023)
- **2023** — **Berkeley A-Lab**: closed-loop materials synthesis (Nature 2023)
- **2024** — **Sakana AI Scientist v1**: end-to-end ML-paper pipeline
- **2024** — **ChemCrow**: tool-using chemistry agent gains visibility
- **2025** — **Sakana AI Scientist v2**: AI-generated paper passes ICLR workshop review (withdrawn after acceptance)
- **2025** — **Autoscience 'Carl'**: Tiny Papers + workshop acceptance claims
- **2025** — **Intology 'Zochi'**: ACL 2025 main-conference acceptance claim
- **2025** — **Stanford Virtual Lab**: nanobody design, Nature 2025
- **2025** — **AIDE**: strong MLE-Bench performance via tree search in code space
- **2026** — **FutureHouse Robin**: dry-AMD drug-repurposing result published in Nature 2026
- **2026** — **Institutional response intensifies**: ICLR / ICML / NeurIPS policy tightening

Representative Systems at a Glance

#	System	Domain	Open source	Core innovation	Milestone
1	Sakana AI Scientist v2	ML	Yes	Progressive agentic tree search + VLM figure review	ICLR 2025 workshop accept (withdrawn)
2	Carl	ML	No	Automated research scientist with strong acceptance claims	ICLR Tiny Papers + workshop
3	Zochi	ML/Bio	Partial	End-to-end artificial scientist	ACL 2025 main proceedings
4	AIDE	ML engineering	Yes	Tree search in code space	~4× medals on MLE-Bench
5	Agent Laboratory	ML	Yes	Cheap human-in-loop pipeline	84% cost reduction claim
6	Google co-scientist	Biomed	No	Elo idea tournaments + debate	AML / superbug hypotheses
7	FutureHouse Robin	Biology	Yes	Hypothesis + literature + analysis pipeline	Nature 2026
8	Stanford Virtual Lab	Biology	Yes	AI PI + critic multi-agent team	Nature 2025
9	SciAgents	Materials	Yes	Knowledge-graph multi-agent reasoning	Advanced Materials 2025
10	AlphaEvolve	Math / CS	No	Evolves whole codebases	Verifiable algorithmic gains

4 THE ACADEMIC SYSTEM IS RESPONDING

Journal authorship norms
- Nature / Science do not allow LLMs as authors.
- AI use must be disclosed.
- Confidential manuscripts should not be uploaded to public LLM tools.

Conference policy tightening
- **ICLR 2026:** LLM-use disclosure rules for authors and reviewers.
- **ICML 2026:** prompt-injection restrictions, detectors, and no-LLM reviewer policy.
- **NeurIPS 2025:** stronger scrutiny for hallucinations and AI-assisted review.

Why the alarm?

21% of 75,800 ICLR 2026 peer reviews were reported as fully AI-generated; more than half showed signs of AI use.

- Hidden prompt-injection instructions surfaced in preprints.
- Hallucinated citations appeared in major-conference submissions.
- Submission quality pressure and review integrity are worsening.

5 PRACTICAL TAKEAWAYS

What to do now
- Use literature agents as accelerators, not authorities.
- Prefer lab-in-the-loop or simulation-validated systems for scientific discovery.
- For prentical open workflows, AIDE and Agent Laboratory are currently among the most usable.
- Adopt explicit AI-use disclosure and review-integrity safeguards.
- Treat verifiable-output systems as the trust benchmark.

Persistent caveats
- Many "first" claims are tied to commercial incentives and publicity.
- Human cherry-picking often remains part of the pipeline.
- Novelty may be literature recycling rather than true discovery.
- Reproducibility is uneven; some headline systems show fragile results.
- This field changes extremely fast — capabilities and policies can quickly become outdated.

Bottom line: *AI scientists are no longer hypothetical, but credibility depends on verification, disclosure, and human scientific judgment.*

传统 AI 使用 vs 研究 workflow

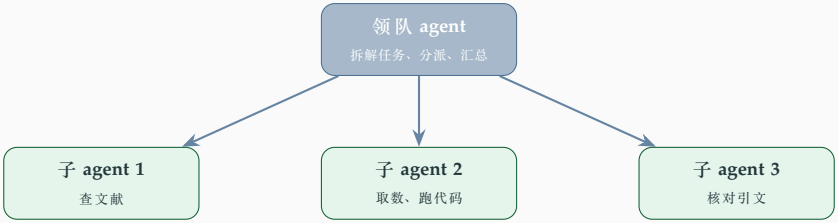
传统 AI 使用

- 问一次，答一次
- 结果很难复查
- 上下文容易丢
- 经验难以积累

研究 workflow

- 任务有步骤、有中间产物
- 中间步骤都能查
- 失败留下来，写进规则
- 下一次还能用

多智能体：一个领队，几个分工的子 agent



每个子 agent 有独立上下文窗口，互相看不见，所以能真并行

Anthropic 的实测

Claude 的 Research 功能就是这个结构：领队规划后并行派 **3–5 个子 agent**。内部评测里，多智能体（Opus 领队 + Sonnet 子 agent）比单模型成绩高 **90.2%**；代价是 token 用量约 **15 倍**。

接下来的案例都是这个形状

DeerFlow 的 Lead + sub-agents、Paper Factory 第 2 步的 6 路并行、Sant'Anna 评审里的双盲 referee——都是领队分派、子 agent 并行。

来源：anthropic.com/engineering/multi-agent-research-system

#1 DeerFlow (ByteDance)：长任务 SuperAgent harness（66.8k stars）

1. 任务进门

- 用户从 **IM** 入口下任务（飞书 / 微信 / Telegram …）
- Message Gateway 接收，转给 Lead
- 无须公网 IP（long-poll / WebSocket）

2. Lead Agent 拆解

- 读任务描述
- 用 **todo** 工具拆解成多个独立子任务
- 对每个子任务决定：自己做 / 起 sub-agent / 起 sandbox

3. Sub-agents 并行

- 每个 sub-agent 独立 **context** + 工具集
- 在 **Sandbox** 跑（文件 / 代码 / shell / 网络）
- **progressive** 加载 **Skill**：用什么加什么
- 返回结构化结果

4. 汇总 + 落盘

- Lead 收子结果，摘要 + 压缩上下文
- 中间产物卸到 **filesystem**，不占 token
- 关键决定写入 **Memory**（跨会话能查）
- 输出文件：报告 / PPT / dashboard

适合什么任务 & 维护状态

跨分钟到小时的长任务：研究 + 写代码 + 出报告 + 做 PPT + 搭 dashboard；附带 200+ MCP server 集成。字节出品，**2026-05** 仍在每天 **push** □。

#2 STORM (Stanford)：自动综述报告生成（28k stars）

1. Research

- 扫同主题已有文章，找出 N 个不同视角
- 每个视角:**writer** □ **expert** 多轮模拟对话
- Expert 发言前必先检索真实来源
- 把所有 URL + 片段收进知识池

2. Outline

- 读完 Stage 1 全部资料
- 生成层级大纲: 章 / 节 / 小节
- 每节标注该引用哪几条 **reference**

3. Article

- 按大纲逐节生成正文
- 每段引用对应的 Stage 1 片段
- 插入 **inline citations**
- 章节之间独立写, 不互相参考

4. Polish

- 通读全文, 加 **introduction / summary**
- 跨节去重: 合并不同章节里重复的段落
- 最终格式化输出

Co-STORM 变种 + 产出

把 Stage 1 改成人 **+ LLM expert + Moderator** 三方对话 + 动态 mind map □ 协作式综述。最终产出 Wikipedia 风格长报告，适合做文献综述前置稿、领域入门材料；不直接当最终稿。

#3 Paper Factory (MIT)：端到端自动论文流水线（81 stars，2026-05-05 开源）

1. 起跑（Steps 1-3）

- 联网搜 20-30 篇文献
- 清洗数据 + 构造变量
- **viability gate**：数据能不能支撑
- **6 路 agent 并行**各跑一个 finding 包，每个配 critic
- decider 从 6 个里选 1

2. 架构（Step 4）

7 个并行扩展 agent：

- identification
- heterogeneity
- primary robustness
- objection robustness
- descriptive support
- supplementary
- moonshot

3. 起稿（Steps 5-10）

- 论证 / 数据 / 方法审计
- 起草 `paper.tex`
- **Gate 1**：codex 复算每个数字——不过不让进
- 评审 + 修订

4. 收尾（Steps 11-16）

- **Gate 2**：校验论证 + 数据一致
- 引文核验 + 表格 + 摘要
- **Step 15 去 AI 味**（derobotification）
- 编译 PDF + 清理中间数据

设计亮点 + 与 DeerFlow / STORM 的区别

2 道硬 Gate 卡住前进；**Step 2 六路并行 + critic** 强制角度多样化；**Step 15 专设 derobotification** 把去 AI 味也工程化。区别在：它跑完是一份**编译好的 38 页 PDF**，不是 workflow 片段。作者 Nathan Wilmers（MIT Sloan 劳动经济学），配套 SocArXiv 论文同步发布。

实测：把 Paper Factory 跑了一晚

时间线

21:59	启动 Step 1
22:51	Step 1d viability gate PASS
23:16	Step 1e 描述统计完成
23:33	Step 2 六路并行启动
00:20	Step 3 decider 选 Stream 1
01:22	Step 4 七路扩展跑完
02:34	Step 7 claude API 崩（故障）
03:11	Gate 1 PASS（复算数字）
03:48	Gate 2 PASS_WITH_DIRECT_FIXES
04:30	Step 15 去 AI 味跑完
04:31	**All 16 steps complete**
总用时	**6 小时 32 分钟**

产物

38 页 PDF + 77 张图 + 204 个表 + 42 个可重跑的 `.do` 脚本 + 50+ 份 markdown 中间报告

两个最有意思的观察

1. 数据自救：agent 发现关键文件是 0 字节空文件，主动翻出 schema 一致的旧副本替代，并写进 `data_wrangle.md` 报告。

2. 主动降级：Step 1d 发现"13 周里只有 1 个 open release"，自动把主线问题从位移事件研究改成质量-价格 grid 描述——**而不是 kill 整个项目**。

中间手动修了 3 个 macOS 兼容性 bug（`sed` / `timeout` / `stat` 的 GNU 写法 + snapshot 路径问题）。剩下全是机器自己的事。

几个开源 workflow 工具：5 大类型

类型	代表项目（含 GitHub stars）	这类工具大概在做什么
平台型	Dify（**141k**）、Activepieces（**22.2k**）、PySpur（**5.7k**）、Eko（**4.9k**）、LLMStack（**2.3k**）	拖拽节点搭 LLM 流水线；不写代码就能连数据源、模型和外部 API；适合非技术研究者搭一次性 demo，或团队里共享一套固定流程
编排型	MetaGPT（**67.9k**）、DeerFlow（**66.8k**）、OpenAI agents-py（**26.2k**）、DeepAgents（**22.6k**）、Swarm（**21.5k**）、CAMEL（**16.9k**）、MS agent-framework（**10.4k**）	把任务拆给多个 agent（规划者 / 执行者 / 评审者），框架负责调度、消息传递、工具调用；做长任务、复杂分工和 critic-fixer 这类多角色 workflow 的骨架
研究型	**Paper Factory**（**81, 新**）、MinerU（**62.7k**）、STORM（**28.2k**）、Haystack（**25.2k**）、PaperQA2（**8.5k**）、MindSearch（**6.9k**）、DeepResearchAgent（**3.4k**）	从 PDF / 网页 / 论文库里抽资料，做检索增强生成（RAG）和引文核验；输出综述、文献回顾、报告和知识问答。**Paper Factory** 是其中最深的：端到端 16 步跑出 38 页 PDF，带 2 道 Gate
治理型	mission-control（**4.8k**）、agentic-radar（**966**）、syftr（**335**）	追 agent 调用链、记 token 成本、扫提示注入、跑离线评测；workflow 上线之后的可观测性和质量底盘
引擎型	Conductor（**31.8k**）、orla（**243**）	把 workflow 编译成持久化状态机：失败可重试、状态可恢复；长时间任务（小时 / 天级）跑起来不丢进度的执行层

stars 取自 GitHub（Paper Factory 为 2026-06-11，其余为 2026-05-12）。

我做的另一个工具：Workflow Designer

`https://taich.tech`

40 个研究 workflow，分 4 类

- **Paper Factory**（MIT / Nathan Wilmers）——16 步端到端论文流水线
- **Auto Research**（我的 v11）——自家研究 workflow
- general-research ——通用研究 workflow
- highstar-research ——GitHub star 高的研究类项目

上面只是几个有代表性的；完整 40 个收录在站点 workflow 库里。

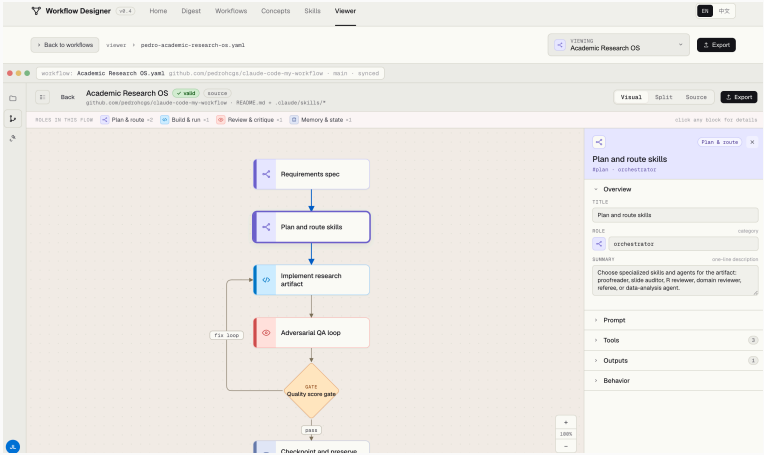
为什么做这个

散在 GitHub 各处的 workflow 风格差异大、文档参差不齐、没法对比。

这个站点把它们抽取成统一 schema：每个 workflow 的 prompt、agent 职责、产物清单都能直接浏览、下载。Paper Factory 那份 195 KB 结构化描述就是这么从 `github.com/nwilmers/paper_factory` 自动生成的。

6 个浏览页（首页 / 主题汇总 / workflow 库 / Concepts 解剖 / Skills 注册表 / Designer 可视化器）· 中英双语 · 部署在腾讯云

Workflow Designer：可视化查看器



Pedro Sant'Anna 的 Academic Research OS 渲染成流程图：规划 → 实现 → 对抗式 QA 循环 → 质量 Gate → 检查点。点任意节点可以看角色、prompt、工具和产物。

类型 1：知识生产 workflow



❖ **应用场景** 文献、项目文档和数据说明太多时，光靠摘要不够；workflow 要把它们整理成**能查、能反复用、能追溯出处**的知识库。

代表案例

- STORM、PaperQA2 / FutureHouse、Elicit、Undermind
- Karpathy 式项目 wiki 与长期笔记
- 研究型知识库：引文图、原文证据、结构化卡片

可迁移设计

- 用结构化卡片替代自由文本总结
- 把引文、页码、出处作为一等字段
- 先单篇抽取，再跨文献聚合

最小版本

建一个 `raw/` 文件夹，定一个卡片 schema，写一个抽取模板。AI 只准产出符合 schema 的卡片。

类型 2：实证、复现与政策分析 workflow



❖ **应用场景** 实证和政策研究里，AI 可以取数、写代码、跑回归、做分类、生成表格。关键是每一步都得留下能跑的代码和能查的记录。

代表案例

- APE：自动取数、回归、写作和多阶段评审
- ImpactAI / EconBERTa：从发展经济学文献中抽取因果证据
- IMF / Fed / ECB：统计问答、文档分类、通胀 nowcasting

可迁移设计

- 把 API 取数、代码沙箱、表格输出连成流水线
- 用 R / Stata / Python 输出和测试当 verifier
- 保存数据版本、脚本版本和表格生成链条

最小版本

AI 负责写脚本、解释报错。脚本对不对，让真实程序、测试、数据版本和复现输出说了算。

类型 3：写作、教学与长期项目 workflow

❖ 应用场景 长文档、课程讲义、论文项目、软件包都得跨很多轮对话。难点不是某一次答得好不好，是长期保持一致、记得住事、能反复用。



从下往上：规则、记忆、检查点逐层增加

代表案例

- Sant'Anna 的课程材料 workflow
- Karpathy / Simon Willison 的项目 wiki 与计划先行模式
- 讲义、教材、代码项目的长期维护

可迁移设计

先写 `CLAUDE.md`；重复的任务做成模板；评审和修改分开；重要决定写进 Memory。

最小版本

别一上来就搭整套系统。先写一份规则文件、一个最常用的模板；等重复错误出现，再加 Gate。

类型 4：高风险产出的验证 workflow



❖ 应用场景 产出要进论文、评审、公开材料或政策讨论时，workflow 首先要防错出门，其次才谈提速。

代表案例

- APE 的 staged review 与外部模型评审
- AI Scientist v2 的树搜索、图表 critic 和 workshop 评审
- AlphaProof / Lean、ChemCrow / Coscientist 的外部 verifier

可迁移设计

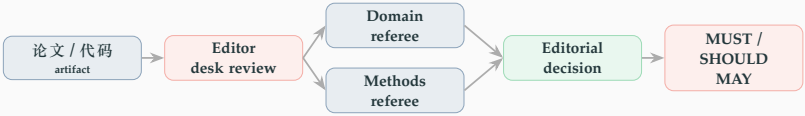
- 生成者不能给自己打最终分
- 完整性核验和内容评审分离
- 失败以后更新规则，避免下次重复同样的问题

什么时候不要用

这套结构只适合高风险场景。在探索阶段就上重型评审，反而会把流程逼僵。

Sant'Anna 系：把论文评审做成 workflow

❖ 核心想法 不只是让 AI "帮我审一下论文"，而是模拟一轮 journal review：先过编辑门，再送两个盲审，最后由 editor 合成决定。



/review --peer [journal]

- editor 先查 novelty 和 journal fit
- 两个 referee 独立报告，不互相看
- 每个 major comment 写清楚：什么证据会改变判断
- editor 把问题分成 FATAL / ADDRESSABLE / TASTE

为什么像真实评审

- reviewer 有不同 disposition：credibility、measurement、theory、skeptic
- R&R 第二轮读取上一轮报告，逐条判断 resolved / partially / not addressed
- stress test 可以把 referee 设成最挑剔版本

可迁移设计

评审不是一句 prompt，而是一个带角色、证据、记忆和决策门的流程。最关键的是：critic 不改稿，editor 做裁决，报告要落盘。

Review workflow 项目列表：可以借什么

项目	主要对象	最值得借鉴的 workflow 逻辑
`clo-author`	经济学论文、代码、R&R	editor desk review → blind referees → editorial decision；第二轮读取上一轮 review memory
`claude-code-my-workflow`	讲义、论文、复现包、长期项目	contractor mode：plan → implement → verify → review → fix → score；80/90/95 gate
`AI-research-feedback`	paper / PAP / grant	轻量 slash command；6 个 reviewer 并行，最后保存成 dated report
`codex-review`	plan 与 implementation	先 snapshot approved plan，再让 Codex 按 plan 审实现；decisions ledger 防止结论丢失
`codex-plugin-cc` / `review-loop`	代码变更	Stop hook review gate：Claude 要结束时自动触发 Codex review，有问题就先处理
`openreview`	GitHub PR	PR comment 触发；sandbox clone、跑 lint/test、发 inline suggestions，必要时 push fix
`agent-review-panel`	code / plan / docs	多 reviewer 独立审查、辩论、blind final、claim verification、judge 裁决

一句话总结

这些项目的共同方向：把 review 从“模型意见”改造成**可路由、可验证、可复审、可回退**的质量控制系统。

AI workflow 在经管研究中的几个例子

Brynjolfsson Stanford

一个周末，用 Claude Code 跑完数据清洗、回归、图表和初稿，最终发表在 PNAS

~1 小时 （原本 2–3 周）

Pedro Sant'Anna Emory

用 Claude Code 多智能体工作流（critic / fixer / 阶段评审）输出课程材料

800+ 页讲义

Yiqing Xu & Yang Stanford

Claude Code + Skills + ivDiag 自动复现 92 篇 IV 论文

87% 端到端 / 100% 条件复现

Aniket Panjwani Northwestern PhD

和 3 位 AP 配对编程；VoxDev 经济学讲座现场 750 人

3 位 AP 共 8 篇 Top 5 / R&R

Kai Li 等 UBC / Iowa / 港中深

用 GenAI 从财报会议、分析师报告里抽取企业文化的因果知识图谱

***Dissecting Corporate Culture*, 2025**

Karpathy 前 OpenAI / Tesla AI

Autoresearch：agent 自动跑实验、改代码、提交/回滚；自主发现架构改进

2 天跑 700 次实验

用 LLM 造变量：从文本到回归



已经发出来的例子

- **企业文化**：Li–Mai–Shen–Yan（RFS 2021）用 20.9 万份财报电话会，给创新、诚信、质量、尊重、团队五个维度打分；后续 *Dissecting Corporate Culture* 改用生成式模型，从同类文本里读出分析师眼中的企业文化
- **货币政策立场**：Hansen–Kazinnik 让 GPT 给 FOMC 声明分类鹰鸽立场，准确度超过常用文本方法，给出的理由接近人工水平

验证就是这条流水线的 Gate

- 抽样人工复核，先对齐人和模型的判断标准
- 关键字段换一个模型再抽一遍，对不上的人工裁决
- 金融文本的幻觉已有专门研究在测量——别跳过验证直接进回归

论文出处与验证清单见附录 A1

把 LLM 当被试：Homo Silicus

Horton（NBER w31122）做了什么

用 GPT 重跑三个经典实验：Charness–Rabin 的独裁者博弈、Kahneman 等人的公平判断、Samuelson–Zeckhauser 的现状偏好。结果与原实验大体一致；出现差异的地方，反而提示了新的研究问题。全部实验跑下来约 **50** 美元。

对研究者的用处

正式实验之前先在模拟被试上预跑：检查指导语、找设计漏洞、扫一遍参数，再决定哪个处理组值得花钱真做。

边界

- 对代表性和行为保真度已有系统评估提出质疑，涉及敏感社会类别时尤其要小心
- 只能当 pilot，不能替代人类被试

同一条线上的延伸

用带人设的 LLM 群体模拟宏观经济活动（EconAgent），或合成专业预测者做宏观情景预测——都还在方法检验阶段。

论文出处与质疑文献见附录 A2

我自己的 Auto Research 实验：复刻 APE 式自动研究

实验目标

不是证明 AI 可以自动做出好论文，而是测试：

- AI 能不能从数据出发，自动形成实证论文草稿
- 哪些环节可以交给 agent 执行
- 哪些环节必须用 workflow 和人工判断约束

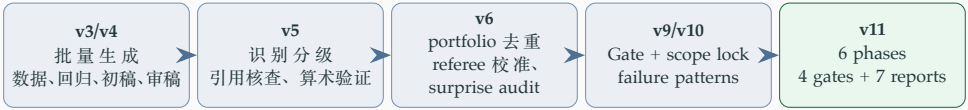
基本立场

这是一组 workflow 实验，不是正式论文成果展示。重点看流程、失败和改进。

实验对象

- AI API 平台：OpenRouter 使用面板、模型进入和退出
- 开源生态：HuggingFace、GitHub LLM repos
- AI 产业结构：Epoch AI 模型、GPU 集群、数据中心
- 科学扩散：OpenAlex、arXiv、ML 跨学科扩散
- 开发者采用：Stack Overflow Developer Survey

做了几轮：从“生成论文”到“约束论文”



早期问题

- 能写出很多“像论文”的文本
- 但容易过度主张、引用不稳、识别不清
- 不同论文之间容易相似

后期变化

- 先锁定对象、数据和 claim ceiling
- major critique 必须触发行动
- 允许 downgrade、merge、abandon

结果怎么样：workflow 能改进草稿，但不能制造好研究

阶段	主要尝试	结果和限制
batch 1–2	大量自动生成论文，加入基础 referee 和评分	产量高，但中位质量偏低；常见问题是 AI 味、引用、识别和主张边界
batch 4	两篇 AI API 市场论文，多轮 referee 后修订	内部评分从约 54–56 提升到约 67；说明迭代有效，但仍主要是 L1/L2 描述
batch 5–7	OpenRouter、HuggingFace、Epoch AI、OpenAlex、Stack Overflow 等多数据源	结构更完整，开始有 gate scores、claim ceiling、portfolio assessment 和 failure patterns
v11	GitHub LLM repos 单篇完整跑通	可达到 30 页、7000 字、6 表 4 图和 validator pass；但最终仍是 L2，不是因果识别

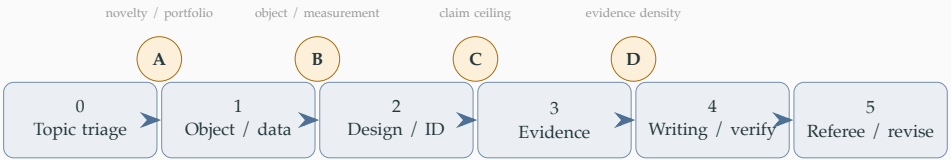
结论

workflow 可以稳定提升草稿完整性、复现性和边界控制；但研究问题、识别策略、解释和贡献判断仍然不能自动化。

共性失败模式：为什么全自动研究不现实

自动研究常见问题	workflow 里的对应约束
题目和已有论文太像，形成 salami risk	Topic gate：先做 portfolio neighbor 检查
平台数据被写成“整个市场”的结论	Scope lock：只写 observed platform / snapshot
proxy 被过度解释成 adoption、quality 或 welfare	Measurement gate：变量只能支持对应 claim
L1/L2 证据被写成 L3 因果语言	Identification level：标题、摘要、结论一起降级
referee 发现问题，但只写进 limitation	Major critique 必须触发新证据、改写或停止

最后一轮更成熟的范式：v11 workflow



7 个核心报告

topic triage；object / measurement / data；design / identification；analysis evidence；claims verification；referee revision；completion report

4 种 gate 结果

PASS 继续；**REVISE** 修；**REROUTE** 降级/合并；**STOP** 停止等待新数据或人工决策。

从 Auto Research 得到的 workflow 设计原则

可以自动化的部分

- 数据清理、合并、描述统计和表图初稿
- 引文、算术、路径、复现性检查
- 多版本草稿、审稿意见矩阵和修订记录
- 对已有 claim 做范围、证据和写作一致性检查

不能自动化的部分

- 研究问题是否值得做
- 经济对象和识别策略是否成立
- 结果是否真的有解释价值
- 什么时候应该降级、合并或放弃

最后得到的经验

成熟 workflow 的价值不在于让 AI 写得更多，而在于让 AI 的执行性工作被限制在可检查、可追溯、可回退的范围内。

三、工作流的解剖与设计

传统研究 vs. 工作流辅助研究



工作流的 6 个组成部分（解剖视角）

(1) **任务拆解** — 大目标拆成能一步步检查的小步骤

(2) **工具配置** — 让 AI 去算、搜、读、写

(3) **角色分离** — 写代码的人不能同时当审稿人

(4) **Gate** — 写清楚什么算通过、什么算退回

(5) **人工检查点** — 高风险的节点必须有人拍板

(6) **日志 / 版本 / 记忆** — 过程留得住、下次还能用

重点是把工作拆成**看得见、能查、能退回的步骤**，不是多写几个 prompt。

示例：一个 DID 项目的项目规则

研究问题：2024 年中国新能源汽车购置税减免政策如何影响消费者购车决策？

项目规则（WORKFLOW.md）

- DID 结果必须报告平行趋势检验
- 未经确认不得使用 `drop if`
- 所有表格必须使用三线表格式
- 标准误必须在处理单元层面聚类
- 引文必须能通过 DOI 核验

示例：同一项目的任务流水线



● 人工主导　● **AI 执行 + 人工审批**　● **AI 自主**　• 圆点数量 = 风险强度

现场演示：有 vs 没有 WORKFLOW.md

同一个 prompt：”跑基线 *DID* 回归”，AI 的两种默认行为：

没有 WORKFLOW.md

- 直接 `drop if` 删极端值
- 不报告平行趋势检验
- SE 用默认 OLS
- 表格用 `esttab` 默认格式

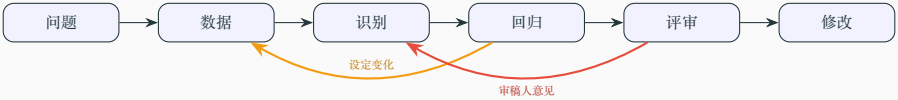
有 WORKFLOW.md

- 主动改用 `winsorize`
- 主动跑并报告平行趋势
- SE 在 city_year 层面 cluster
- 输出三线表格式

工作流不是线性的

流水线图看起来是一条直线往前，但真实研究一定会来回折返。好的工作流要做到三件事，才扛得住反复迭代：

1. 让每一步都能独立重跑
2. 保留中间产物，这样就能从任意节点重新开始
3. 记录改了什么、为什么改



六步设计框架（装配视角，1/3）：目标、拆解、工具



先定目标，再拆任务、配工具、留人工节点、设 Gate，最后写进规则和日志。

步骤 1：先定义目标与边界

最终产出是什么？哪些环节可以交给 AI？哪些错误不可接受？

步骤 2：把任务拆成能逐步检查的小步

每一步都写清：输入是什么、输出是什么、怎么检查。

步骤 3：为每一步配置工具、角色与检查方式

子任务	工具	执行	审核
T2 数据清洗	pandas + FRED API	数据处理	数据复核
T5 回归分析	R fixest / statsmodels	回归执行	方法复核

六步设计框架（装配视角，2/3）：人工节点与 Gate

步骤 4：决定人工该介入哪里

- **人工主导** — 因果识别、问题选择
- **AI 执行 + 人工批准** — 数据合并、标准误设定
- **AI 自主** — 描述统计、格式整理

步骤 5：设置 Gate

`drop if` → 阻断　　no DOI → 返修　　fabricated data → 失败

现场演示：Gate 触发与回退

你

帮我跑基线 DID，把极端值排掉。

Gate

WORKFLOW.md 第 2 条：未经确认不得使用 `drop if`。请确认是否修改这一规则，或改用 `winsorize`?

你

改用 winsorize at 1/99。

Claude Code

已用 winsorize 1/99；这次决定已写入 `MEMORY.md`。下次类似情况会主动询问。

六步设计框架（装配视角，3/3）：规则、记忆与从小做起

步骤 6：把规则写进项目文件

```
project/
├── WORKFLOW.md （目标、阈值）
├── rules/
│   ├── latex.md
│   └── data.md
└── MEMORY.md （常见坑、修复）
```

适合起步的任务

- 描述性统计表
- 回归结果输出
- 文献结构梳理
- 审稿回复初稿

不适合作为起点

- 全自动选题
- 无人审批的因果识别
- 从零到投稿的全流程

四、风险、边界与研究者角色

先分清：workflow 控制什么，研究者判断什么

工作流可以降低

- ✓ 数据幻觉
- ✓ 代码错误
- ✓ 格式错误
- ✓ 缺少测试

数据层

方法层

工作流不能替代

- × 因果识别
- × 研究问题的重要性
- × 过度外推
- × 伦理风险

解释层

价值层

workflow 能压住低级错误，研究者的注意力留下来做识别、解释、研究设计这几件事。

把人工检查点放在哪里



可以自动化更多的节点

- 文献抽取、代码执行、表格排版
- 格式检查、引用检查、单元测试
- 失败日志、版本记录、复现包整理

必须保留人工判断的节点

- 研究问题是否重要
- 识别策略是否可信
- 结论是否越过证据边界

设计 workflow 先做一件事：标出哪几步是**要人来拍板的**，再围绕它们配工具和 Gate。

学术规范：披露、隐私与复现

披露

论文里说清 AI 在哪一步、什么模型、有没有人类复核。

- 方法 / 数据 / 致谢都要写
- 注明模型 + 版本号
- AEA、JF、Nature 已纳入要求

隐私

敏感信息不进云端 prompt，先按敏感度分级。

- DUA / IRB / 企业数据用本地模型（Qwen、Llama）
- 公开数据也先去标识
- prompt 自己也是要归档的日志

复现

AI 介入的步骤都要能重跑，不只是代码可复现。

- 完整 prompt + response 一起存
- 锁模型版本：`claude-sonnet-4.5-20251022`
- 报告同一 prompt 多次运行的差异

4 个常见失败和怎么修

1. AI 自测自审

现象：它写完代码自己说"测过了，没问题"。

修法：critic / fixer 拆开；让另一个独立 agent 跑测试。

2. `drop if` 改名规避

现象：你禁了 `drop if`，它改用 `keep if`。

修法：Gate 检查任何过滤命令，不只看名字。

3. 复述但不执行

现象：它说"好的，我会按 WORKFLOW.md 来"，然后没读。

修法：让它先输出 WORKFLOW.md 摘要再继续。

4. 上下文丢失

现象：跨对话它不记得上一次定的规格。

修法：重要决定写进 `MEMORY.md`，下次先读。

Paper Factory（以及任何 workflow）做不到什么

不替代问题选择

Paper Factory 的 `launch_agents.sh new <name> "<question>"` 里，`<question>` 是用户必填字段。流水线第一步是 deep research，不是"找问题"。

输入的问题不够有意思，流水线只会把这个"不够有意思"包装得更加工整。

不替代领域判断

agent 切"价格 tier"作为分析切片，是因为 brief 里就这么写了。

它不"懂"产业组织、也不"懂"AI 市场，只是把已经会的方法套到给定的数据上，做得很整齐。

"为什么这是个好切片"那一层判断，没有自动化。

不替代外部效度

13 周的 OpenRouter 数据只能讲 OpenRouter 这个平台上的事。

agent 自己在 limitations 写得很清楚：

"...patterns on a specific marketplace; we make no claims about model use elsewhere."

回到开篇的 80 / 20

Paper Factory 自动化的，是研究里执行性的那 **80%**——清数据、跑代码、排版、改稿、对引文。

而剩下 **20%** 的判断（问题选什么、贡献怎么定位、识别策略合不合理、伦理怎么取舍）——一行没有自动化。

五、补充：常用工具与现场练习

Markdown：AI workflow 里的默认文件格式

Markdown 是用纯文本写带格式的文档的语法。.md 文件用记事本就能打开,Cursor、Claude Code 这些 AI 工具默认就读.md。

常用语法

`# 一级标题`	井号数 = 几级标题
`** 粗体 **`	两个星号包起来
`* 斜体 *`	单星号
`- 列表项`	短横加空格
`1. 编号项`	数字加点
`[文字](URL)`	链接
`` `代码 ` ``	反引号包起来

为什么 AI workflow 总用.md

- 纯文本——AI 直接读, 人也能直接看
- `git diff` 一眼看清改了哪个字
- 各种 AI 工具默认就用.md 写规则和文档

研究项目里常见的.md 文件

`README.md`	项目说明、目录结构、怎么跑
`WORKFLOW.md`	项目规则,AI 必须遵守什么
`MEMORY.md`	跨对话记下的决定
`SKILL.md`	一个 Skill 的入口文件
`CLAUDE.md`	Claude Code 项目级指令
笔记 / 文献卡片	想法、待办、卡片

打开方式

文本编辑器、VS Code / Cursor、Typora、Obsidian——任何能开纯文本的程序都能用。

git：给项目文件夹拍快照



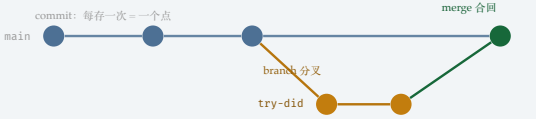
看

`git status`	改了啥、存了没
`git log`	所有快照清单
`git diff`	改了哪几行

撤销与试错

`git restore 文件`	撤掉还没提交的改动
`git reset --hard`	整个项目回到某张快照
`git checkout -b`	开分支单独试错

多人 / 多版本：分支与合并的逻辑



分支上改完 → 发 **Pull Request** 请求合回 `main`，他人审阅通过再 merge

为什么 AI 要配 git

撤回 AI 的修改

AI 改文件又快又多——一次任务可能动十几个文件。没有 git, 错了不知道怎么改回去; 有 git, 一行 `git reset` 就回到上次干净的状态。

让 AI 在支线上试错

重要改动先开一条支线 (`git checkout -b try-did`), 让 AI 在支线上改; 主线保持上个干净版本, 试错都不动它, 测好了再 `git merge` 回主线。

给 AI 一份项目的过去

把 `git log` 和 `git diff` 喂给它, 它就拿到了完整的项目时间线——“上次这段为什么改成这样”立刻能查。

推到 GitHub: 云端备份 + 学术披露

`git push` 一下整个仓库上 GitHub。private 只自己看、public 让人复现都行; 关键文件什么时候改、改了什么、谁改的, 都查得到。跨机器同步, 学术披露也比单纯写文字可信。

我做的三个研究工具

Research Planner

给“不确定何时能动”的任务排期。某阶段被 blocked, 就把这一块挪到时间轴外, 等条件回来再接回主线。

普通日历假设每件事都有起止时间; 研究里“等审稿回信、等数据授权、等合作者”反而是日常。

Lit Panel

让 AI 按预设的 workflow 读一批文献, 把它们之间的关系 (引用、概念、方法的共用) 抽出来, 拼成一张可视图。

一篇一篇读看不出格局;workflow + 图谱让你看一批文献的整体结构。

PaperReviewer

文档丢进去,AI 先给一份润色清单; 然后分批逐段改语言, 每段过完再继续。

一次让 AI 改全文容易跑偏、丢上下文; 分批跑, 每段都能人工 gate。

我做的一个 MBA 数据探索教学网站

https://www.sufe-mkt-analysis.online

放开: 工具、数据、提问

- 真实营销数据 + 在线分析环境
- 学生自由提问、自由跑模型
- AI 在旁, 帮跑代码、读报错、解释输出

克制: 不发结论

- 不预设“正确答案”
- 同一道题, 不同学生可以走出不同分析路径
- 课上讨论方法和取舍, 不背诵结果

跟这堂课的主线对得上: 执行交给工具, 判断留给做研究的人——这里就是学生。

参考资料

工作流案例与指南

- Anton Korinek, *AI Agents for Economic Research*（NBER w34202）：`https://www.nber.org/papers/w34202`，配套站点 `https://GenAIforEcon.org`
- Pedro H. C. Sant'Anna, *Claude Code Academic Workflow*: `https://psantanna.com/claude-code-my-workflow/`
- Project APE（Social Catalyst Lab, Zurich）：`https://ape.socialcatalystlab.org/`
- Paul Goldsmith-Pinkham, *A Causal Affair*（Substack，含 AI 辅助经济学工作流札记）：`https://paulgp.substack.com`
- Anthropic, *Claude Code* 官方文档: `https://docs.claude.com/en/docs/claude-code`

工具入口

- Cursor: `https://cursor.com`
- Trae: `https://trae.ai`
- Claude Code: `https://claude.com/claude-code`

附录：给读者的延伸材料

附录 A1　用 LLM 造变量：论文与验证清单

代表论文

- Li, Mai, Shen, Yan, *Measuring Corporate Culture Using Machine Learning*, RFS 2021：词嵌入 + 20.9 万份财报电话会转录稿，五个文化维度（创新 / 诚信 / 质量 / 尊重 / 团队），6.27 万个 firm-year（2001–2018）；文化与经营效率、风险承担、盈余管理、并购等结果相关。
  `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3256608`
- Li, Mai, Shen, Yang, Zhang, *Dissecting Corporate Culture Using Generative AI*：改用 GPT / PaLM 这类生成式模型，从财报电话会里抽取分析师视角的企业文化信号（HKUST 会计系工作论文，按标题可检索到全文）。
- Hansen, Kazinnik, *Can ChatGPT Decipher Fedspeak?*：GPT 给 FOMC 声明分类政策立场，准确度超过常用文本方法；GPT-4 给出的解释接近人工推理，还能按 Romer–Romer 叙事法识别宏观冲击。
  `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4399406`

验证清单（流水线里的 Gate）

抽样人工复核、对齐人机判断标准 → 关键字段双模型交叉、不一致人工裁决 → 报告与人工标注的一致率。金融文本幻觉的实证测量见 *Deficiency of LLMs in Finance*：`https://arxiv.org/abs/2311.15548`

附录 A2　模拟被试：论文与争议

出发点与复现

- Horton, *LLMs as Simulated Economic Agents: What Can We Learn from Homo Silicus?*（NBER w31122）：把 LLM 当作隐含的人类计算模型，设定禀赋、信息、偏好后重跑 Charness–Rabin（2002）、Kahneman 等（1986）、Samuelson–Zeckhauser（1988）的经典实验，结果与原实验大体一致，总成本约 50 美元。
  https://www.nber.org/papers/w31122
- 后续工作用 GPT-4o、Claude、Llama 等多个模型重复独裁者博弈，跨模型比较结果。

延伸：让一群 LLM 模拟经济活动

EconAgent：带人设与记忆的 LLM 群体模拟宏观经济（https://arxiv.org/abs/2310.10436）；用合成 LLM 人物做宏观情景预测（https://arxiv.org/abs/2511.02458）。

质疑

对合成被试的系统评估指出代表性与行为保真度的问题，涉及敏感社会类别时偏差更明显；结论：可以当 pilot 工具，不能直接替代人类样本。https://arxiv.org/abs/2509.26080

附录 A3　多智能体与 MCP

Anthropic 的多智能体研究系统

领队–子 agent（orchestrator–worker）结构：领队分析问题、定策略，并行派出 3–5 个子 agent，每个有独立上下文窗口和工具、互相不可见；领队汇总后另跑一遍引文核对。内部评测：Opus 4 领队 + Sonnet 4 子 agent，比单用 Opus 4 高 90.2%；token 用量约为普通对话 15 倍。

`https://www.anthropic.com/engineering/multi-agent-research-system`

MCP：把 AI 接上你的数据源

MCP（Model Context Protocol）是连接 AI 与外部工具 / 数据的开放协议，2024 年底由 Anthropic 开源，主要厂商都已接入；到 2026 年 3 月，公开 MCP server 超过 1 万个，SDK 月下载约 9700 万次。

对研究者直接有用的几类 server：**Zotero**（让 AI 查、引、总结你的文献库）、**学术检索**（arXiv / Semantic Scholar / Google Scholar 跨库搜索）、**PostgreSQL**（在数据库上安全跑 SQL）。

`https://www.digitalapplied.com/blog/mcp-adoption-statistics-2026-model-context-protocol`

附录 A4　给经济学家的 agent 教程：Korinek（NBER w34202）

这篇在讲什么

目标是把 AI agent 讲明白，让**没有编程背景的经济学家**也能搭自己的研究 agent。论文配完整代码示例，覆盖四类任务：跨来源自动文献综述、写和调计量代码、取数与分析、协调多步研究流程。

用什么搭

用自然语言驱动的编程（vibe coding）加上 LangGraph 这类 agent 框架，几分钟能搭出一个能用的研究助手；从最初的调研到最后的分析都有对应示例。

入口

论文：`https://www.nber.org/papers/w34202`（2025 年 9 月）
配套资源站：`https://GenAIforEcon.org`
同方向综述：*Agentic Workflows for Economic Research*，`https://arxiv.org/abs/2504.09736`

把执行性工作交给 AI

把研究者的判断力，留给识别、解释和研究设计。

谢 谢

陶学臻 · 上海财经大学商学院

微信公众号：**AI 经研派**