Accelerating Science with AI and Agentic Workflows

Santiago Balseiro

Paul Dütting

Renato Paes Leme

Columbia University, May 2026

Google Research

Plan for this tutorial

- Part 1: Directly talk to the AI chatbot
  - Automating the research workflow
  - Generating novel insights
- Part 2: Agentic Workflows
  - Intro to AI agents
  - Agent-driven research workflows
  - Data science applications
  - Theorem proving

This Tutorial:
Goals and Outline

Goal 1: How can AI and LLMs help you?

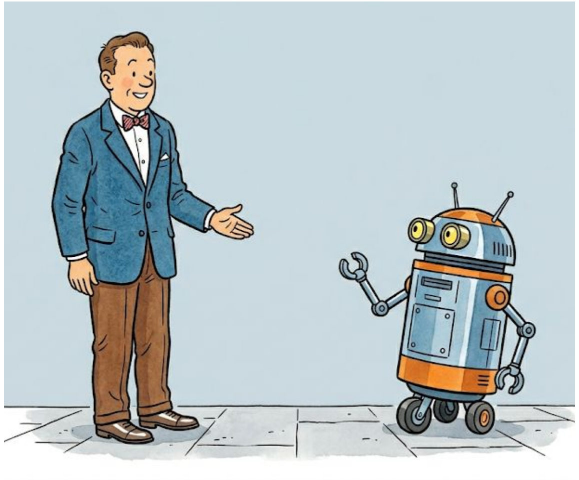
Automating the Research Workflow

enabling more efficient task execution

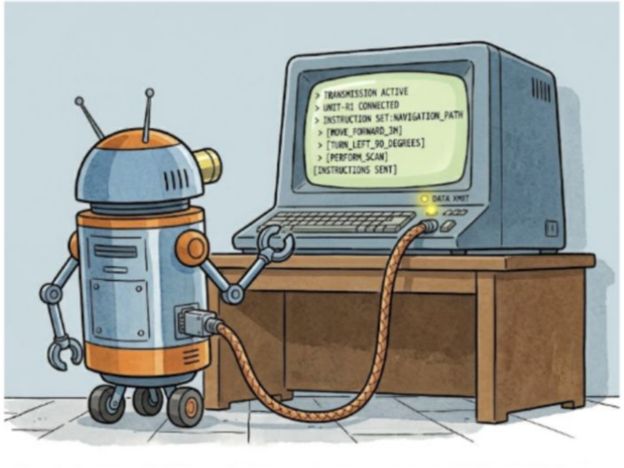
Generating Novel Insights

pushing the boundaries of knowledge

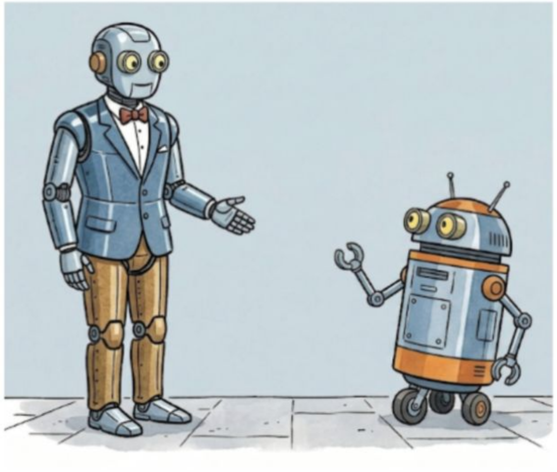
Goal 2: Automating the use of AI



Directly talk to the AI



Write programs that call the AI



Get AI to direct other AIs

Important Disclaimers

Rapidly evolving technology

AI is evolving very fast, and the current tutorial reflects the technology right now. In a few months, the tools available could be very different.

Personal experience

Tutorial is based on our own experienced trying to do Research using AI and reflects our subfield (Theoretical Computer Science, Algorithmic Game Theory, Operations Research).

Our intuition and understanding is also evolving as we go.

Practical / Example-Driven

We will try to give many ideas of prompts and example of what can be done.

Gemini Examples

Examples will be based on Google technologies (Gemini, Gemini CLI, Colab, Antigravity,...) but the tutorial should apply to any similar technology out there.

Things to keep in mind

- **Try again if you had a bad experience in the past:** Tools have evolved remarkably over the past six months, resulting in significantly better performance.

- **Own it and declare it:** Most academic conferences and journals allow the use of AI, provided its application is explicitly declared.

- **Be aware of the pleasing bias:** LLMs are trained to be helpful, which means they will do their best to provide an answer even if it involves producing a highly plausible, yet entirely flawed, proof.

No magic bullet

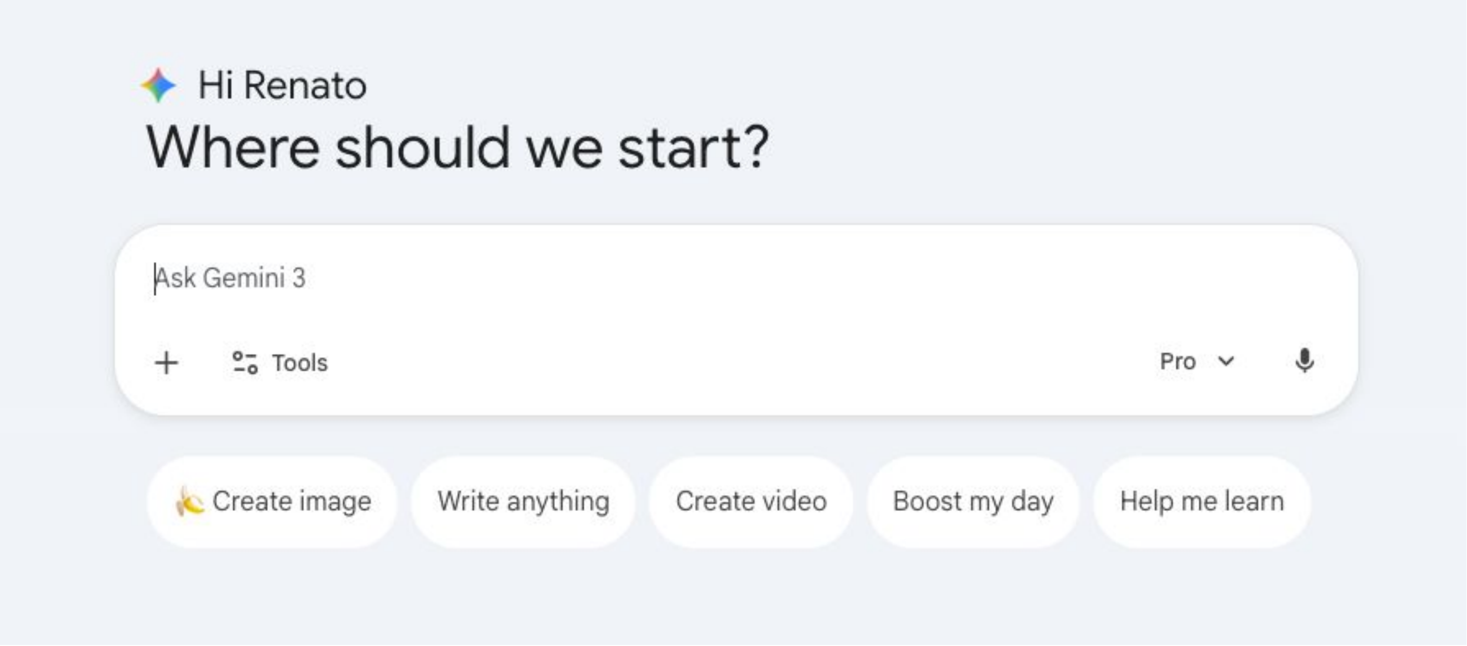
- Research still requires lot of effort (even with AI)
  - Asking the right questions & providing the right intuition
  - Ensuring scientific and mathematical rigor
  - Using the tools efficiently

- … but you may end up spending time doing other things
  - Some formerly labor-intensive activities become effortless
  - You spend more time trying prompt variations, checking arguments, proof-reading, …

- **Long run**: With tedious tasks out of the way, we hope AI will help us focus on what matters and allow us to tackle more difficult tasks.

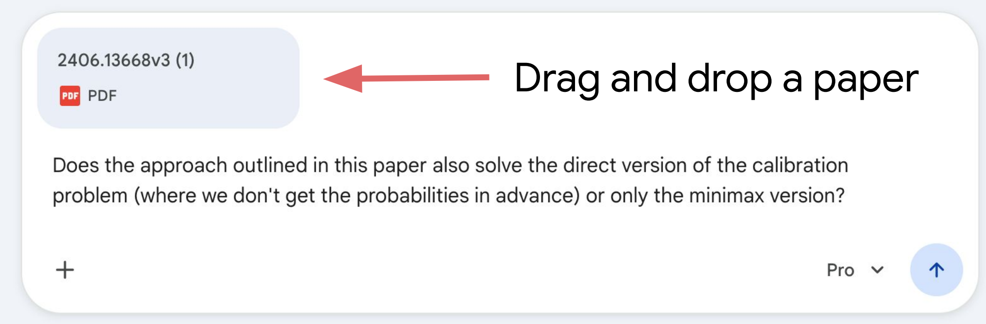
Part 1:
How can AI and LLMs help you?

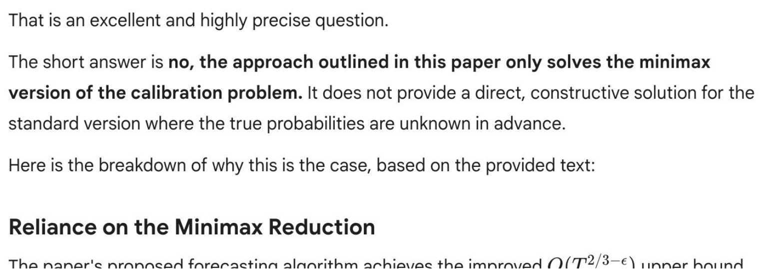
Getting started: http://gemini.google.com/



Part 1a: Automating the Research Workflow

Let’s read a paper by talking to it




That is an excellent and highly precise question.

The short answer is **no, the approach outlined in this paper only solves the minimax version of the calibration problem.** It does not provide a direct, constructive solution for the standard version where the true probabilities are unknown in advance.

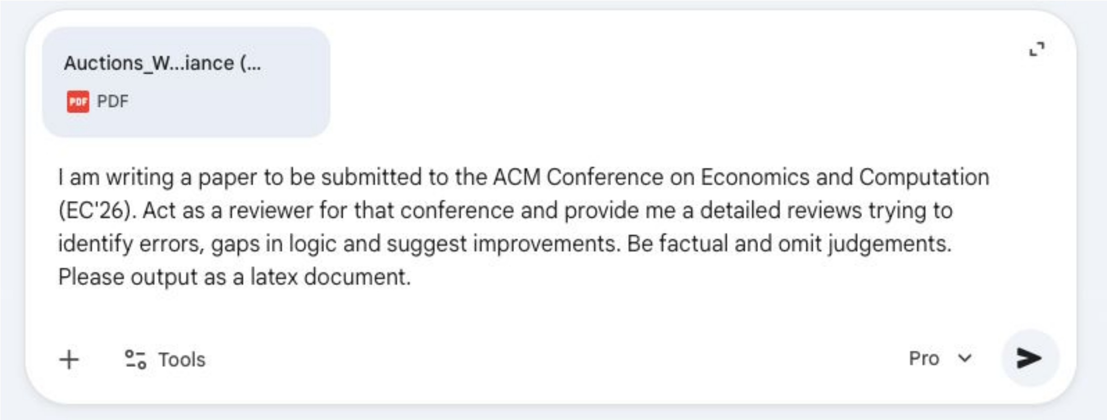
Here is the breakdown of why this is the case, based on the provided text:

Reliance on the Minimax Reduction

The paper's proposed forecasting algorithm achieves the improved $O(T^{2/3-\epsilon})$ upper bound

- Ask clarifying questions about the model
  - “How does it compare to the setting in paper X”
  - “Please write the full derivation of equation Y”
  - “Please explain what concentration inequality is used in Lemma Z”
  - …

Let’s review a paper



- Give as many details as possible
    - Which conference/journal you are submitting
    - What should reviewer focus on? (Grammar, math errors, gaps in logic, missing references, ...)
    - Maybe specify something about the reviewer (adversarial, unconvinced about your research agenda, looking for reasons to reject, ...)
    - It is useful to specify the output (latex document)

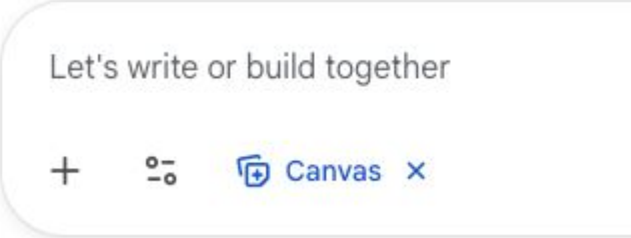
Here is a review of the paper "Auctions With Small Variance" formatted as a LaTeX document, suitable for the ACM EC'26 conference context.

Code snippet

```
\documentclass[11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage{geometry}
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{enumitem}

\geometry{a4paper, margin=1in}

\title{Review of "Revenue Variance Minimization: Beyond First Price Auctions"}
\author{ACM EC'26 Review Committee}
\date{}

\begin{document}

\maketitle

\section*{Summary of the Paper}

This paper investigates the minimization of revenue variance in auction settings

The paper's main contributions are:
\begin{enumerate}
    \item \textbf{Single-Item Interim IR:} The authors show that while WPB remair
    \item \textbf{Multi-Item Interim IR:} For symmetric regular environments, the
```

Copy and paste to Overleaf or your favorite LaTeX editor.

Or alternatively use Canvas:



please create the same review in canvas

Show thinking

I have converted the review into a LaTeX file that is ready for compilation or submission.

Review of Revenue Variance Minimization
Jan 30, 6:02 PM

Try again without Canvas

Let's write or build together

Review of Revenu... Code Preview Share

Review of "Revenue Variance Minimization: Beyond First Price Auctions"

ACM EC'26 Review Committee

January 30, 2026

Summary of the Paper

This paper investigates the minimization of revenue variance in auction settings with risk-neutral, unit-demand bidders and independent private values. The authors aim to generalize the classical result of Waehrer et al. (1998), which established the optimality of the First-Price (or Winner-Pays-Bid, WPB) auction for variance minimization under ex-post Individual Rationality (IR).

The paper's main contributions are:

1. **Single-Item Interim IR:** The authors show that while WPB remains optimal for i.i.d. valuations under interim IR, this optimality breaks down in asymmetric settings. They characterize the variance-minimizing payment rule for asymmetric cases using a decomposition technique and prove existence using fixed-point theorems in Banach spaces.
2. **Multi-Item Interim IR:** For symmetric regular environments, the authors construct a mechanism (derived via a differential equation) that achieves zero revenue variance while maintaining non-negative payments and interim IR.
3. **Multi-Item Ex-post IR:** The authors demonstrate via a counterexample that WPB is not variance-minimizing in multi-item settings (specifically 2 items, 3 bidders). However, they prove that the discriminatory WPB auction still yields lower variance than the uniform $(k+1)$-st price auction without requiring log-concavity assumptions.

General Comments

Examples of AI feedback

2. Numerical vs. Analytical Rigor in Section 5

While Sections 3 and 4 rely on rigorous analytical derivations (decomposition techniques, ODEs), Section 5 relies heavily on numerical examples (e.g., the counterexample in Section 5.1 and the discretized optimization in 5.2).

- In Section 5.1, the variance reduction is shown via numerical integration (≈ 0.26431 vs ≈ 0.2613). While the example is illustrative, an analytical bound or a local perturbation argument would strengthen the claim that WPB is strictly suboptimal.
- In Section 5.2, the optimal mechanism is approximated via discretization. Given Lemma 5.1 reduces the problem to functions of order statistics, is it possible to derive the Euler-Lagrange equations for the calculus of variations problem defined in [582-586]? Even if no closed form exists, characterizing the properties of the solution analytically would be superior to a purely numerical plot.

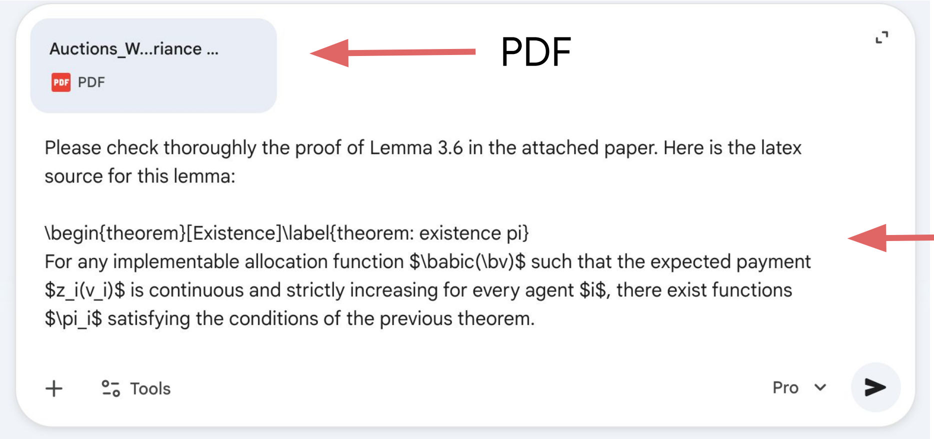
3. Theorem 3.6 (Existence via Fixed Point Theorems)

The proof of existence for the asymmetric variance-minimizing mechanism utilizes Schauder's fixed point theorem on the space of probability measures. While technically impressive, this machinery seems heavy for the problem at hand. Is the operator T a contraction under some metric? If so, the proof could be simplified, and potentially imply uniqueness or convergence of simple iterative algorithms. The authors should comment on the uniqueness of the solution $\pi_i(v_i)$, as Schauder does not guarantee it.

Minor Comments and Typos

- **General:** The citation style uses bracketed numbers [1], but the text often refers to "Waehrer et al. (1998)". Consistent formatting is recommended.
- **Page 13, Eq 306:** The text reads *"...evaluates to evaluates to..."*. Duplicated words.
- **Page 13, Eq 307 & Page 14, Eq 319:** The use of $\approx$ without error bounds or exact expressions in an analytical section is slightly jarring.
- **Page 16, Footnote 1:** The footnote regarding jumps and countable discontinuities appears disconnected from the main text flow or is formatted essentially as a sentence fragment at the bottom of the page.
- **Page 20, Step 3:** The integral substitution uses $\frac{d}{dx}[x(1-F(x))]$. Ensure the sign conventions in the subsequent integration by parts are clear, particularly the negative sign on $(n-k)\phi(x)$.
- **Page 25, Eq 551:** The definition of $b_1(v_i)$ appears to have formatting issues in the submitted PDF (e.g., *"Vis U $\leq$ 0.6"*). The conditional cases are hard to parse.
- **Page 27, Figure Captions:** Figures 3, 4, and 5 have redundant or unclear labels (e.g., *"#1 Highest Bidder"*, *"Highest Bidder"*). The axis labels are too small to be legible.
- **References:** Ensure all arXiv/SSRN preprints (e.g., Pekeč and Smilgins) are updated to the latest versions if published.

Keep in mind the “thinking budget”

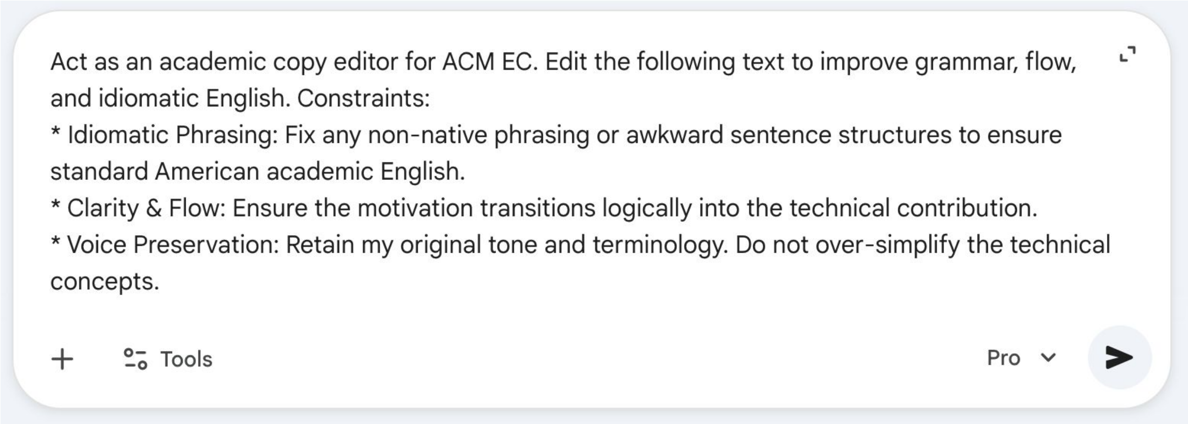


PDF

LaTeX source

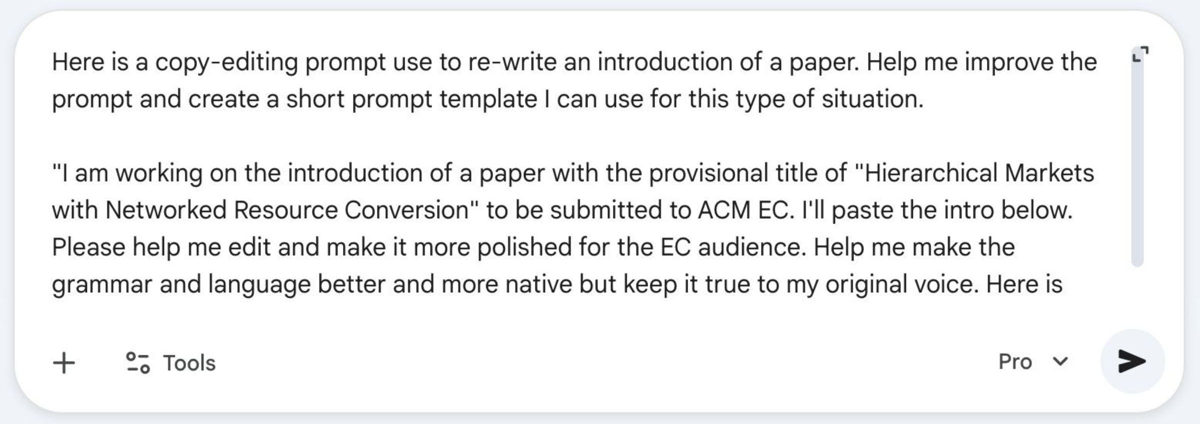
- The model has limited “attention” and time spent thinking
- It is useful to make the model focus on a specific part of the paper
- PDF is fine, but LaTex source is even better since there are no errors introduced by PDF parsing

Copy-editing



- Copy-editing works best with small section at a time
- Give the full context, conference, topics…
- Directions such as “keep my original voice”, “make it idiomatic”, …
- “Keep the sentence structure relatively similar to the original unless a change significantly improves clarity”

Prompt for prompting



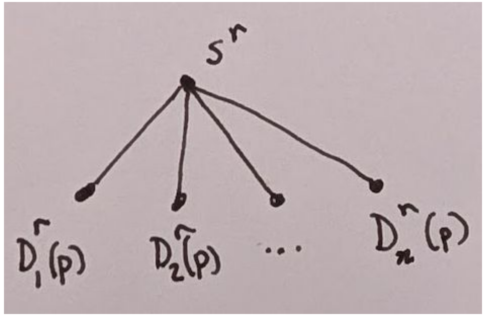
- If unsure what prompt to use, use Gemini itself to suggest the prompt.
- It is useful to keep a list of "prompt templates" that worked for you in the past.

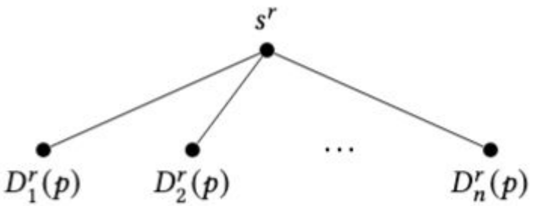
Let’s draw a diagram

Drag and drop a photo

please create a tikz version of the following image:

+ Tools Fast





Code snippet

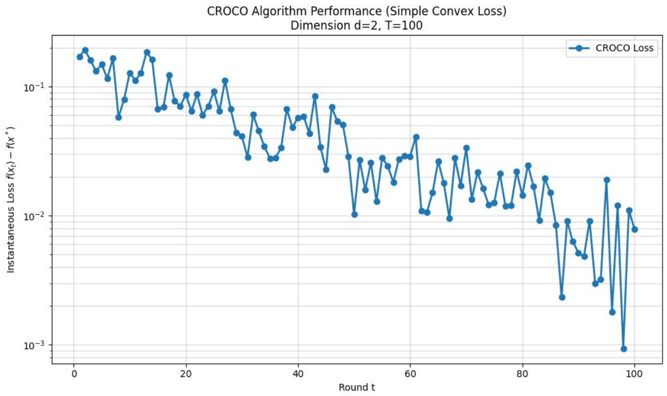
```
\documentclass[tikz,border=2pt,png]{standalone}
\usepackage{tikz}
\usetikzlibrary{trees}

\begin{document}

\begin{tikzpicture}[
    level distance=3cm,          % Vertical distance between levels
    sibling distance=2.5cm,      % Horizontal distance between nodes
    dot/.style={circle, fill=black, inner sep=2pt}, % Style for the black dots
    every label/.style={font=\small} % Font size for labels
  ]

  % Root node
  \node [dot, label=above:$S^n$] {}
    % First child
    child { node [dot, label=below:$D_1^r(p)$] {} }
    % Second child
    child { node [dot, label=below:$D_2^r(p)$] {} }
    % The "..." node. We use edge from parent[draw=none] to not draw a line to it
    child { node {\dots} edge from parent[draw=none] }
    % Last child
    child { node [dot, label=below:$D_x^n(p)$] {} };

\end{tikzpicture}

\end{document}
```

Implementing Algorithms

MOR___cor...ual_se...
PDF

Please provide an implementation of the Algorithm 1 "Log-Concave Density Algorithm for CROCO" in the attached paper in python. Use MALA to compute the centroid of a log-concave distribution. Also produce a plot showing the value of a loss function of a simple convex function as a function of the number of interations.

Canvas
Pro



Not what I expected but 90% there.

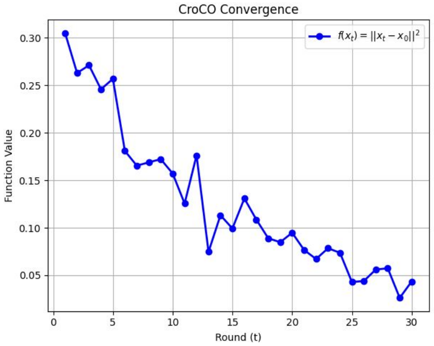
Corruption Robust Convex Optimization

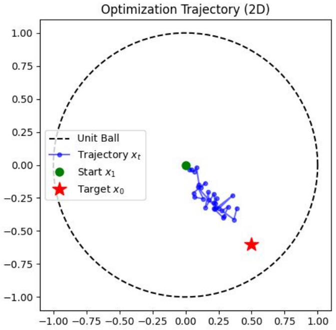
Export to Colab

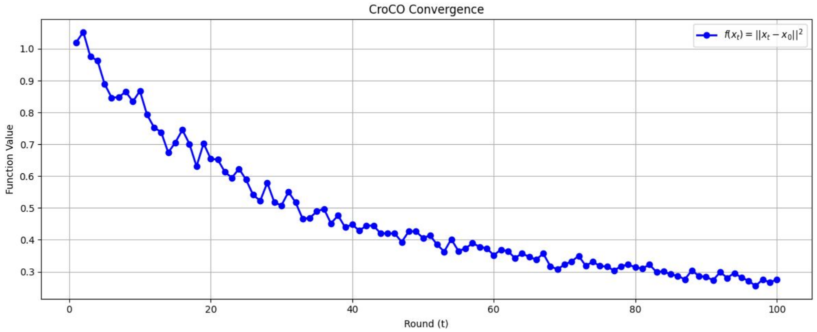
```python
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm

class LogConcaveDensityCROCO:
    def __init__(self, dim, L, D, gamma_scale=3.0):
        """
        Initializes the CROCO algorithm.

        Args:
            dim (int): Dimension of the problem.
            L (float): Lipschitz constant of the function.
            D (float): Diameter of the domain K.
            gamma_scale (float): Scaling factor for gamma. Paper suggests gamma = 1/(3LD).
        """
        self.dim = dim
        self.L = L
        self.D = D
        # Gamma setup as per Theorem 1: gamma = 1 / (3LD)
        self.gamma = 1.0 / (gamma_scale * L * D)

        # History of queries (centroids) and feedback (gradients)
        # x_history[i] corresponds to x_{i+1} in paper notation (0-indexed)
        self.x_history = []
        self.grad_history = []

        # MALA parameters
        self.mala_step_size = 0.05 / np.sqrt(dim)  # Heuristic scaling
        self.last_sample = np.zeros(dim) # Start chains from previous point (warm start)

    def log_density_unnormalized(self, x):
        """
        Computes the log of the unnormalized density mu_t(x).
        mu_t(x) = mu_1(x) * Product_{s=1}^{t-1} (1 - gamma * <grad_s, x - x_s>)

        Returns -inf if x is outside the domain (Unit Ball).
        """
        # Check domain constraint (Unit Ball)
        if np.linalg.norm(x) > 1.0:
            return -np.inf

        # Base density mu_1 is uniform (log constant), we ignore the constant.
        if not self.x_history:
```

Implementing Algorithms

- After a few rounds of interaction (improving prompts, manually modifying the code and the presentation):

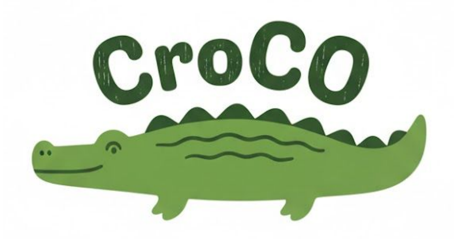






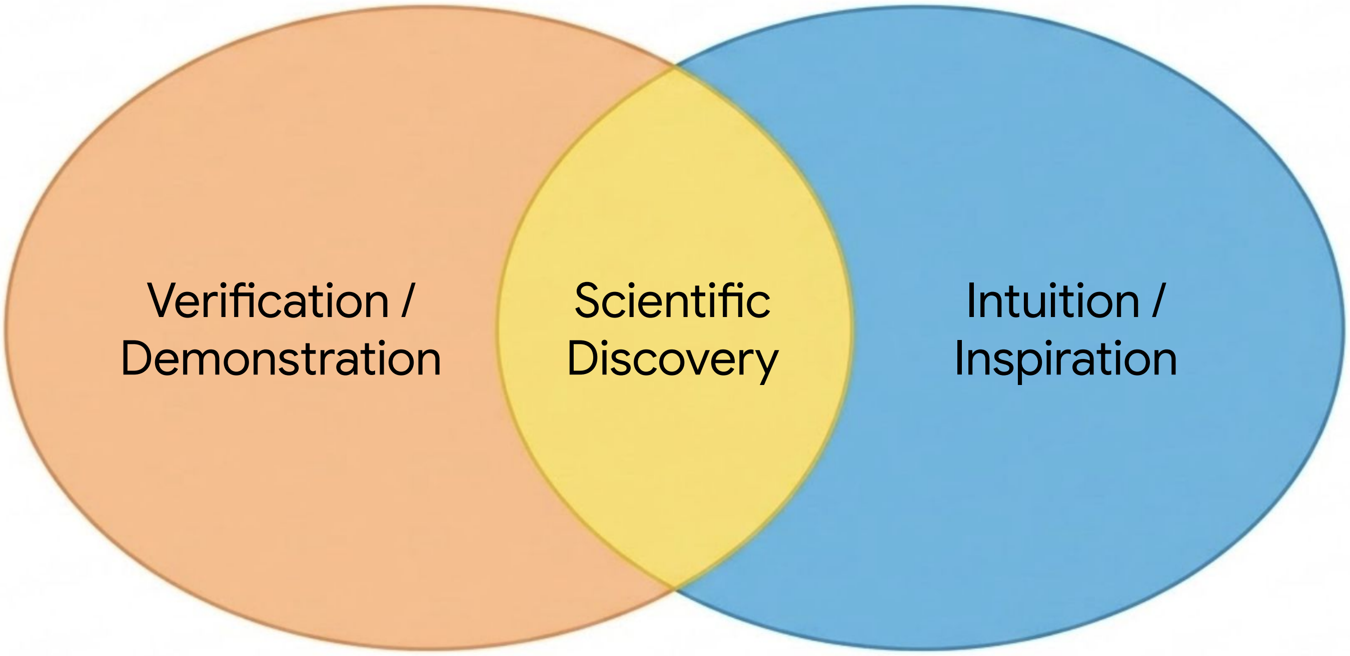
- Usually requires verifying the code, fixing mistakes, tweaking parameters, …
- Some of that is manual and some of that you can ask the model to do using follow-up prompts.

- **Bonus**: “Can you design a logo for the algorithm:”



Part 1b: Generating Novel Insights

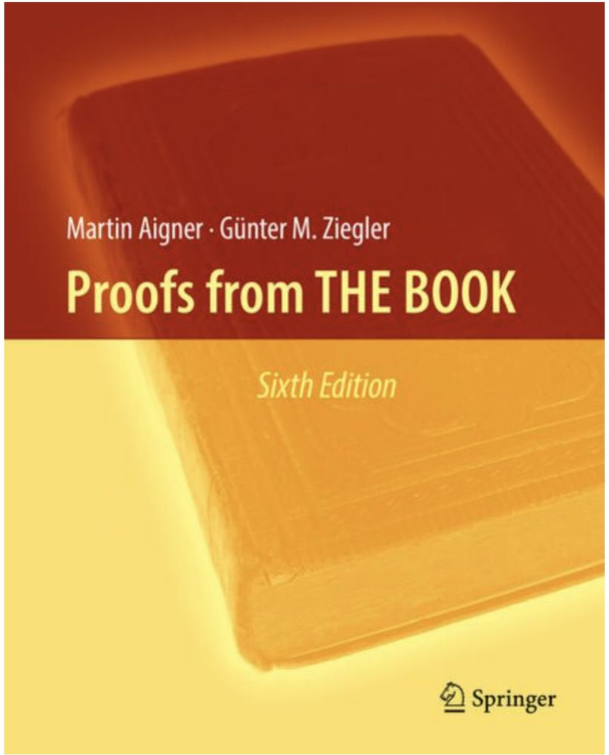
Scientific discovery



E.g. Henri Poincaré, “L’invention mathematique” (Paris, 1908)

Theorem proving

- Proving new results is as much art as it is science
- Requires mathematical rigor
- But also relies on creativity, intuition, mathematical taste, a sense of beauty, the ability to see hidden connections …



Two types of theorem proving

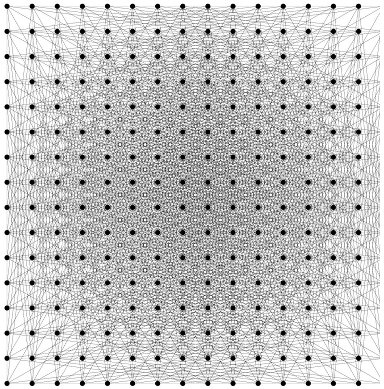
- **Type 1**: “I don’t know the proof of that statement but the right mathematician probably knows it.”

  This is the domain where AI tends to be more useful. However, guiding the AI to a rigorous proof may still require careful human steering.

- **Type 2**: It is truly an open problem for the research community.

  AI is beginning to make progress here, but this is significantly harder. Success typically requires combining human intuition with structured AI workflows.

Type 2 success stories



- Formal reasoning & verification
  - Olympiad-level formal mathematical reasoning with reinforcement learning (Nature)
- Structured Human-AI workflows
  - Semi-autonomous mathematics discovery with Gemini: A case study on the Erdős problems (arXiv) Also see Wiki
- Algorithmic search & evolutionary discovery
  - Mathematical exploration and discovery at scale (arXiv)
  - AlphaEvolve: A coding agent for scientific and algorithmic discovery (arXiv)
- Agentic loops & extended reasoning
  - Early science acceleration experiments with GPT-5 (arXiv)
  - Accelerating scientific research with Gemini: Case studies and common techniques (arXiv)
  - Autonomous mathematical discovery via test-time compute (blog post, proof, arXiv)

Techniques that are usually helpful

The AI as a smart PhD student with expertise across many fields: What advice / hints would you give a student working on this problem?

- Ask the AI to brainstorm possible proof techniques, or provide intuition on how a proof might go.
- Give as much detail as possible (what special cases are known, impossibilities, potentially some useful lemmas).
- Add related references to the context ("this may be related to Lemma ... in the PDF attached").

> I am working on the attached project on combinatorial contracts. I am trying to address a gap in one of the proofs. This is is Lemma lem:subset-approx, in the final step. Here the equality in the final step does not hold, but we hope that we can get rid of the constant. See the comment \pdc{...} in the source file. Can you help me develop ideas for how to approach this?

[this did not work]

> Let me propose a different direction. There is recent literature (e.g., Multi-Agent Combinatorial Contracts, SODA 2025) which studies a different contracting problem, but encounters a similar challenge. There the authors solve this differently: They decompose the solution in a constant number of small agents, and then partition the remaining agents into a small number of sets. This way they achieve two things at once: (a) ... and (b) ....

[this worked eventually]

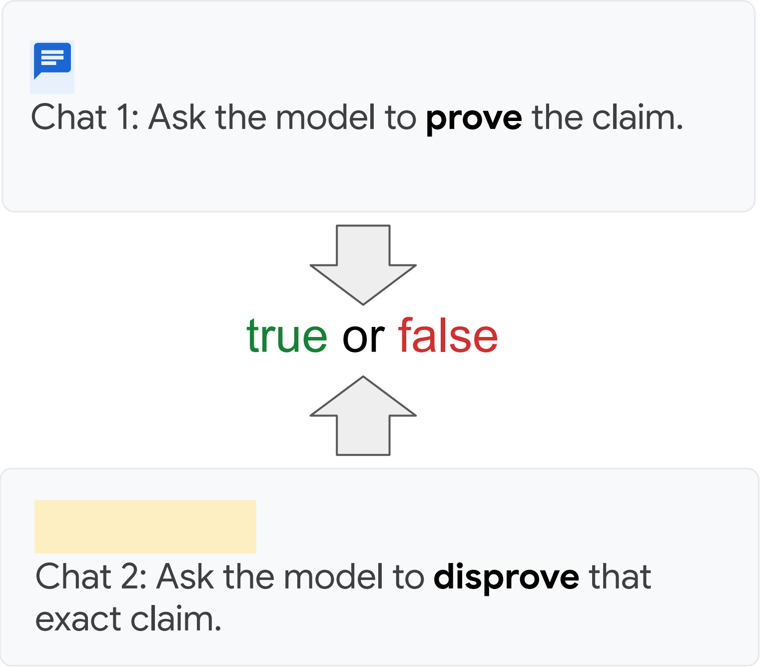
Techniques that are usually helpful

The primal-dual approach: If you ask an AI to "help prove" a statement, it may try to force a proof even if the statement itself is wrong.

To avoid this, consider running two separate chats in parallel:

- **Chat 1 ("The Primal"):** Ask the model to prove the claim.
- **Chat 2 ("The Dual"):** Ask the model to disprove that exact claim.

Even if neither of the two "succeed", it can be an effective tool to identify the mathematical core of the problem.

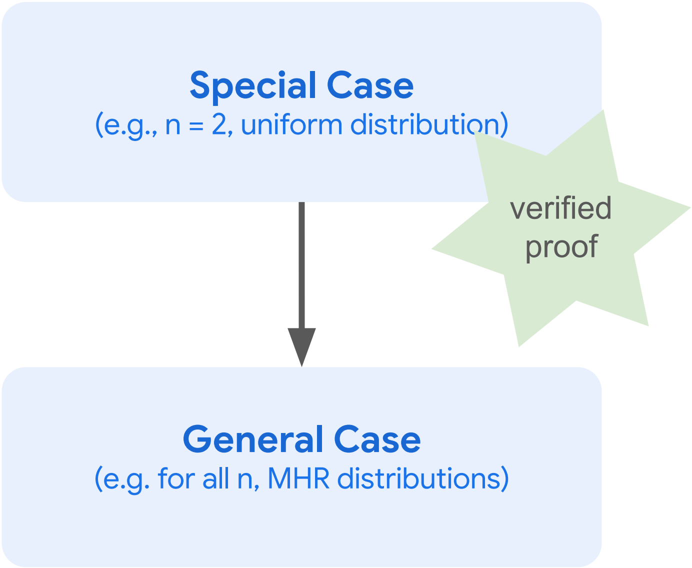


Techniques that are usually helpful

From specific to general: First prove a special case (e.g., n = 2 or assume the uniform distribution), either "by hand" or with the help of AI.

Iterate on the proof for the restricted case until you are fully satisfied with its logical soundness and rigor.

Finally, ask the AI to use your verified proof as a template to generalize to the broader case.



Techniques that are usually helpful

Create a "cheat sheet" (context injection): Ask the AI to create a summary of useful facts and properties about a mathematical object.

For example: "produce a Latex document with a collection of useful lemmas and properties of regular/MHR distributions"

You can then feed this "cheat sheet" back to the AI to tackle the problem you want to solve.

User • 12:02 PM

I want to create a "cheat sheet" with useful properties of regular distributions (in the sense of Myerson). 1. Please find a few important references from the recent literature in algorithmic mechanism design. 2. Scan these papers and potential sources they cite to identify some properties and useful lemmas of regular distributions. 3. Combine a 3-4 page long Latex document which states the definition of a regular distribution and lists the useful properties you identified. 4. Carefully review and possibly update this document. Check for consistent notation and correctness, and correct attribution of the results. 5. Return the revised Latex as a self-contained Latex document that I can download and compile.

1. Literature Review & Reference Selection

To build this document, I identified foundational literature and heavily cited mod algorithmic mechanism design that establish or extensively rely on regularity:

- **Myerson (1981):** Introduced the definitions of virtual valuation and regularity.
- **Bulow & Roberts (1989):** Translated Myerson's theory to "quantile space," establishing that regularity is mathematically equivalent to revenue curve concavity.
- **Bulow & Klemperer (1996):** Demonstrated that market expansion (adding a bidder) strictly supersedes optimal algorithm design (knowing the distribution to set the optimal reserve).
- **Chawla, Hartline, Malec, & Sivan (STOC 2010):** Leveraged regularity to reduce complex multi-parameter auctions to sequential posted pricing using prophet inequalities.
- **Dhangwatnotai, Roughgarden, & Yan (EC 2010 / JACM 2015):** Relied on the concavity of regular revenue curves to prove that a single random sample yields a $\frac{1}{2}$-approximation of optimal revenue.
- **Cole & Roughgarden (STOC 2014):** Formalized the "Pareto tail bound" of regular distributions to prove tight bounds on the sample complexity of empirical revenue maximization.

2. Extracted Properties & Lemmas

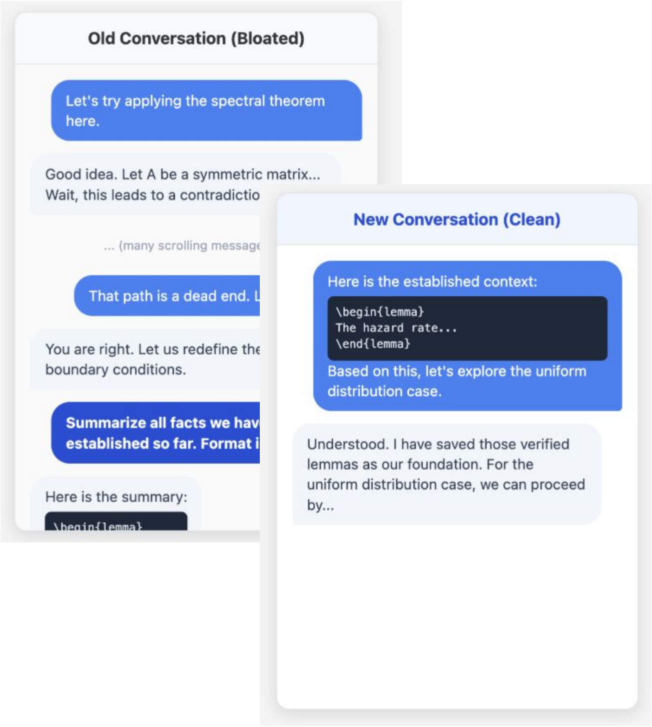
From scanning these sources, I identified several highly useful properties:

Techniques that are usually helpful

Resetting context (manage context window): Over long derivations, the model can get confused by past mistakes or tangential ideas; the model has "too much on its mind."

Ask the AI to summarize all verified progress and facts established so far. (Requesting this in LaTeX or Markdown format makes it easy to copy).

Open a completely new chat window, paste only this refined summary, and continue the work with a fresh context window.



Techniques that are usually helpful

Multi-step instructions (chain of thought prompting): If you ask for a final proof immediately, the AI often skips logical steps. Force the model to plan first.

E.g.: After an AI review, it's often useful to follow up and ask for a LaTeX formalization of the proposed fix:

> Can you help me address the bullet "\textbf{...}". How would you change the proof of this <Lemma/Proposition/Theorem> to formalize the argument that you make in the review? Proceed as follows:
>
> 1. Review the critique and the proposed fix.
> 2. Plan how to update the proof.
> 3. Write out a formal proof in Latex.
> 4. Very carefully check the proof you are proposing. If there are any issues fix them.
> 5. Return the LaTeX code for the new proof.

"The complete LaTeX code for the streamlined proof is provided above, and can safely overwrite the entire original proof."

Brilliant idea!

Other use cases

- “Can you check if an argument like this one exists in the literature?
- “I have this vague setup/approach in mind. Can you re-define it in terms of existing concepts in the literature?”

OK, attaching a draft of my rewrite of section 5.

The algo is more streamlined (fewer oracle calls, only one Seymour algo call, no explicit checks for 3-connectedness).

I discovered that the thing we're calling id-path maps to a concept called theta matroid.
A theta matroid is the P_1, P_2 P_3 structure where each pair is a circuit that we use to define 3-connectivity.

Summary & next up

- Chatbots increasingly prove to be very effective research assistants
  - For automating standard tasks (drawing tikz pictures,...)
  - For deriving novel insights (incl. proving theorems)
- Direct interaction with the chatbot gives us maximum control, but can also be exhausting
  - e.g. it requires constant, manual verification
  - e.g. it demands careful step-by-step steering
- From operator to orchestrator: Next we will discuss delegating execution loops to AI agents

Part 2: Agentic Research Workflows

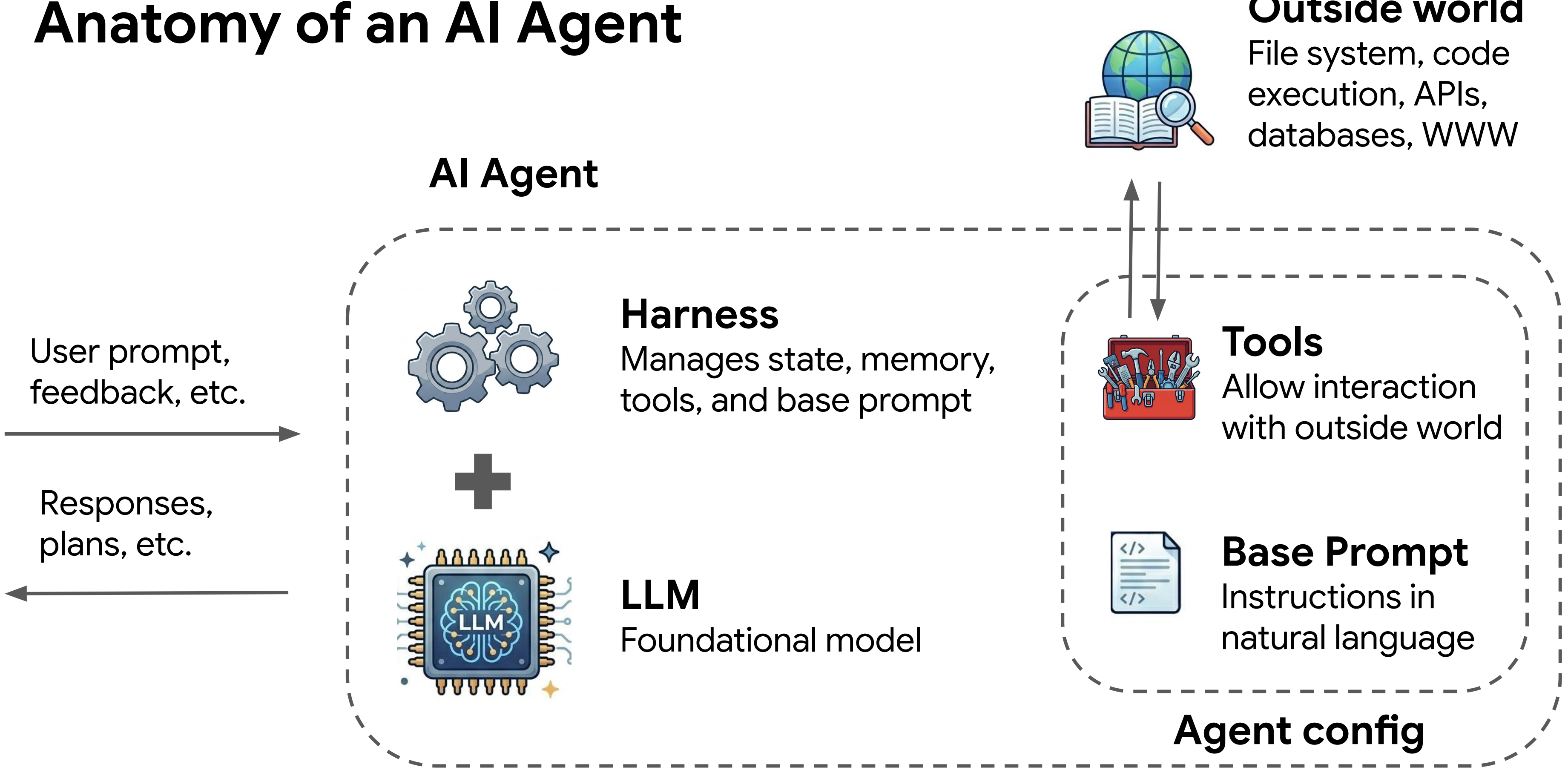
Outline

- Introduction to AI Agents
- Overview of the Antigravity app
- Editorial work
- Research and math
- Data analysis

Why AI Agents?

- From **Passive** to **Active**: Shift the paradigm from generative AI (responding to prompts) to agentic AI (autonomous goal-pursuit)
- An **AI agent** is a system capable of reasoning through complex objectives, breaking tasks into steps, interacting with its environment, and executing actions using external tools
- Impact of Agents:
    - Implement complex code (even a whole operating system)
    - Solve complex math problems
    - Identify vulnerabilities in critical infrastructure
    - Customer support and sales agents
    - And many more applications!

Anatomy of an AI Agent



Antigravity IDE (https://antigravity.google/)

- Integrated development environment (IDE) built on VS Code
- Native app originally designed for coding but can do a lot more
  - Can compile code, access to computer tools (browser, terminal)
  - Edit and compile Latex files
  - Agentic platform: agents can perform tasks on your behalf
- Probably 10-100x more powerful than using a chatbot alone
  - From copying and pasting to autonomous goal-pursuit
- Similar software provided by other companies too
- Login with a Gmail account

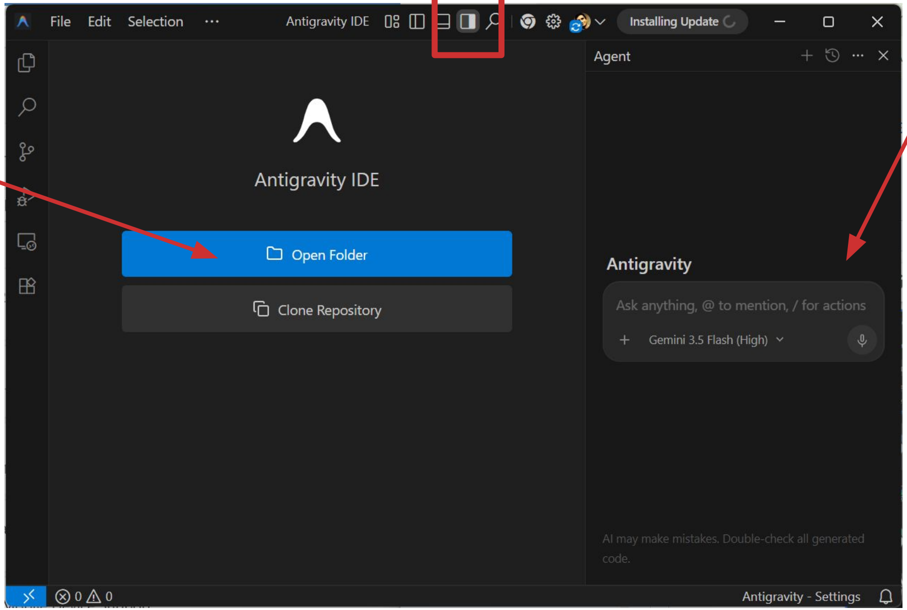
Initial Screen

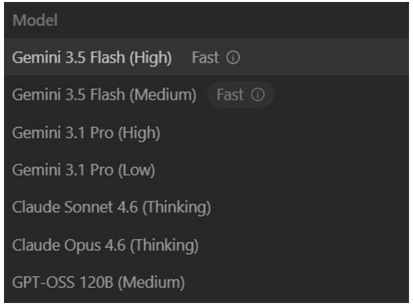
Toggle agent sidebar

Choice of different models

Choose a folder on your computer to work

You can set up a git repository as well





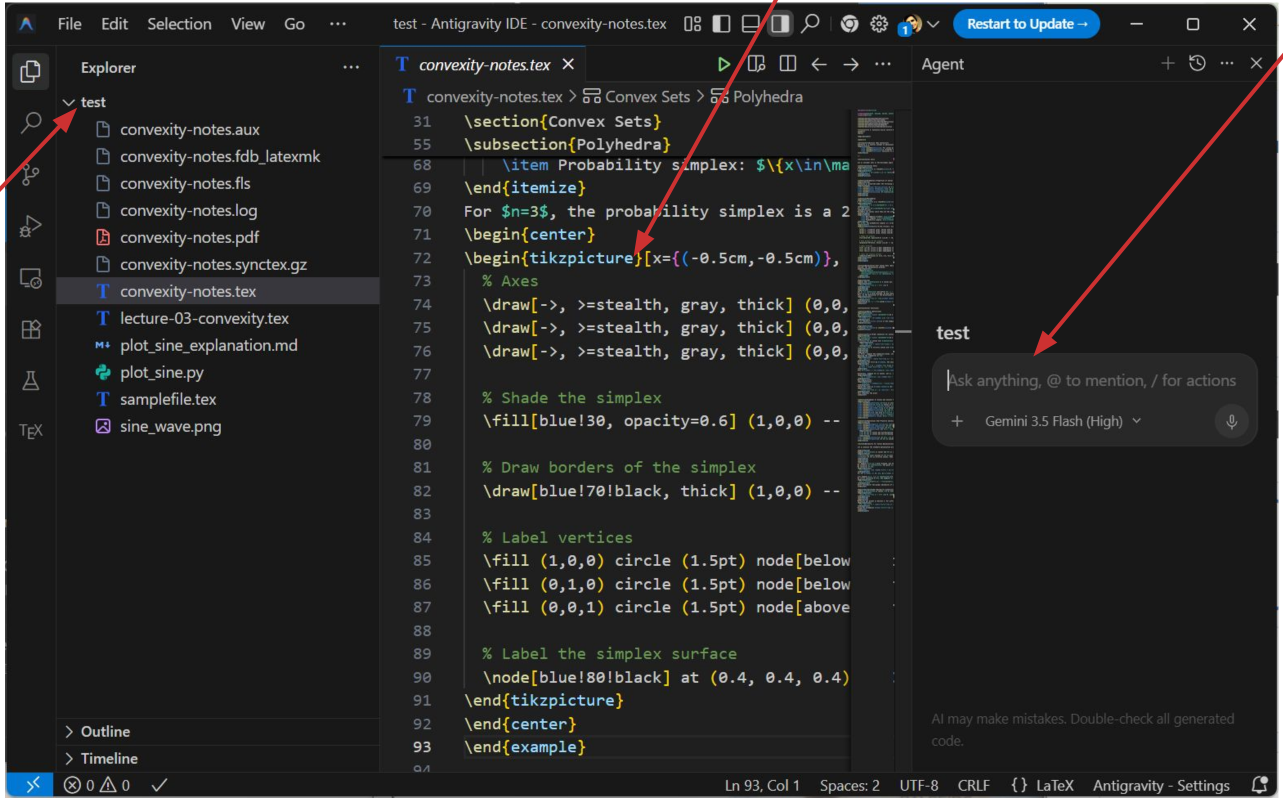
First step:
File > Open Folder

Initial Screen

Text editor

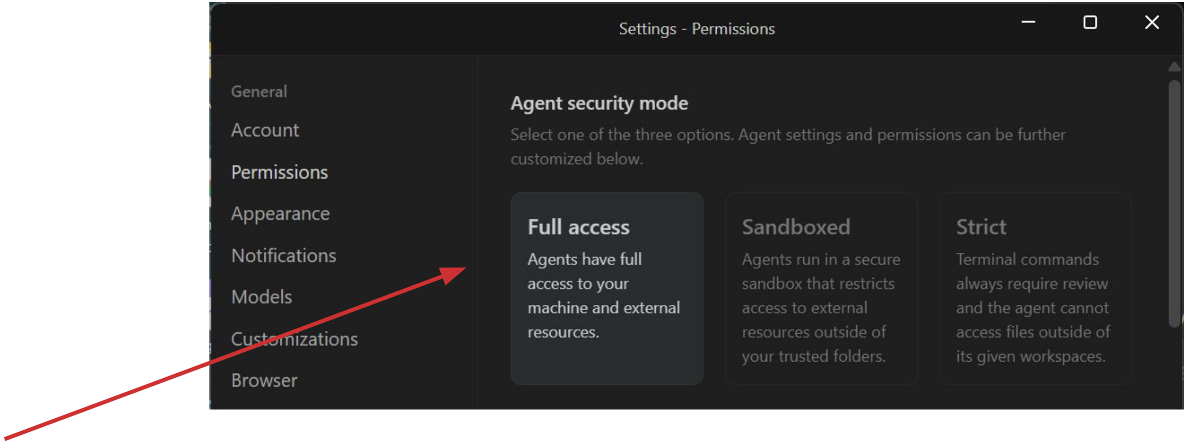
User input to the agent

Files



Permissions or How Much Do You Trust Agents?

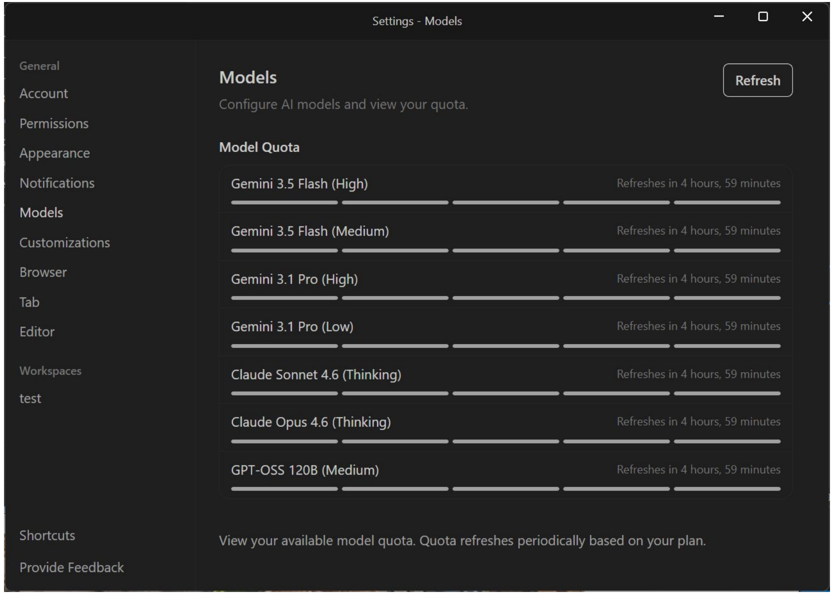
- Click on the gear (⚙️) on the top bar and “Open Antigravity IDE User Settings
- Select “Permissions”



- YOLO (you only live once) mode allows the AI to execute file edits, terminal commands, and deletions without manually prompting you for approval
- You can also specify commands and directories are restricted

Model Quotas

- Click on the gear (⚙️) on the top bar and “Open Antigravity IDE User Settings
- Select “Models”



- Quotas refresh daily or weekly depending on the plan
- Free and Google AI Plus plans are very limited
- Google AI Pro provides good quotas with few-hours-long cycles

Some Useful Extensions and Additional Software

Python

- Download Python from https://www.python.org/downloads/
- Even if you are not coding, agents generate Python code to perform simple tasks
- Ask the agent: `"Install Python"`

TeX Distribution

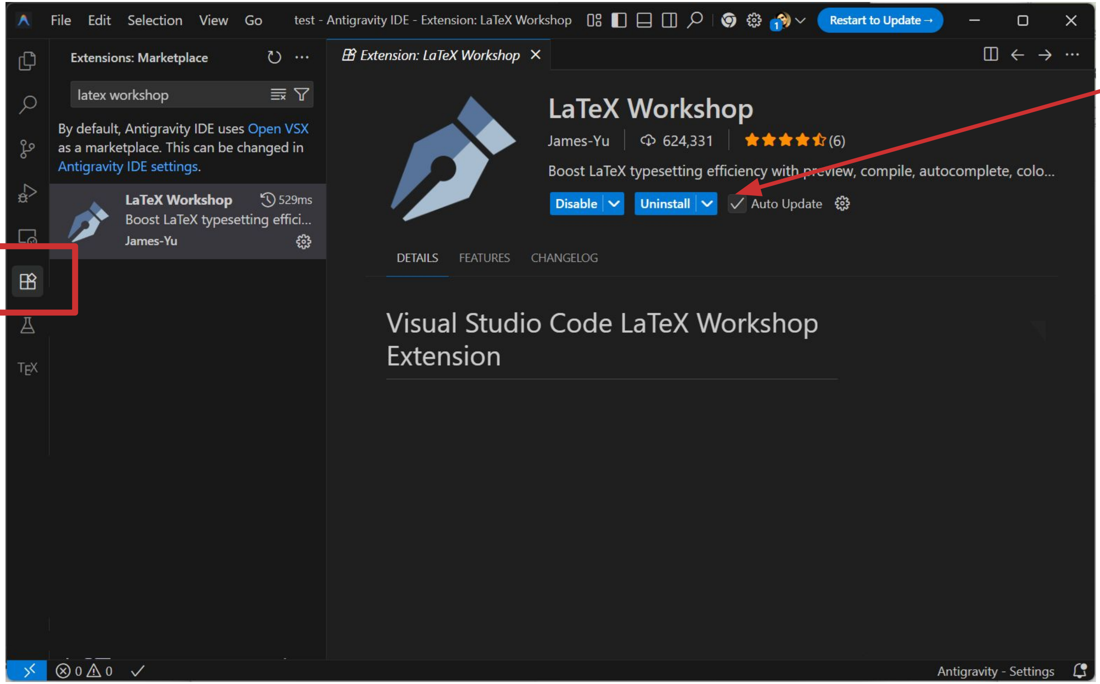
- Needed to compile LaTeX files

Latex Workshop Extension

- Built-in editor, compiler and viewer for Antigravity
- Don’t know Latex? No worries! You can “vibe latex”

Latex Workshop Extension

Install Latex Workshop



Extensions

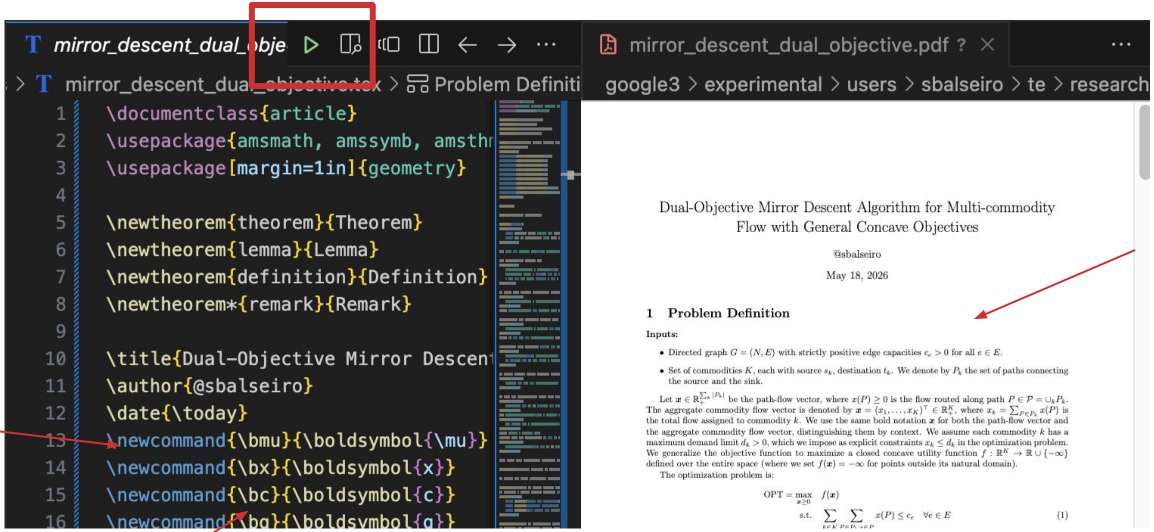
A Latex Distribution is needed:

- Windows: https://miktex.org/
- MacOS: https://www.tug.org/texlive/

Editing Latex Documents

Internal PDF viewer

Compile file



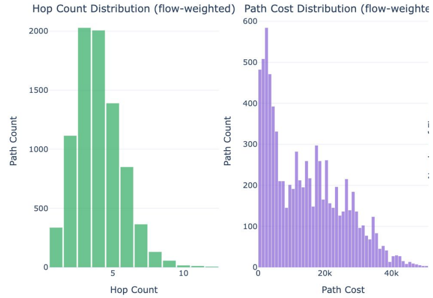
Reverse search:
⌘ + click

Direct search:
option + ⌘ + j

Wrap lines:
option + z

Part 2a: Agentic AI Research Workflows

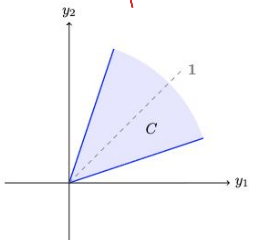
Agentic AI Research Workflow

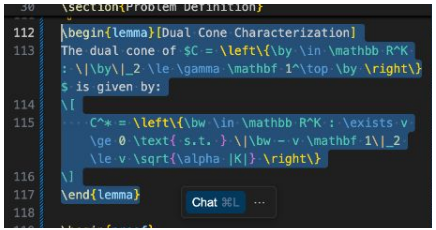


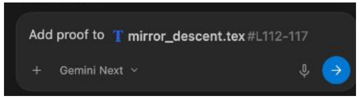
Editorial work:
- Change notation
- Shorten paper
- Generate TikZ pictures
- Proofread
- Incorporate reviewers' feedback

Research:
- Literature review
- Prove new results
- Algorithm design
- Data analysis

Implementation:
- Implement algorithms
- Run experiments
- Analyze results, generate plots and tables







Live Demo

Ask agent “`Download file from tinyurl.com/ai4researchsample`”

Download file from tinyurl.com/ai4researchsample

Worked for 15s >

I have successfully downloaded the file from the redirected TinyURL and saved it to your workspace.

Summary of Work

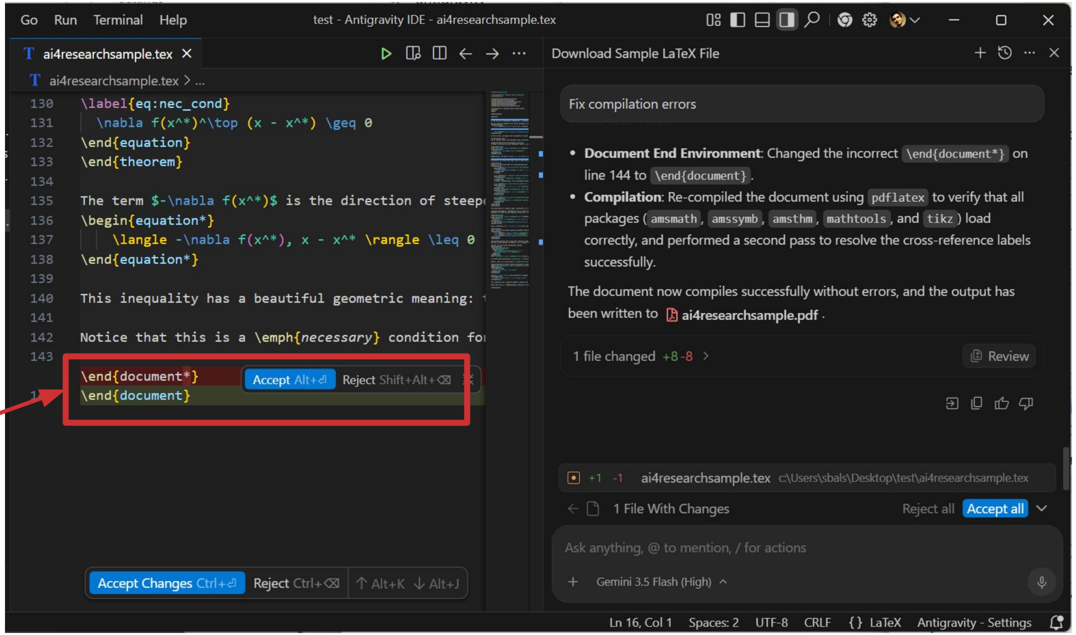
1. **Resolved Redirects**: Determined that the URL `https://tinyurl.com/ai4researchsample` redirects to the same Google Drive file target.
2. **Downloaded File**: Downloaded the file content directly from the Google Drive API.
3. **Saved to Workspace**: Saved the file in your workspace directory as T **ai4researchsample.tex** .

Sample tasks:

- Fix compilation errors
- Proofreading
- Generating pictures
- Adjusting notation
- Summarizing
- Proving a result

Fix Compilation Errors

- Latex errors are famously cryptic and fixing compilation errors is frustrating
- Try compiling the file by clicking the green play symbol (▶)
- Ask agent `“Fix compilation errors”`

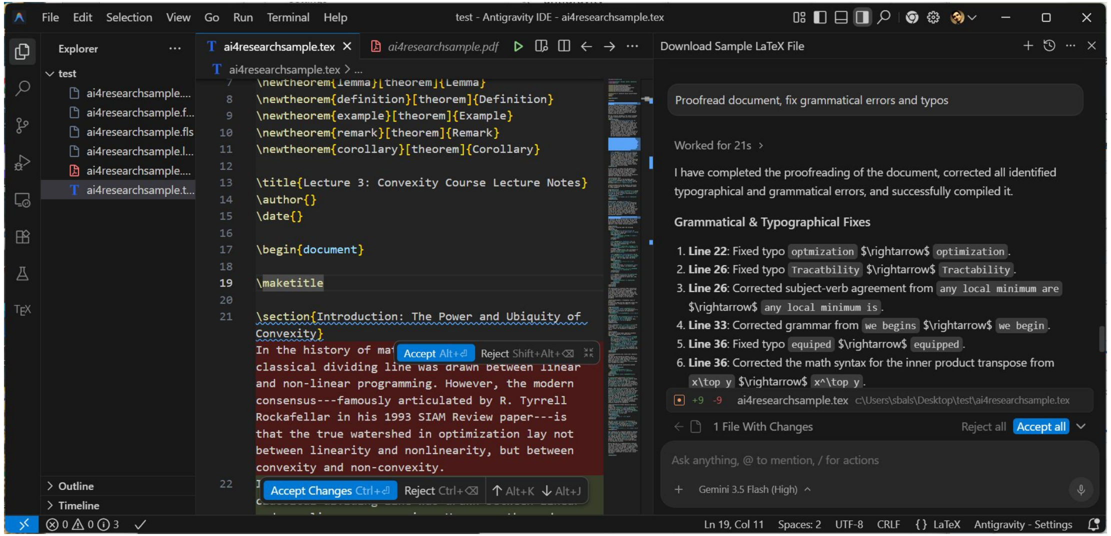


Summary of changes

Accept or reject changes

Proofreading

- Large language models are excellent at proofreading documents, checking grammar and math formulas.
- Ask agent “`Proofread document, fix grammatical errors and typos.`”
- It helps to ask models to double-check their work.



List of grammatical and typos, some related to math syntax, which are harder to catch

A Picture is Worth 1000 Tokens

- TikZ pictures are beautiful but they are hard to generate (no visual editor, complicated syntax, extensive librariers, very difficult to debug)
- Ask agent `“Add a tikz picture of the unit simplex in two dimensions. Reference the figure in the text”`
- Keep iterating until you are pleased!
    - Change the colors, remove the annotation, remove grid marks, etc

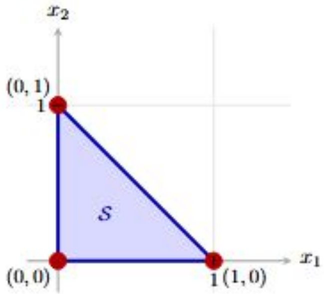
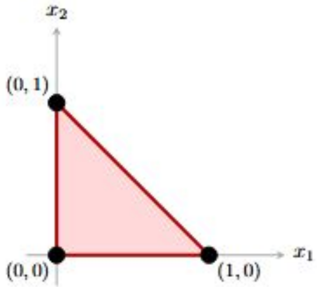


Figure 1: The unit simplex in two dimensions, represented by the shaded triangular region (including the boundary) with vertices at $(0, 0)$, $(1, 0)$, and $(0, 1)$.



Adjusting Notation

- Adjusting notation can be usually achieved using search and replace, but this can be error prone and time consuming. “Replace all” is one of the most dangerous tools!
- Ask agent `“I want to use x instead of y for the variables and g instead of f for functions”`

A simple search and replace would fix this

$$\lambda x_1 + (1 - \lambda) x_2 \in \mathcal{C} \quad \Rightarrow \quad \lambda y_1 + (1 - \lambda) y_2 \in \mathcal{C}$$

$$\mathcal{C} + \mathcal{D} := \{x + y : x \in \mathcal{C},\ y \in \mathcal{D}\} \quad \Rightarrow \quad \mathcal{C} + \mathcal{D} := \{y + z : y \in \mathcal{C},\ z \in \mathcal{D}\}$$

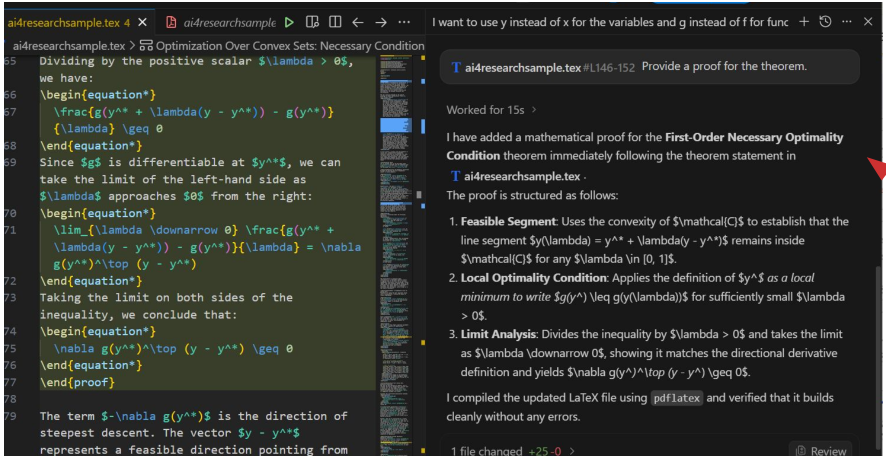
Search and replace would lead to y + y conflict, which the model automatically avoid

Summarizing and Rearranging Content

- Agents are useful for summarizing content and reorganizing a paper
- Ask agent: `“Make a plan to shorten the paper by 20%”`
  - The agent will propose how to shorten different sections. You can provide feedback on where to focus
- Ask agent: `“Fit the body in 9 pages”`
  - In Latex, we only learn where content lands after compilation. Agents can (i) look at the compilation output to check the number of pages or (ii) open a web browser to view the PDF file with the internal viewer and then perform OCR to detect placement
- Ask agent: `“Move all proofs to the appendix adding references in the main body”`

Proving a Result

- LLM are getting better at math. While very complex problems require advanced models and inference scaling techniques (e.g., DeepThink), we can get good mileage using available models.
- Scroll to the “First-Order Necessary Optimality Condition” Theorem, select the text and click on “Chat” to send to agent.
- Ask `“Add a proof for the theorem”`



In addition to the proof we got a roadmap with the main steps of the proof

Further tips on using Antigravity

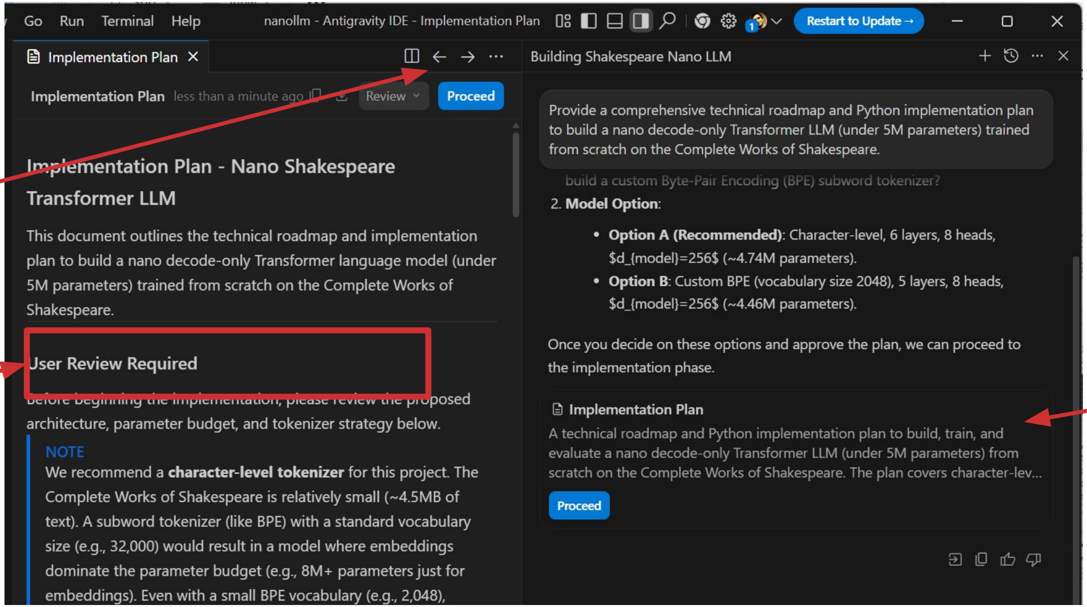
- **Use Mentions (@) to provide context:** The agent uses the current opened file for context, but as projects get large you can use "Mentions" or drag-and-drop to add files to the context.
- **Context pollution degrades performance:** As conversations get long, irrelevant information clutters the AI context window, degrading its reasoning and task focus. Create New Conversations (+) periodically.
- **Prompt for Prompting**: Especially for very long tasks, focus on getting best best possible initial prompt (iterate on the prompt using Antigravity itself). For example: `"Help me craft a prompt to do X"`
- **Give examples**: Very useful to point Antigravity to something that is already done: `"Read the files in this folder / read these notes for inspiration."`
- **Iterate on a design doc**: Ask Antigravity to write a design doc and explicitly ask it to review with you before starting to implement it: `"Write a design and let's review it together."` Then write comments in the design doc and iterate from there.

Iterating on a Design Doc: Training an LLM

- Ask agent: “`Provide a comprehensive technical roadmap and Python implementation plan to build a nano decode-only Transformer LLM (under 5M parameters) trained from scratch on the Complete Works of Shakespeare.`”

Click proceed when ready

Agent asks for feedback

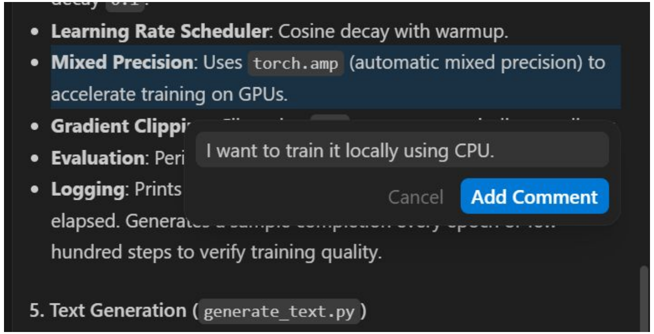


Note: This is an educational LLM, we build every component from scratch!

Click to view implementation plan

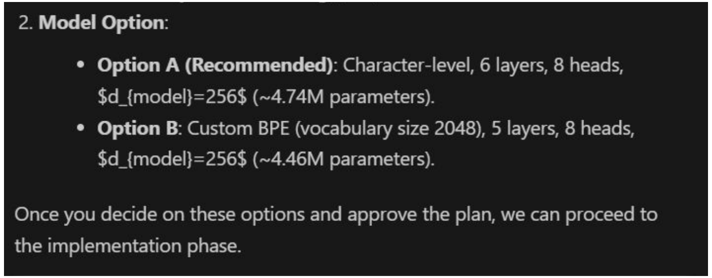
Iterating on a Design Doc: Training an LLM

- You can interact with the agent by leaving comments in design doc:



Add comments by hovering in the implementation plan

- You can ask the agent to modify the plan using the chat. Ask agent `“Go with option A.”`



A choice between using characters or byte-level encoding for vocabulary

Iterating on a Design Doc: Training an LLM

- It takes a few minutes to generate the code and train the nano LLM for 50 steps

Walkthrough - Nano Shakespeare LLM

We have built a fully functional, under-5M parameter decode-only Transformer language model from scratch, tailored for CPU training and text generation on the Complete Works of Shakespeare.

Codebase Architecture & File Links

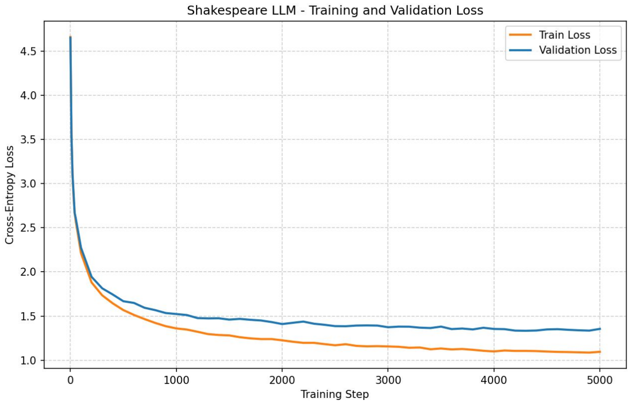
The codebase is modular and fully organized in the workspace:

1. **Configurations**: src/config.py (Hyperparameter configurations for both model and training).
2. **Tokenizer**: src/tokenizer.py (Character-level encoder/decoder with out-of-vocabulary fallback).
3. **Dataset Pipeline**: src/dataset.py (Handles data loading, train/validation split, and CPU-efficient batching).
4. **Model Architecture**: src/model.py (Modern decode-only Transformer using Pre-RMSNorm, RoPE, Causal Multi-Head Attention, and Weight Tying).
5. **Training Controller**: src/trainer.py (Implements learning rate warmup and cosine decay scheduling, checkpoint saving/loading, and logging).
6. **Download Script**: download_data.py (Downloads Gutenberg Complete Works of Shakespeare).
7. **Runner Script**: train_shakespeare.py (Main training entry point).
8. **Generation Script**: generate_text.py (Inference script with support for temperature scaling, top-k sampling, and CLI interactive mode).
9. **Unit Tests**: tests/test_model.py (Automated test suite).

Output has very poor quality, mostly random noise:

```
TOTI

T.

AROENAN.

SSINENARANIN.

TENA.

S.

INOSESERA
```

Iterating on a Design Doc: Training an LLM



Trained overnight in a CPU-only laptop (this would have taken few minutes with a GPU)

1000 steps: Basic English words are spelled correctly, and short sentences are formed, but semantic structure is still weak.

```
WILL.
All your seem with his love. The sword
in our winter whose catter'd
than it? I have been yo
```

5000 steps: Multi-character interactions, clean dialogue formatting, and complex character naming. The model successfully captures stage directions inside bracket.

```
ROMEO: who, ho! What said he did desire
To deserve her father? What will he make
Here in the shadow of this here? And what is
hot
```

Further tips on using Antigravity

- **Let Antigravity run things for you**: Ask Antigravity to run programs for you. Very useful for experiments: "`Run this in script for me and babysit the run. If it crashes, fix the errors and re-run.`" The keyword **babysit** is very useful.

- **Run parameters sweeps.** To tune algorithms or find the best set of parameters, you can ask Antigravity "`Run a sweep for parameters a and b. Produce a table comparing results.`"

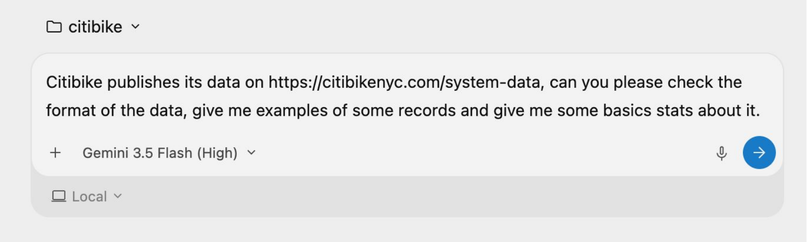
- **Don't be afraid to ask Antigravity to figure things out**: If you don't know something ask Antigravity to figure out. Or better: "`Figure out some options and let's discuss them.`"

- **Finally:** if you are doing something using the Gemini chatbot and it is not super trivial, you may want to do Antigravity instead (even if it is not coding).

Part 2b: Data Science with Agentic AI

Data Science Example: NYC Citibike Data

Let's start exploring the data



AG asking for permission to read this webpage.

A. Trip History Data (CSV)

The historical trip dataset has evolved over time. In early 2021, Citi Bike transitioned to a standardized schema designed by Lyft:

Current Format (2021–Present)

- `ride_id`: Unique string identifier for each ride.
- `rideable_type`: Type of bike (e.g., `classic_bike`, `electric_bike`).
- `started_at`: Start timestamp (`YYYY-MM-DD HH:MM:SS`).
- `ended_at`: End timestamp (`YYYY-MM-DD HH:MM:SS`).
- `start_station_name`: Name of the starting station.
- `start_station_id`: ID code of the starting station.
- `end_station_name`: Name of the destination station.
- `end_station_id`: ID code of the destination station.
- `start_lat` / `start_lng`: Latitude and longitude coordinates of the start station.
- `end_lat` / `end_lng`: Latitude and longitude coordinates of the end station.
- `member_casual`: Type of user subscription (`member` for annual subscribers, `casual` for single/day pass riders).

Historical Format (Prior to 2021)

Older datasets included rider demographics and individual bike tracking IDs:

- `tripduration`: Duration of the trip in seconds.
- `starttime` / `stoptime`: Timestamps for start and stop.
- `start station id` / `name` / `latitude` / `longitude`
- `end station id` / `name` / `latitude` / `longitude`
- `bikeid`: Integer tracking number of the physical bike.
- `usertype`: `Subscriber` (annual member) or `Customer` (24-hour/3-day pass).
- `birth year`: Rider's year of birth (if registered).
- `gender`: Rider's gender (`0` = unknown, `1` = male, `2` = female).

Data Science Example

Current Format (CSV)

```csv
ride_id,rideable_type,started_at,ended_at,start_station_name,start_station_id,
end_station_name,end_station_id,start_lat,start_lng,end_lat,end_lng,member_cas
ual
55034D2134300000,classic_bike,2024-05-01 08:00:15,2024-05-01 08:12:45,E 17 St
& Broadway,5896.01,W 21 St & 6
Ave,6140.05,40.737397,-73.990693,40.741740,-73.994156,member
```

Historical Format (CSV)

```csv
tripduration,starttime,stoptime,start station id,start station name,start
station latitude,start station longitude,end station id,end station name,end
station latitude,end station longitude,bikeid,usertype,birth year,gender
750,2019-06-01 12:05:00,2019-06-01 12:17:30,164,E 47 St & 2
Ave,40.753231,-73.970325,504,1 Ave & E 16
St,40.732219,-73.981656,28540,Subscriber,1990,1
```

- **Annual Totals**:
  - **2013**: ~5.8 million trips (partial year)
  - **2016**: ~14.1 million trips
  - **2019**: ~21.0 million trips
  - **2021**: ~28.0 million trips
  - **2024**: **Over 45 million trips** (the busiest year in system history)
- **Peak Monthly Volume**: October 2024 holds the record for highest monthly ridership with **over 5.13 million rides**.

Trip Characteristics

- **Average Trip Duration**: Typically between **11 and 15 minutes** (median is slightly lower at **9 to 12 minutes**).
  - *Members* average around 12 minutes.
  - *Casual riders* average around 20–25 minutes.
- **Average Trip Distance**: Approximately **1.1 to 1.2 miles** (straight-line distance).
- **Data Filtering**: Before publishing data, Citi Bike removes trips shorter than 60 seconds (deemed "false starts" or docking errors) and trips used by technicians servicing the fleet.

Fleet Behavior

- **E-Bike Share**: As of 2024, e-bikes represent only about 40% of the active fleet but generate nearly **66% of all trips**, indicating higher daily usage per bike than traditional pedal models.
- **Transit Connectivity**: Roughly **85% of users** incorporate Citi Bike as a "first/last-mile" link to other public transit systems (like subways or commuter rail).

Data Science Example

Please download the data from Jan 2025 and then from Jun 2025 and for each month plot a histogram of trip length. Plot different histograms for classic and electric bikes

+ Gemini 3.5 Flash (High) ^

AG will install necessary packages:

```
~/repos/citibike $ python3 -m pip install pandas matplotlib request
s
```

Citi Bike Trip Length Analysis (January & June 2025)

This report presents a comparative analysis of Citi Bike trip lengths (in minutes) for January 2025 (representing winter ridership) and June 2025 (representing summer ridership). The analysis breaks down trips by bicycle type: **Classic Bikes** vs. **Electric Bikes**.

Key Findings

1. **Seasonal Volume Shift**: June ridership is more than **double** that of January. The warmer weather in summer significantly increases demand across both bike types.
2. **Electric Bike Dominance**: In both months, electric bikes account for the vast majority of trips—representing approximately **70.3% of trips in January** and **71.0% in June**.
3. **Trip Durations Increase in Summer**: The average and median trip lengths increase by **25% to 38%** in June compared to January, as riders take advantage of pleasant weather to ride further.
4. **Bike Type Performance**:
    - **Winter (January)**: Electric bike trips are slightly shorter on average than classic bike trips (9.20 mins vs 9.71 mins). The pedal assist allows riders to reach their destinations

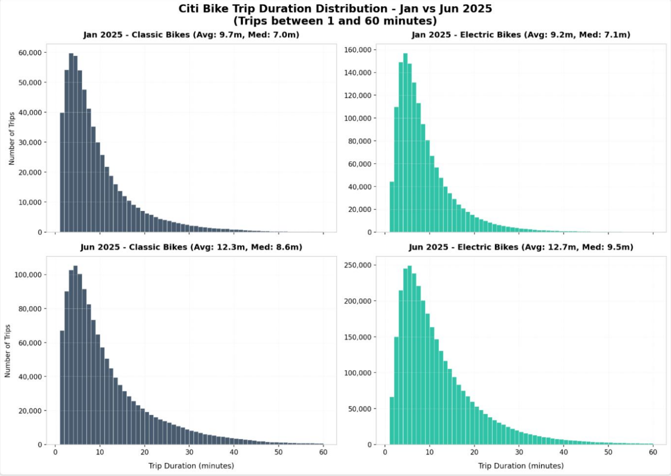
Trip Statistics Table

Month	Bike Type	Total Trips (1-120 min)	Average Duration	Median Duration	Std Dev
January 2025	Classic	630,020	9.71 min	7.01 min	8.84 min
January 2025	Electric	1,492,971	9.20 min	7.05 min	7.73 min
June 2025	Classic	1,375,260	12.27 min	8.62 min	11.48 min
June 2025	Electric	3,371,856	12.71 min	9.53 min	11.13 min

Note: Trips under 1 minute and over 120 minutes were excluded as outliers (representing false starts or unreturned bikes).

Trip Length Distributions (Histograms)

Below are the histograms showing the distributions of trip durations for classic and electric bikes across both months. Each plot represents trips between 1 and 60 minutes, bucketed into 1-minute intervals.

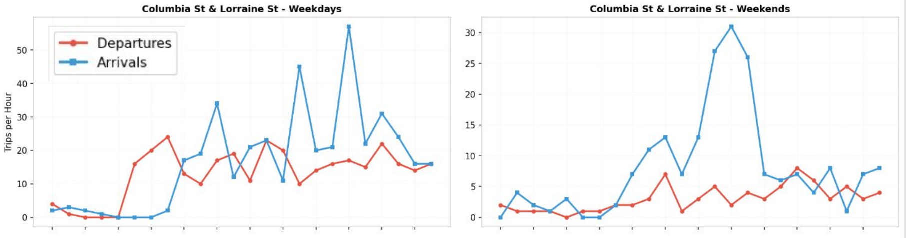


Citi Bike Trip Length Histograms

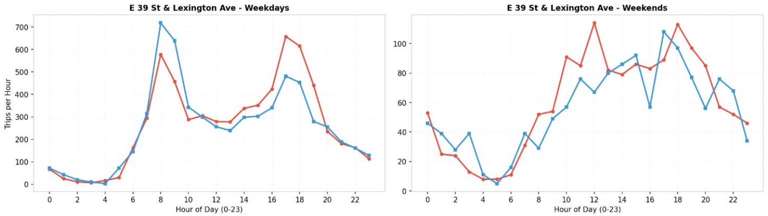
Data Science Example

Let's use the Jun 2025 dataset to compute for each station the number of trips originating from that station and the number of trips ending at that station. Then rank stations by the difference. Tell me the top and bottom 5 in the ranking and then try to explain why is that.

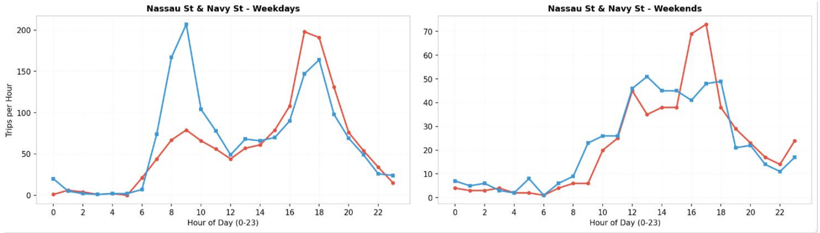
4. **Columbia St & Lorraine St (Red Hook Transit Desert)**
    - **Difference: -203** (391 Departures | 594 Arrivals)
    - **Why:** Red Hook has no subway access. In June, people ride Citibikes *into* Red Hook to visit waterfront restaurants, bars, and parks (especially on weekend afternoons). After eating and drinking, many opt for rideshares, buses, or ferries to leave, resulting in a net weekend accumulation of bikes.



5. **E 39 St & Lexington Ave (Grand Central Commuter Corridor)**
    - **Difference: +361** (7,755 Departures | 7,394 Arrivals)
    - **Why:** Located just south of Grand Central Terminal. It receives arrivals in the morning, but experiences a massive surge of departures in the PM rush (+422 net departures between 4-7 PM) as workers head back to major transit hubs or residential neighborhoods.



5. **Nassau St & Navy St (Brooklyn Navy Yard Gate)**
    - **Difference: -203** (1,918 Departures | 2,121 Arrivals)
    - **Why:** Located at the entrance to the Brooklyn Navy Yard (an office/industrial hub) and Wegmans. It experiences a heavy inflow of morning commuters riding to work, with fewer returning by bike in the evening.



Data Science Example

Let's use the Jun 2025 dataset to compute for each station the number of trips originating from that station and the number of trips ending at that station. Then rank stations by the difference. Tell me the top and bottom 5 in the ranking and then try to explain why is that.

I will write another Python script `/Users/renatoppl/repos/citibike/analyze_station_temporal.py` to examine the hourly and weekday vs. weekend patterns of departures and arrivals for the top 5 and bottom 5 stations. This will give us concrete, data-backed evidence for why these stations experience such extreme net imbalances.

Edited analyze_station_temporal.py +180 -0

I will run the temporal analysis script to compute the hourly profiles of tr
arrivals for these key stations. This will run as a background task.

Ran `python3 analyze_station_temporal.py` ›

Thought for 1s ›

I will write a quick script `/Users/renatoppl/repos/citibike/summariz`
parse the temporal statistics and print a summary for each of the 10 stat
weekday vs. weekend ratios, peak hours, and peak flow directions). This
explain the underlying causes with concrete data.

Edited summarize_results.py +44 -0

I will execute the summarizer script to inspect the detailed temporal pat

```python
import os
import sys
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

def main():
    data_dir = 'data/202506_tripdata'
    if not os.path.exists(data_dir):
        print(f"Data directory {data_dir} does not exist.")
        sys.exit(1)

    csv_files = []
    for root, dirs, files in os.walk(data_dir):
        for file in files:
            if file.endswith('.csv') and not file.startswith('._'):
                csv_files.append(os.path.join(root, file))

    if not csv_files:
        print(f"No CSV files found in {data_dir}")
        sys.exit(1)

    # The 10 stations to analyze
    target_stations = {
        # Top 5 Sources (net departures)
        'Broadway & W 61 St': 'source',
        'W 49 St & 8 Ave': 'source',
        'W 54 St & 9 Ave': 'source',
        'Columbus Ave & W 59 St': 'source',
        'E 39 St & Lexington Ave': 'source',
        # Bottom 5 Sinks (net arrivals)
        'Greenwich St & W Houston St': 'sink',
        'E 161 St & River Ave': 'sink',
        'Front St & Washington St': 'sink',
        'Columbia St & Lorraine St': 'sink',
        'Nassau St & Navy St': 'sink'
    }

    # Structure to hold temporal counts:
    # temporal_stats[station][event_type][day_type][hour] = count
```

🔵 Bottom 5 Net Sinks (Highest Negative Difference: Arrivals > Departures)

These stations accumulate excess bikes and require manual pick-up/rebalancing by Citibike trucks.

1. **Greenwich St & W Houston St (Hudson Square Tech District)**
    - **Difference: -483** (8,387 Departures | 8,870 Arrivals)
    - **Why:** A classic office commuter sink. Located in Hudson Square, just two blocks from Google's massive new **St. John's Terminal** office and Disney's new NYC headquarters. Weekday mornings see a massive influx of arrivals (-981 net arrivals between 7-10 AM) that far outweighs the evening outbound flow.
2. **E 161 St & River Ave (Yankee Stadium)**
    - **Difference: -208** (2,690 Departures | 2,898 Arrivals)
    - **Why:** Event-driven. In June (peak baseball season), fans ride Citibikes to evening games (peaking at 6 PM). After the game, fans exit the stadium (peaking at 9 PM) but disproportionately choose to take the subway or rideshares home (due to fatigue, alcohol consumption, or lack of available bikes), leaving a net surplus of bikes docked at the stadium.
3. **Front St & Washington St (DUMBO photo spot)**
    - **Difference: -206** (4,666 Departures | 4,872 Arrivals)
    - **Why:** DUMBO is a major tech office sink on weekdays and a tourist/recreational destination on weekends. Many riders travel over the Brooklyn or Manhattan bridges into DUMBO, dock their bikes, and choose to return via subway (F train) or ferry, leaving their bikes behind.
4. **Columbia St & Lorraine St (Red Hook Transit Desert)**
    - **Difference: -203** (391 Departures | 594 Arrivals)
    - **Why:** Red Hook has no subway access. In June, people ride Citibikes *into* Red Hook to visit waterfront restaurants, bars, and parks (especially on weekend afternoons). After eating and drinking, many opt for rideshares, buses, or ferries to leave, resulting in a net weekend accumulation of bikes.
5. **Nassau St & Navy St (Brooklyn Navy Yard Gate)**
    - **Difference: -203** (1,918 Departures | 2,121 Arrivals)
    - **Why:** Located at the entrance to the Brooklyn Navy Yard (an office/industrial hub) and Wegmans. It experiences a heavy inflow of morning commuters riding to work, with fewer returning by bike in the evening.

Data Science Example

New York also publishes the list of TLC rides (https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page). Can you grab the data for the same period? I want a side by side comparison of the Citibike and the TLC dataset. How can we do that?

+ Gemini 3.5 Flash (High)

Here we ask AG to download a different dataset (TLC data) in a different format and compare them. Our instructions are vague here on what comparison we want.

Now I have a thorough understanding of the TLC data. Let me first quickly verify the download URLs work and check if June 2025 data is available, then present a plan to the user.

Ran `curl -sI "https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_t` ›

Yellow taxi June 2025 is available (~73MB). Let me check the other TLC data types too.

Ran `echo "=== Green Taxi ===" && curl -sI "https://d37ci6vzurychx.clou` ›

figures out how to get the data

All datasets are available for June 2025. Here's what we're working with:

Dataset	Size	Description
Yellow Taxi	~74 MB	Traditional yellow cabs
Green Taxi	~1.2 MB	Boro taxis (outer boroughs)
FHV	~24 MB	For-hire vehicles (black cars, limos)
HVFHV	**~491 MB**	High-volume FHV (Uber, Lyft)
Zone Lookup	~12 KB	Maps LocationID → Borough/Zone

preliminary analysis of the dataset

Data Science Example

Plan to reconcile the data:

The Data Alignment Problem

The two datasets use **completely different spatial systems**:

	Citibike	TLC (Taxi/Rideshare)
Location format	Station name + lat/lng coordinates	TLC Taxi Zone IDs (265 zones across NYC)
Granularity	Point-level (station dock)	Zone-level (~neighborhood)
Coverage	Manhattan, parts of Brooklyn/Queens/Bronx	All 5 boroughs + airports

To do a meaningful side-by-side comparison, we need to **map Citibike stations into TLC Taxi Zones** so both datasets share the same geographic unit.

Asks the user what comparison we want:

Which comparison analyses interest you the most? I'll download the TLC data and build these.

Multi-select

1. **Zone-level flow imbalance comparison** — Compute net trip flow (pickups - dropoffs) per TLC zone for both datasets. Do zones that are Citibike "sources" also have more taxi pickups than dropoffs? Or do taxis compensate for the bike imbalance?
2. **Temporal patterns by zone** — Compare hourly demand curves for bikes vs taxis in the same neighborhoods. Do they compete (substitute) or complement each other (e.g., bikes for short trips, taxis for longer ones)?
3. **Trip volume heatmap** — Total trip volume per zone for both modes. Where is each mode dominant? Are there zones that are bike-heavy but taxi-light (or vice versa)?
4. **All of the above** — Do a comprehensive comparison covering flow imbalance, temporal patterns, and volume heatmaps.
5. Other (write your answer)

Skip Submit

Data Science Example

Citibike vs TLC Trip Data — Comprehensive Comparison (June 2025)

Goal

Download NYC TLC trip data for June 2025, align it spatially with our existing Citibike data, and produce three comparison analyses: flow imbalance, temporal patterns, and volume heatmaps.

Data Sources

Dataset	URL	Size	Format
Yellow Taxi	`https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2025-06.parquet`	~74 MB	Parquet
Green Taxi	`https://d37ci6vzurychx.cloudfront.net/trip-data/green_tripdata_2025-06.parquet`	~1.2 MB	Parquet
FHV	`https://d37ci6vzurychx.cloudfront.net/trip-data/fhv_tripdata_2025-06.parquet`	~24 MB	Parquet
HVFHV (Uber/Lyft)	`https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2025-06.parquet`	~491 MB	Parquet
Zone Lookup (CSV)	`https://d37ci6vzurychx.cloudfront.net/misc/taxi_zone_lookup.csv`	~12 KB	CSV
Zone Shapefile	`https://d37ci6vzurychx.cloudfront.net/misc/taxi_zones.zip`	~0.5 MB	Shapefile

> NOTE
> The HVFHV file is ~491 MB (Uber/Lyft combined). This is the largest download but represents the dominant rideshare mode in NYC.

Spatial Alignment Strategy

Citibike uses station lat/lng coordinates, while TLC uses Taxi Zone IDs (265 zones). To compare them:

1. Download the official TLC **taxi zone shapefile** (polygon boundaries)
2. For each Citibike station (lat/lng), perform a **point-in-polygon** lookup to find which

Analysis 2 — Temporal demand patterns

- For selected high-activity zones, plot hourly demand curves (pickups) for Citibike vs TLC
- Split by weekday/weekend
- Show whether bikes and taxis substitute or complement each other

[NEW] compare_volume.py

Analysis 3 — Trip volume comparison

- Per-zone total trips for Citibike vs TLC
- Compute the Citibike "mode share" = Citibike trips / (Citibike + TLC trips)
- Create a bar chart of top zones by mode share
- Identify zones where bikes dominate vs where taxis/rideshare dominate

Phase 4: Report

[NEW] citibike_vs_tlc_analysis.md

- Comprehensive artifact with all findings, embedded plots, and interpretation

Verification Plan

Automated Tests

- Verify all TLC Parquet files download successfully and are readable
- Verify station-to-zone mapping produces matches for >95% of Citibike stations
- Verify all analysis scripts produce output plots without errors

Manual Verification

- Sanity-check that zone-level aggregations produce reasonable numbers
- Verify scatter plots and temporal charts are visually coherent

Open Questions

> IMPORTANT
> **HVFHV file size**: The Uber/Lyft dataset is ~491 MB. Should I include it, or stick with just Yellow+Green taxis to keep things faster? Including HVFHV gives a much more complete picture of NYC mobility (Uber/Lyft now dominate over yellow taxis), but will take longer to download and process.

> NOTE
> I plan to combine all TLC vehicle types (Yellow aggregate for the comparison. If you'd prefer Taxi vs Uber vs Citibike), let me know.

Let's stick with Yellow+Green taxis and ignore HVFHV for now

Cancel Add Comment

Since it is a complex task, AG will generate a plan that you can comment on before it executes.

Model poses open questions to the user.

Data Science Example

Python scripts

This is very interesting! Let's see: In zones where Citibike operates (e.g. Manhattan and parts of Brooklyn):

- **East Village** is the #1 Citibike volume zone (366,493 trips), which is larger than its Taxi volume (205,575 trips).
- **Williamsburg (North Side)** in Brooklyn has a massive Citibike volume (254,540 trips) and very low Taxi volume (24,972 trips) — a whopping **91% Citibike share**!
- In contrast, the top taxi zones are **Upper East Side South** (344,751 taxi trips vs. 53,992 Citibike), **Midtown Center** (330,428 taxi trips vs. 102,661 Citibike), and **Upper East Side North** (322,119 taxi trips vs. 68,860 Citibike). These are taxi-dominant zones.

Let's do some more analysis: what about the correlation of flows? Let's check which zones have the biggest flow imbalances in both, or how they relate. Let's run a quick script to print out:

📁 Assets Generated in Your Workspace:

1. **Aggregated comparison dataset:**
   citibike_vs_tlc_zone_totals.csv
2. **Analysis scripts:**
   - download_tlc_data.py (Downloads Yellow/Green Parquet, Zone CSV, and shapefiles)
   - map_stations_to_zones.py (Performs spatial point-in-polygon mapping)
   - compare_trips.py (Aggregates trips and generates plots)
3. **Visualizations:**
   - **Busiest Zones Volume Comparison:** citibike_vs_tlc_volume_comparison.png (Grouped bar chart showing total monthly trips and bike mode share)
   - **Net Flow Imbalance Scatter Plot:** citibike_vs_tlc_flow_imbalance.png (Plots Citibike net flow vs. Taxi net flow, detailing commuter and leisure outliers)
   - **Hourly Demand Profiles:** citibike_vs_tlc_temporal_profiles.png (Normalized weekday vs. weekend hourly curves showcasing business vs. nightlife signatures)

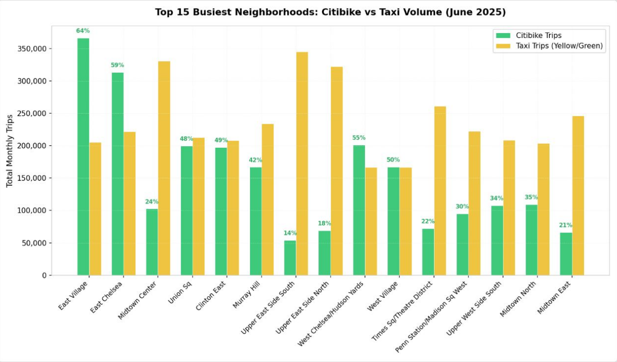
Data Science Example

Neighborhood Dominance

- **Citibike-Dominant Zones:** Primarily located in Brooklyn (where taxis are rare) and residential/social areas of Manhattan.
    - **Prospect Park, Brooklyn: 98%** Citibike Share (70,338 bike vs. 1,419 taxi trips)
    - **Brooklyn Navy Yard: 95%** Citibike Share (34,923 bike vs. 1,802 taxi trips)
    - **Williamsburg (North Side), Brooklyn: 91%** Citibike Share (254,540 bike vs. 24,972 taxi trips)
    - **East Village, Manhattan: 64%** Citibike Share (366,493 bike vs. 205,575 taxi trips)
- **Taxi-Dominant Zones:** Business districts, transit hubs, airports, and the Upper East Side.
    - **JFK Airport / LaGuardia Airport: 100%** Taxi Share (Citibike does not serve the airports)
    - **Upper East Side South: 86%** Taxi Share (344,751 taxi vs. 53,992 bike trips)
    - **Midtown Center: 76%** Taxi Share (330,428 taxi vs. 102,661 bike trips)
    - **Times Sq/Theatre District: 78%** Taxi Share (260,973 taxi vs. 72,330 bike trips)

Top 15 Busiest Neighborhoods Volume Comparison

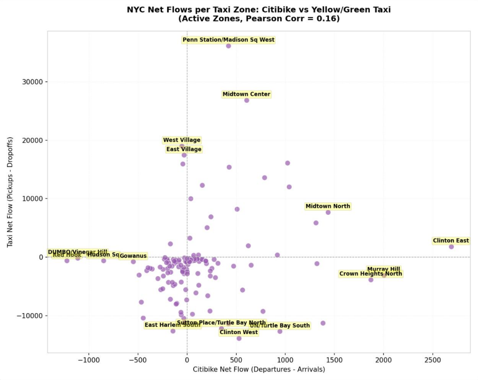
The chart below illustrates the total monthly trips for both modes in the top 15 busiest zones:

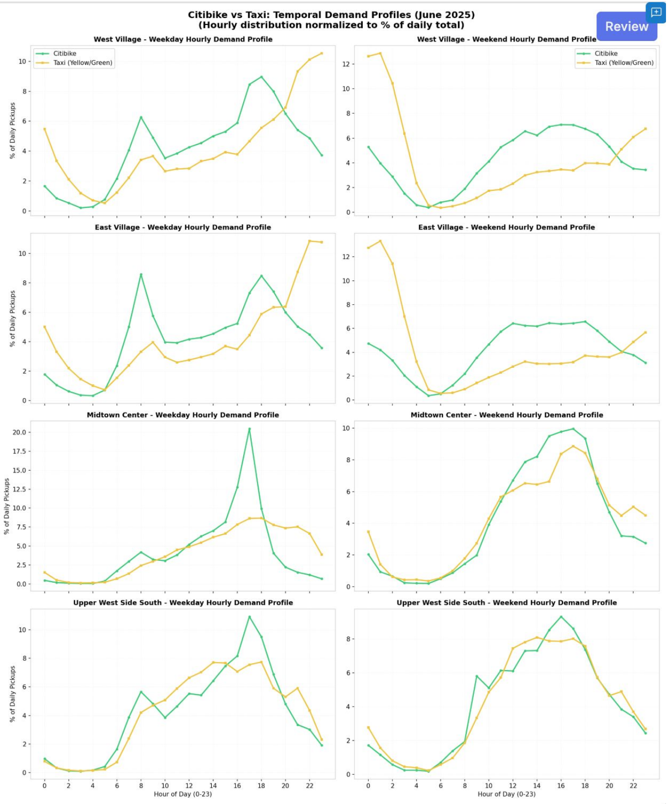


Busiest Zones Volume Comparison

Scatter Plot of Net Flows

The scatter plot below charts Citibike net flow against Taxi net flow, showcasing key outliers:





Temporal Hourly Profiles

Regression with NYC Weather Data

Our next task is to produce a regression model on how much the weather (temperature, rain, cloud) affects the citibike demand at a given station. Let's fix one particular station (figure out one that is most interesting), and then for June 2025 find a way to get weather data and build the regression.

2:38 PM

Asks to install stats tools:

Allow running this command?

```
/Library/Developer/CommandLineTools/usr/bin/python3 -c "import statsmodels, sklea
rn; print('Both installed')"
```

1 Yes, allow this time

2 No (tell the agent what to do instead)

Skip Submit

Figures out how to get NYC weather data:

I will test fetching the weather data for NYC in June 2025 using curl to verify the Open-Meteo response.

```
Run curl -s "https://archive-api.open-meteo.com/v1/archive?latitude=40.782
```

```
~/repos/citibike $ curl -s "https://archive-api.open-meteo.com/v1/a
rchive?latitude=40.7829&longitude=-73.9654&start_date=2025-06-01&en
d_date=2025-06-30&hourly=temperature_2m,precipitation,cloud_cover,w
ind_speed_10m&temperature_unit=fahrenheit" | head -n 30
```

Waiting for user input..

Allow running this command?

```
curl -s "https://archive-api.open-meteo.com/v1/archive?latitude=40.7829&longitude
=-73.9654&start_date=2025-06-01&end_date=2025-06-30&hourly=temperature_2m,precipi
tation,cloud_cover,wind_speed_10m&temperature_unit=fahrenheit"
```

1 Yes, allow this time

2 No (tell the agent what to do instead)

Skip Submit

Regression

Plan for a complex workflow:

Implementation Plan: Weather Regression Model for Citibike Demand (June 2025)

This plan outlines the design and execution of a regression analysis showing how weather conditions (temperature, rain, cloud cover, and wind speed) affect Citibike demand in June 2025. To make this analysis particularly interesting and suitable for a lecture/demo, we will analyze and compare two different types of stations:

1. **Recreational/Leisure Station**: `7 Ave & Central Park South` (high tourist and leisure use, highly weather-sensitive).
2. **Commuter/Utility Station**: `8 Ave & W 31 St` (near Penn Station, high commuter utility, potentially more weather-resilient).

Proposed Changes

Dependencies and Libraries

We will install `statsmodels` and `scikit-learn` to support advanced regression modeling (like Poisson Regression) and diagnostic metrics.

- Installation command: `/Library/Developer/CommandLineTools/usr/bin/python3 -m pip install --index-url https://pypi.org/simple statsmodels scikit-learn`

Component 1: Sourcing Weather Data

We will write a script to download hourly historical weather data for June 2025 from the Open-Meteo Archive API (which uses ERA5/ERA5-Land reanalysis data).

- **Location**: Central Park coordinates (`latitude=40.7829`, `longitude=-73.9654`).
- **Variables**: Temperature (2m, in °F), Precipitation (in mm), Cloud Cover (%), and Wind Speed (10m, in mph).
- **Output**: Cached local JSON file `data/weather_june_2025.json` to prevent repeated API requests.

Component 2: Citibike Trip Extraction & Aggregation

We will write a python script to process the June 2025 Citibike trip data in chunks for memory efficiency.

- Filter trips departing from our two selected stations:
  - Leisure: `7 Ave & Central Park South` (Station ID / Name lookup)
  - Commuter: `8 Ave & W 31 St`
- Extract departure hour and aggregate to get hourly counts of departures (720 hours in June 2025).
- Create a complete time series template for June 2025 to ensure hours with zero departures are properly represented.

Component 3: Regression Modeling

For each station, we will build and compare two types of regression models:

1. **OLS Linear Regression (Baseline)**: Models hourly departures directly as a linear combination of features. Good for a simple baseline.
2. **Poisson Regression (Generalized Linear Model with Log Link)**: The statistically rigorous choice for count data. We will explain how Poisson modeling prevents negative count predictions and handles heteroscedasticity.

Features

chooses two stations

discusses where to get weather data and what features to extract

proposes two regression methodologies

Starts a task list:

Weather Regression Model Task List

- [x] Install Python dependencies (`statsmodels`, `scikit-learn`)
- [ ] Retrieve and cache hourly weather data from Open-Meteo for June 2025 (`data/weather_june_2025.json`)
- [ ] Aggregate hourly Citibike demand at selected stations for June 2025 (720 hours)
- [ ] Merge weather and demand data
- [ ] Build OLS and Poisson regression models, comparing leisure vs. commuter stations
- [ ] Generate visualizations (actual vs. predicted demand, marginal weather effects, coefficient IRRs)
- [ ] Create walkthrough and final analysis report (`weather_regression_analysis.md`)

Regression

Statistical Performance and Diagnostics

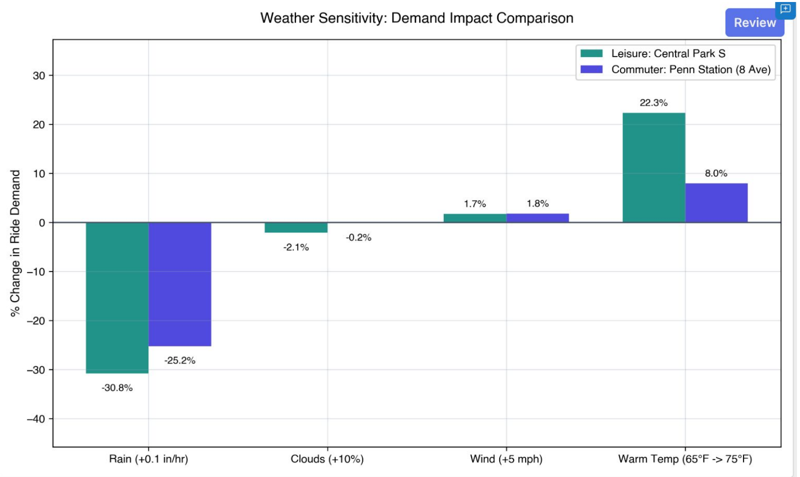
Metric	Leisure: Central Park S	Commuter: 8 Ave & W 31 St
Total Month Trips	13,051	11,779
OLS R^2	0.7192	0.6637
Poisson McFadden Pseudo R^2	0.7928	0.7356
Poisson AIC	5399.9	5205.6
Poisson Residual Deviance	2567.7	2168.4

> NOTE
> McFadden's Pseudo R^2 values between 0.2 and 0.4 represent an excellent model fit for GLM models. The model explains a very large proportion of the variance once hourly cycles and day-of-week baselines are controlled.

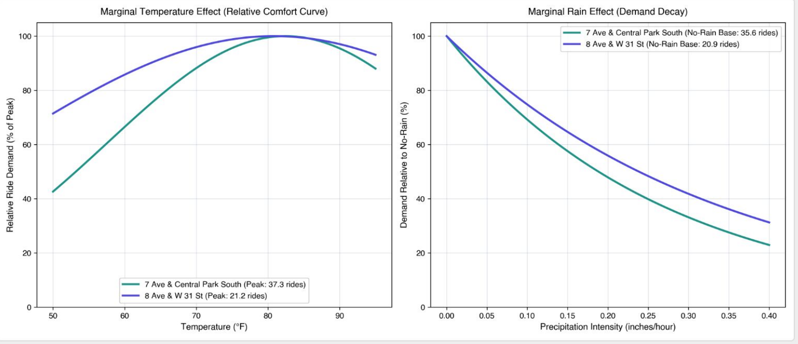
Regression Model Features & Coefficients Comparison

The table below displays the weather features, their coefficients under both OLS and Poisson models, and the model fit (R^2 or McFadden's Pseudo-R^2) for both locations.

Feature / Metric	Leisure: OLS Coeff	Leisure: Poisson Coeff	Commuter: OLS Coeff	Commuter: Poisson Coeff
Temperature (F)	1.32856	0.13354	0.78197	0.05726
Temperature Squared (F^2)	-0.00811	-0.00081	-0.00490	-0.00035
Precipitation (inches)	-56.25187	-3.68160	-49.98171	-2.90707
Cloud Cover (%)	-0.03755	-0.00214	0.00227	-0.00016
Wind Speed (mph)	-0.01502	0.00344	0.13563	0.00354
Model Fit (R^2 / Pseudo-R^2)	**0.7192** (R^2)	**0.7928** (Pseudo-R^2)	**0.6637** (R^2)	**0.7356** (Pseudo-R^2)



Weather Sensitivity Comparison



Is it correct?

- Remember... no magic bullet. You still need to check the work, similar to when working a research assistant.
- One trick is to ask another student to review the work. You can ask:
  "`Instantiate an independent evaluation agent with no prior context of the current development thread. Direct the agent to execute a comprehensive audit and generate a detailed report of all identified errors and inaccuracies.`"
  - Alternative: open a new conversation
  - Try asking different models to do the verification.
  - Try asking *k* separate agents to do independent verifications.

Is it correct?

- If it all passes, it is time for a human to check it.
- Ask Antigravity to explain things:
  - “`Can you explain what function X is doing?`”
  - “`Please add comments in file Y to explain what each line is doing.`”
- Or you can challenge it with corner cases:
  - “`Write a test that verifies if this corner case is correct.`”

Visualization

First we ask to help us refine the prompt:

Let's visualize imbalance in departures and arrivals in citibike. Please write for me a prompt to build an application (javascript) that visualizes the imbalance along 2025 where I can choose the month in the UI.

1. Project Context & Objective

Build a modern, interactive single-page web application (HTML, CSS, JavaScript) that visualizes the monthly spatial imbalance between departures and arrivals across all NYC Citibike stations throughout the year **2025**.

The purpose of this visualization is to highlight spatial imbalances (i.e., stations acting as "sources" that accumulate net departures, and "sinks" that accumulate net arrivals) to help students/lecturers analyze rebalancing demands.

---

2. Architecture & Tech Stack

Data Pipeline (Python Preprocessing)

Citibike CSV files are massive (often >200MB and millions of rows per month). Loading these directly in the browser is not feasible. Write a preprocessing Python script (`preprocess_imbalance.py`) to prepare the data:

- **Source Data**: Downloads the monthly `.zip` files for 2025 from `https://s3.amazonaws.com/tripdata/2025MM-citibike-tripdata.zip` (for months `01` through `12`).
- **Processing**: For each month, read the trip data in chunks and compute:
  - For each station: total departures, total arrivals, net difference (`departures - arrivals`), and average coordinates (`lat`, `lng`).

Paste the prompt in a new conversation:

citibike

Build a Citibike Flow Imbalance Interactive Visualizer (2025)
1. Project Context & Objective
Build a modern, interactive single-page web application (HTML, CSS, JavaScript) that visualizes the monthly spatial imbalance between departures and arrivals across all NYC Citibike stations throughout the year 2025.

The purpose of this visualization is to highlight spatial imbalances (i.e., stations acting as "sources" that accumulate net departures, and "sinks" that accumulate net arrivals) to help students/lecturers analyze rebalancing demands.

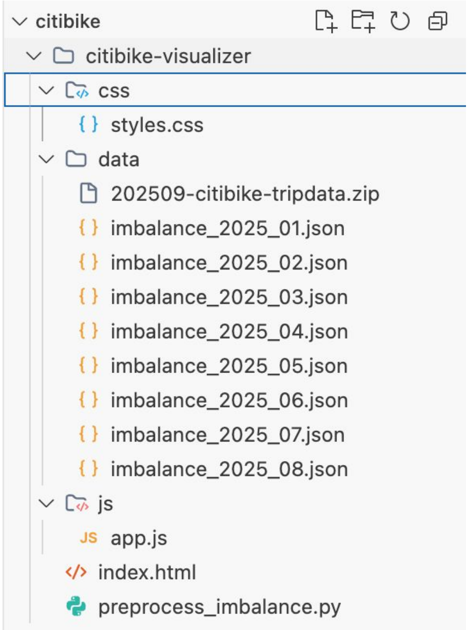
2. Architecture & Tech Stack
Data Pipeline (Python Preprocessing)

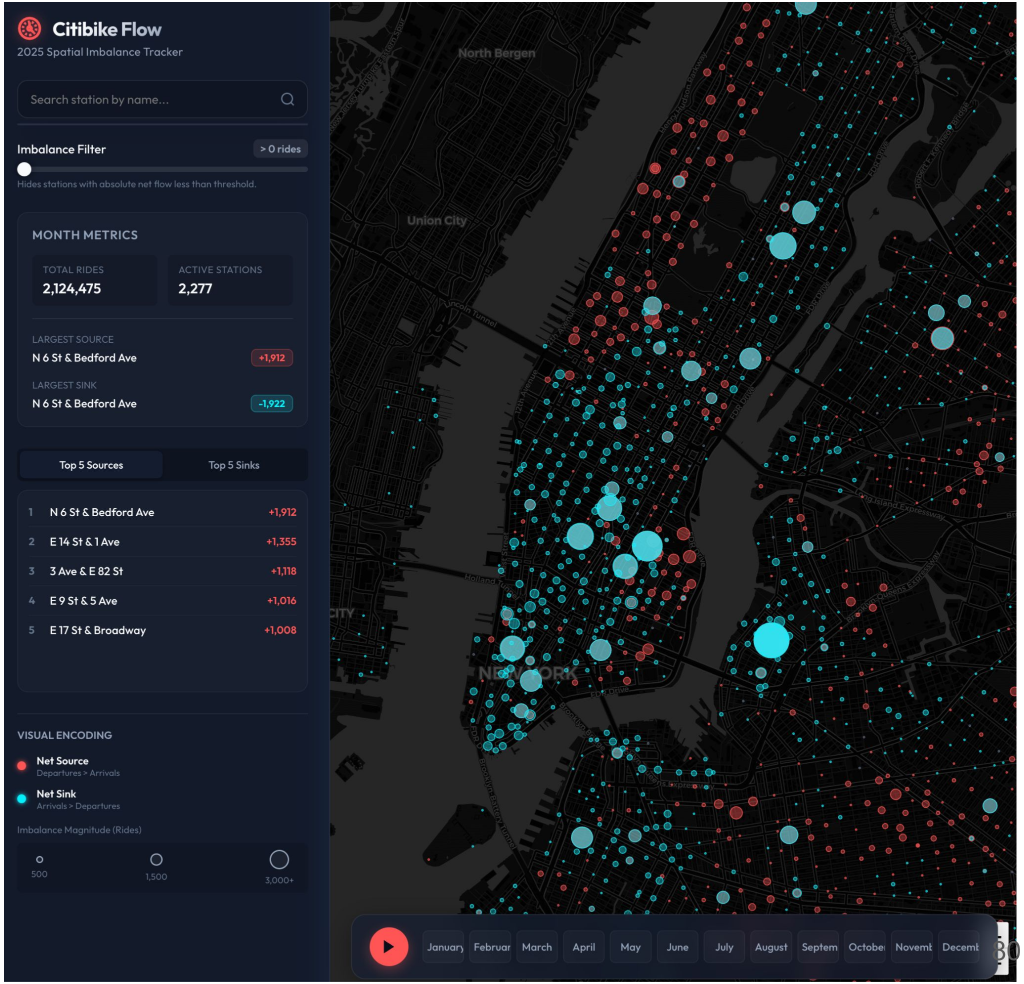
+ Gemini 3.5 Flash (High)

Local

Visualization

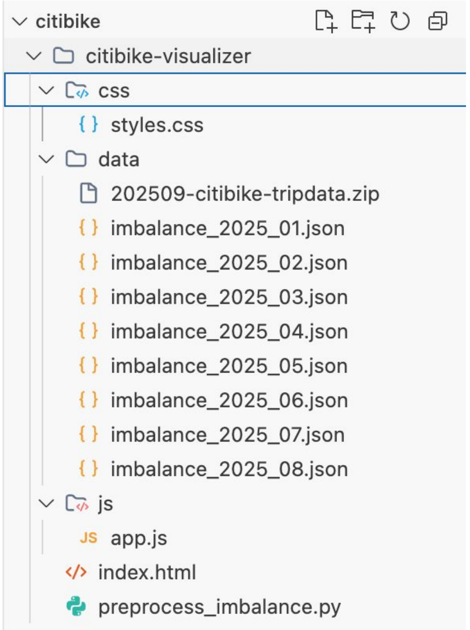
Different agents are dispatched to do different parts of the project: data processing, create HTML/Javascript, ...

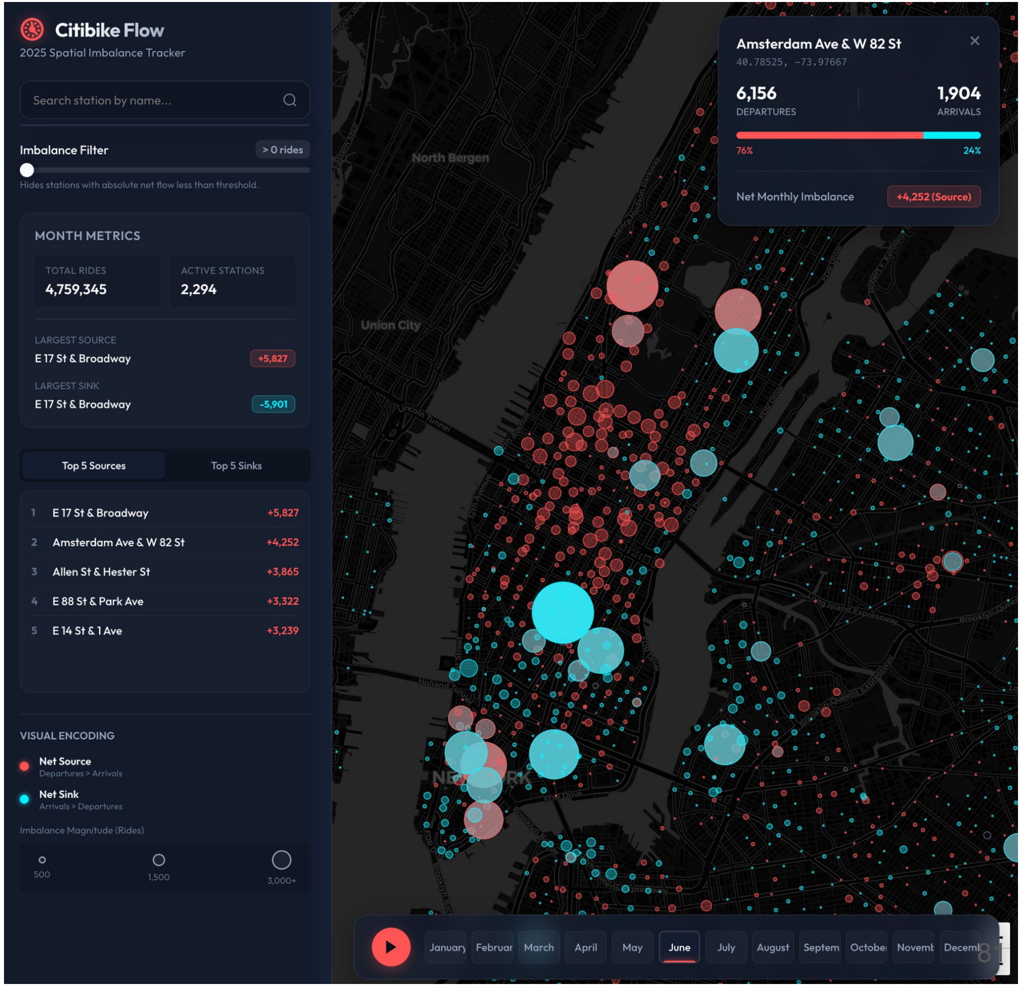




Visualization

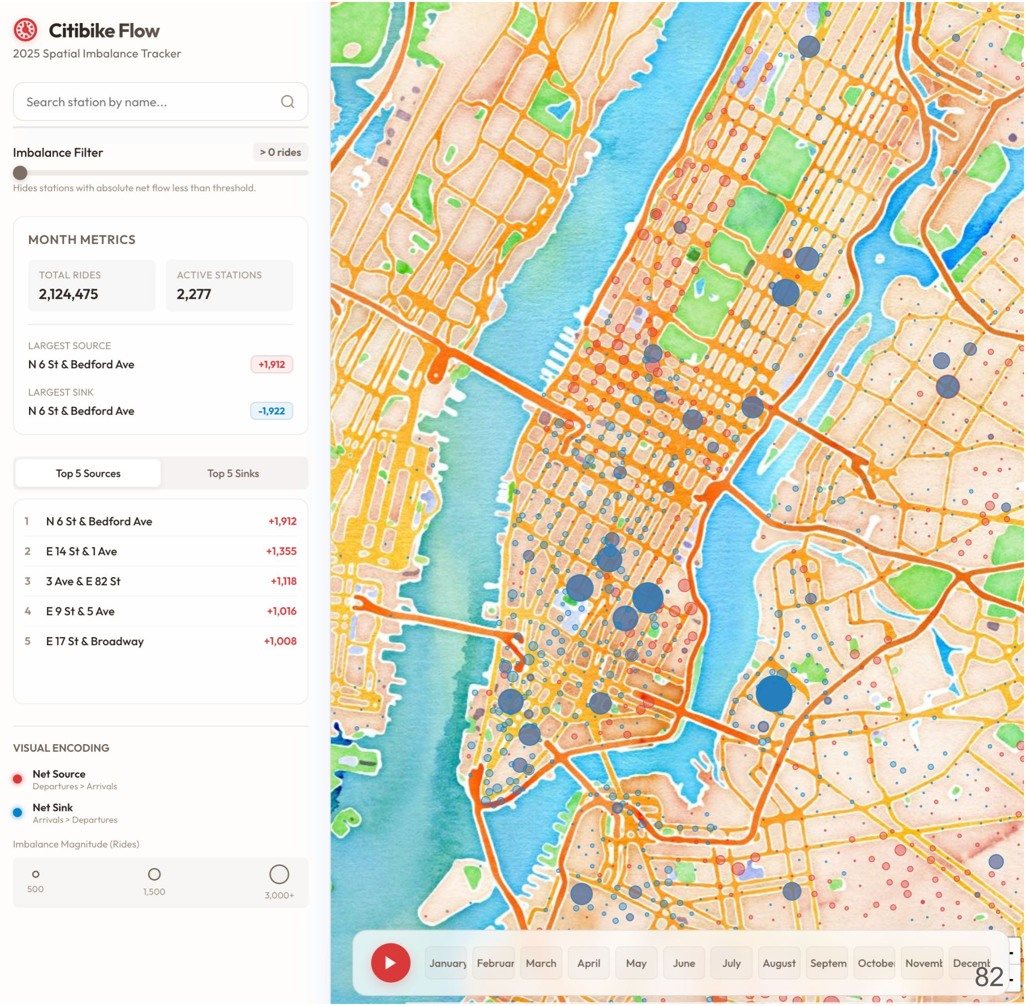
Different agents are dispatched to do different parts of the project: data processing, create HTML/Javascript, ...





Visualization

Change to a light color scheme with a hand-drawn map



Part 2c: Theorem Proving with Agentic AI

Brainstorm strategies

bilateral-trade

In my paper https://arxiv.org/abs/2602.12903 I left open the problem of designing a poly(d) regret for contextual bilateral-trade under one-bit feedback and per-period budget-balance. Please read the paper, understand the open problem and present to me a summary of the problem and three new ideas to attack it.

+ Gemini 3.1 Pro (High)

Local

Summary of the Open Problem

The Setting: You study a contextual bilateral trade problem where an online platform mediates between a seller and a buyer. At each time step t, the platform observes a d-dimensional context x_t. The seller and buyer have private valuations $s_t = \langle s, x_t \rangle$ and $b_t = \langle b, x_t \rangle$, parameterized by unknown d-dimensional vectors s and b. The platform posts a price p_t to the seller and q_t to the buyer.

The Constraints:

1. **One-bit feedback**: The platform only observes whether the trade occurs (i.e., both agents accept, meaning $s_t \leq p_t$ and $b_t \geq q_t$) or fails. It does not observe individual rejections.
2. **Per-period budget-balance**: The platform can never run a deficit on any round, meaning it must always post $p_t \leq q_t$.

The Issue: Under two-bit feedback (or one-bit feedback with small budget violations), you achieve $O(d \log d)$ regret by posting "safe prices" to disambiguate whose valuation caused a rejection. However, under strict one-bit feedback and per-period budget-balance, a "no-trade" outcome is inherently ambiguous. To circumvent this, your current algorithm relies *only* on successful trades to update its confidence regions. In the "strong overlap" case, it posts a single uniform random price $p_t = q_t$ over the union of the uncertainty intervals. Because the algorithm must guess blindly, the probability of hitting the "sweet spot" that guarantees a trade is very small, leading to a Steiner potential reduction of only $(1 - 4^{-d})$ upon a successful trade.

The Open Problem: This random sampling causes the regret to scale exponentially with the dimension $O(d6^d)$ for efficiency maximization, and $O(d6^d \log T)$ for profit maximization. The open problem is to design an algorithm that achieves $\mathrm{poly}(d)$ **regret** in this strict setting.

Three New Ideas to Attack the Problem

To achieve a $\mathrm{poly}(d)$ regret, an algorithm must extract information efficiently even from "no-trade" outcomes, or radically change how it probes the uncertainty space. Here are three novel ideas to explore:

1. Joint-Space Volume Cutting via Centroid/Ellipsoid Methods

Currently, your algorithm maintains independent confidence regions S_t and B_t in $\mathbb{R}^d$. A "no-trade" outcome gives the feedback: $\neg(\langle s, x_t \rangle \leq p_t \wedge \langle b, x_t \rangle \geq q_t)$. This translates to removing a convex wedge from the **joint parameter space** $\mathbb{S} \times \mathbb{B} \subseteq \mathbb{R}^{2d}$.

- **The Idea:** Instead of updating S_t and B_t independently (which requires unambiguous feedback), maintain a joint uncertainty set (or an enclosing volumetric barrier/ellipsoid) in $\mathbb{R}^{2d}$. By using cutting-plane methods (e.g., querying at the volumetric center or analytic center of the joint space), you can guarantee that *regardless* of whether a trade happens or fails, a constant fraction of the joint volume is eliminated.
- **Why it yields poly(d):** Standard geometric arguments (like Grunbaum's Theorem for centroids) guarantee that any cut through the centroid removes at least a $1/e$ fraction of the volume. This would ensure the joint volume shrinks by a constant factor every step, entirely bypassing the exponential waiting time for a successful uniform random probe.

2. Self-Funded Disambiguation (Dynamic Budget Buffers)

You established that violating the budget by an overall $O(d \log d)$ amount is sufficient to recover $\mathrm{poly}(d)$ regret via "safe prices" that disambiguate feedback.

- **The Idea:** Maintain strict per-period budget balance ($p_t \leq q_t$) most of the time, but maintain a running ledger of the **profit surplus** accumulated from the "well-separated" and "weak overlap" cases. In the problematic "strong overlap" case, instead of guessing uniformly, dynamically allow a local budget inversion ($p_t > q_t$) to act as a definitive disambiguating probe—but **strictly cap this inversion by the surplus profit generated in previous rounds**.
- **Why it yields poly(d):** As long as the algorithm only spends profit it has already materialized, the *actual* realized profit per round never dips below zero cumulatively, and the platform never requires external subsidies. This provides the algorithmic power of budget-violation (yielding $\mathrm{poly}(d)$ regret) while arguably satisfying the economic spirit of a self-sustaining, non-subsidized platform.

3. Optimism in the Face of Ambiguity with Relaxed Steiner Potentials

The core geometric bottleneck is that updating non-convex sets (resulting from the union of half-spaces in a "no-trade" event) breaks the Steiner potential analysis, which relies on

3. Optimism in the Face of Ambiguity with Relaxed Steiner Potentials

The core geometric bottleneck is that updating non-convex sets (resulting from the uni of half-spaces in a "no-trade" event) breaks the Steiner potential analysis, which relies
convex bodies.

- **The Idea:** Define a "Smoothed" or "Relaxed" Steiner potential over non-convex regio for example by measuring the Steiner volume of the **convex hull** of the surviving hypotheses, or by using log-concave sampling (like the hit-and-run algorithm) to maintain a probability measure over the surviving non-convex set.
- **The Execution:** Use an Optimism-in-Face-of-Uncertainty (OFU) approach. Pick the p $(\tilde{s}, \tilde{b})$ in the surviving set that maximizes the potential gain from trade, and post the budget-balanced price $p_t = q_t = \frac{\langle \tilde{s}, x_t \rangle + \langle \tilde{b}, x_t \rangle}{2}$. If the trade fails, we immediately invalida this highly optimistic region. Because the distribution over the version space is tracke probabilistically rather than with strict interval intersections, we can prove that the Gaussian/Steiner measure of the convex hull drops by a polynomial fraction $1/\mathrm{poly}(d$ every failure, rather than $(1 - 4^{-d})$.

Problem Solving Idea: Ralph Loop

Strategy #1: **Ralph Loop**: keep trying until you succeed

I think direction 3 "Optimism in the Face of Ambiguity with Relaxed Steiner Potentials" is promising. Please try to prove it. Follow those steps:
1) write an algorithm and proof
2) have a different agent verify the proof and write a review and either accept or reject it
3) if the proof is accepted, produce a publication-quality tex file
4) if the proof is rejected, go back to step 1 and using the feedback try to solve it again
Don't stop until you find a solution. Don't hesitate!

+ Gemini 3.1 Pro (High)

er https://arxiv.org/abs/2602.12903 I left open the problem of designing a
gret for contextual bilateral-trade under one-bit feedback and per-period
ılance. Please read the paper, understand the open problem and present to me
y of the problem and three new ideas to attack it.

ection 3 "Optimism in the Face of Ambiguity with Relaxed Steiner Potentials" is
. Please try to prove it. Follow those steps:
ı algorithm and proof
different agent verify the proof and write a review and either accept or reject it

Worked for 6m

I have written the initial proof for the "Optimism in the Face of Ambiguity with Relaxed Steiner Potentials" idea and launched an expert math subagent to review it rigorously, as requested. The subagent is currently verifying the geometric lemma regarding the drop in the first intrinsic volume and will provide feedback or a mathematical fix shortly. I will wait for its review and iterate until the proof is solid.

Proof Draft
Drafting the proof of poly(d) regret to send to the math expert subagent

1 subagent running
Math Reviewer

Ask anything, @ to mention, / for actions

+ Gemini 3.1 Pro (High)

Proof Draft

Proof of poly(d) Regret for Contextual Bilateral Trade

Algorithm: OFU with Relaxed Steiner Potentials

1. Initialize the convex version space $K_1 = \mathrm{B}_1^d \times \mathrm{B}_1^d \subset \mathrm{R}^{2d}$.
2. At each step t, receive context $x_t \in \mathrm{R}^d$.
3. Compute the optimistic bounds: $\bar{s}_t = \min_{(s,b)\in K_t} s^\top x_t$ $\bar{b}_t = \max_{(s,b)\in K_t} b^\top x_t$ Let $W_t = \bar{b}_t - \bar{s}_t$. If $W_t \le 0$, post $p_t = 1$ (no trade, regret 0).
4. Else, post the budget-balanced price $p_t = \frac{\bar{s}_t+\bar{b}_t}{2}$.
5. Observe feedback $y_t \in \{0,1\}$.
   - If $y_t = 1$ (trade occurs): $K_{t+1} = K_t \cap \{s^\top x_t \le p_t\} \cap \{b^\top x_t \ge p_t\}$.
   - If $y_t = 0$ (no trade occurs): $K_{t+1} = \mathrm{Conv}(K_t \setminus \{s^\top x_t \le p_t \wedge b^\top x_t \ge p_t\})$.

Regret Analysis: Let r_t be the regret. If $y_t = 1$, the platform gains $b^{*\top} x_t - s^{*\top} x_t$, matching the benchmark, so $r_t = 0$. If $y_t = 0$, the platform gains 0. The benchmark gain is at most W_t. Thus $r_t \le W_t$. We only suffer regret on no-trade steps.
Potential Function: We use the first intrinsic volume (mean width) $V_1(K_t)$ of the convex body K_t as the potential. $V_1(K_1) \le O(d)$.
Lemma: If $y_t = 0$, the mean width drops by a polynomial fraction of W_t, i.e., $V_1(K_t) - V_1(K_{t+1}) \ge \Omega(\frac{W_t}{d^{3/2}})$. *Proof Outline:* The corner removed, $C_t = \{s^\top x_t \le p_t \wedge b^\top x_t \ge p_t\}$, corresponds to a deep cut of depth $W_t/2$ relative to the bounding box of K_t in the projection on x_t. Because K_{t+1} is the convex hull of the remaining points, removing this corner truncates the extreme points in the joint direction, reducing the mean width by at least $\Omega(W_t/d^{3/2})$ by standard properties of intrinsic volumes of convex bodies.
Summing over all steps where $y_t = 0$, the total regret is $\sum W_t \le O(d^{3/2}) V_1(K_1) \le O(d^{5/2})$, which is $poly(d)$.

Problem Solving Idea: Ralph Loop

Review

Review of Proof Draft

Flaw in the Original Proof

The proof draft relies on the claim that removing the "no-trade" corner $C_t = \{s^\top x_t \le p_t \wedge b^\top x_t \ge p_t\}$ and taking the convex hull of the remaining set $R_t = K_t \setminus C_t$ strictly reduces the mean width V_1 by a polynomial factor.
This claim is **mathematically false**. The corner C_t is the intersection of the convex body with a quadrant. The boundary of this quadrant passes through the interior of K_t, and the extreme points of C_t lie exactly on the boundary of the cut. Because these extreme points are in the closure of R_t, taking the convex hull of R_t perfectly reconstructs them. **Counterexample:** Let K_t be the L_1 diamond $\{|u| + |v| \le 1\}$ and $p_t = 0$. The removed corner $C_t = \{u \le 0 \wedge v \ge 0\}$ is the top-left triangle with vertices $(0,0), (0,1), (-1,0)$. The remaining set R_t contains points arbitrarily close to $(0,1)$ and $(-1,0)$. Thus, taking the convex hull of $\overline{R_t}$ reconstructs the entire segment between $(-1,0)$ and $(0,1)$, restoring the exact triangle C_t. Therefore, $\mathrm{Conv}(K_t \setminus C_t) = K_t$, and the volume (and mean width) drop is exactly zero. The algorithm will get stuck infinitely proposing the same prices.
Conclusion: I reject the original proof.

Rigorous Mathematical Fix: Decoupled Bandit Ellipsoid Method

To achieve $\mathrm{poly}(d)$ regret, we abandon the single convex hull and decouple the learning of the seller and buyer functions using the budget balance slack. We maintain two separate ellipsoids E_s and E_b containing s^* and b^*.
At each round t, we compute the ranges $[\underline{s}_t, \bar{s}_t]$ and $[\underline{b}_t, \bar{b}_t]$ of $s^\top x_t$ and $b^\top x_t$ over E_s and E_b. Let $\hat{s}_t$ and $\hat{b}_t$ be the centers of these intervals (which correspond to the projections of the ellipsoid centers). Let $W_s = \bar{s}_t - \underline{s}_t$ and $W_b = \bar{b}_t - \underline{b}_t$ be the widths. We proceed with the following strategy:

1. **Central Cut for Seller:** If $\hat{s}_t \le \underline{b}_t$, we post $p_t = \hat{s}_t$ and $q_t = \underline{b}_t$. Budget balance $p_t \le q_t$ holds.
   - Feedback $y_t \in \{0,1\}$ provides a guaranteed central halfspace cut for s^* (either $s^\top x_t \le \hat{s}_t$ or $> \hat{s}_t$). We update E_s via the standard Ellipsoid method. Volume of E_s drops by a constant factor, and regret is bounded by 1.

I think direction 3 "Optimism in the Face of Ambiguity with Relaxed Steiner Potentials" is promising. Please try to prove it. Follow those steps:
1) write an algorithm and proof
2) have a different agent verify the proof and write a review and either accept or reject it

Worked for 6m ›

I have written the initial proof for the "Optimism in the Face of Ambiguity with Relaxed Steiner Potentials" idea and launched an expert math subagent to review it rigorously, as requested. The subagent is currently verifying the geometric lemma regarding the drop in the first intrinsic volume and will provide feedback or a mathematical fix shortly. I will wait for its review and iterate until the proof is solid.

Message from Math Reviewer (math_expert)

Worked for 2m ›

I have rewritten the proof incorporating the Decoupled Bandit Ellipsoid Method and corrected a subtle bug in the first review that would have allowed $O(T)$ regret by proving a new geometric lemma (the "empty gap" lemma). The subagent is currently verifying this revised proof. Once it confirms and formally accepts the proof, I will proceed to step 3 and generate the publication-quality LaTeX file.

Proof Draft
Drafting the proof of poly(d) regret to send to the math expert subagent

Proof Draft V2
Writing the revised proof for the math expert to review

1 subagent running
Math Reviewer

Proof Draft V2

Proof Draft V2

Decoupled Bandit Ellipsoid Method

Algorithm:

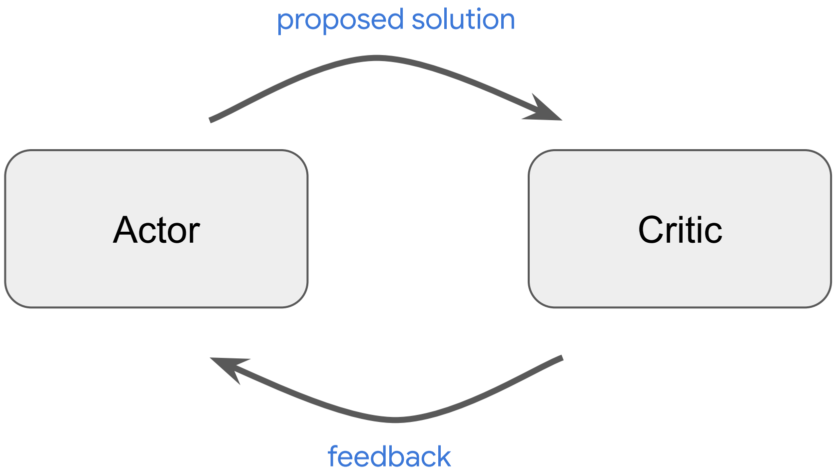
1. Maintain two ellipsoids E_s and E_b in R^d.
2. For each context x_t, compute the intervals $[\underline{s}_t, \bar{s}_t]$ and $[\underline{b}_t, \bar{b}_t]$ of $s^\top x_t$ and $b^\top x_t$ over E_s and E_b. Let $\hat{s}_t, \hat{b}_t$ be the centers and W_s, W_b be the widths.
3. **Case 1 (Central Cut for Seller):** If $\hat{s}_t \le \underline{b}_t$, post $p_t = \hat{s}_t, q_t = \underline{b}_t$. Budget balance holds ($p_t \le q_t$). q_t is safe for the buyer, so $y_t \in \{0,1\}$ provides a central halfspace cut for E_s.
4. **Case 2 (Central Cut for Buyer):** If $\bar{s}_t \le \hat{b}_t$, post $p_t = \bar{s}_t, q_t = \hat{b}_t$. p_t is safe for the seller, so $y_t \in \{0,1\}$ provides a central cut for E_b.
5. **Case 3 (Randomized Price):** If both fail, $\hat{s}_t > \underline{b}_t$ and $\bar{s}_t > \hat{b}_t$. Post $p_t = q_t \sim \mathrm{Unif}[\underline{s}_t, \bar{b}_t]$. If $y_t = 0$, do not update. If $y_t = 1$, update both E_s (with $s^\top x_t \le p_t$) and E_b (with $b^\top x_t \ge p_t$).

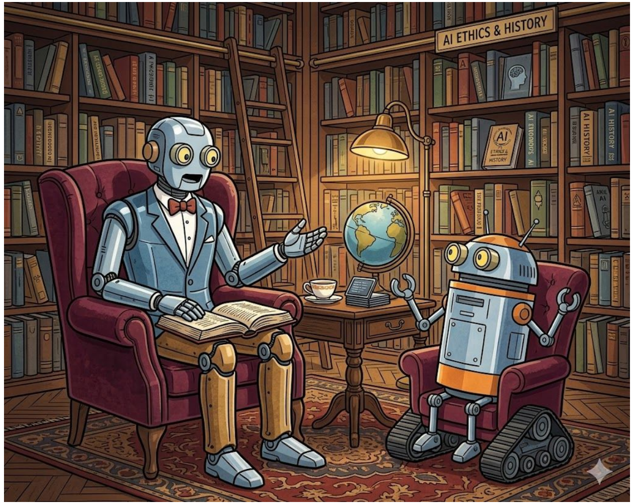
Proof of poly(d) Regret: In Cases 1 and 2, one ellipsoid is cut centrally, reducing its volume by a constant factor $1 - \Omega(1/d)$. Regret is at most 2. This can happen $O(d^2 \log(1/\epsilon))$ times. In Case 3, we prove that *any* trade guarantees a deep cut. **Lemma:** Every price p_t is a deep cut for E_s (depth $\ge W_s/4$) or for E_b (depth $\ge W_b/4$). *Proof:* Suppose p_t is shallow for both: $p_t > \hat{s}_t + W_s/4$ and $p_t < \hat{b}_t - W_b/4$. This implies $\hat{b}_t - \hat{s}_t > (W_s + W_b)/4$. However, failure of Cases 1 and 2 means $\hat{b}_t - \hat{s}_t < W_b/2$ and $\hat{b}_t - \hat{s}_t < W_s/2$. Thus $\hat{b}_t - \hat{s}_t < \min(W_s/2, W_b/2) \le (W_s + W_b)/4$. Contradiction. The gap of shallow cuts is empty.
Since every trade guarantees a deep cut of depth $\ge W/4$, whenever $y_t = 1$, the volume of at least one ellipsoid shrinks by $1 - \Omega(1/d)$. Let $\Phi = \log \mathrm{Vol}(E_s) + \log \mathrm{Vol}(E_b)$. Expected potential drop: $\mathrm{E}[\Delta\Phi] \ge \mathrm{P}(y_t = 1) \cdot \Omega(1/d)$. Trade probability: $\mathrm{P}(y_t = 1) = \frac{\Delta_t}{\bar{b}_t - \underline{s}_t} \ge \frac{\Delta_t}{W_s + W_b}$, where $\Delta_t = (b^* - s^*)_+$. Rearranging: $\Delta_t \le (W_s + W_b) \cdot O(d) \cdot \mathrm{E}[\Delta\Phi] \le O(d)\mathrm{E}[\Delta\Phi]$.
Summing over all Case 3 steps gives total expected regret $O(d \sum \mathrm{E}[\Delta\Phi]) = O(d^2 \log(1/\epsilon))$. Total regret is $poly(d)$ and independent of T.

LLM call patterns: Actor-Critic / Self Reflection

- Before: manually call the model to prove a statement, then call it to review the statement, then call it to fix the bugs, then call the verifiers ...

- Automate using: Actor-Critic loops / AI Debate / Self-Reflection





Other LLM call patterns

- **Parallelize & Aggregate**: Call the model n times and then ask to choose the best solution or choose majority voting if binary/categorical recommendation
- **Plan & Execute**: Ask the model to propose a plan and then make one call the execute each of the steps of the plan
- **Least-to-Most**: Ask the model to propose a simpler/approachable subproblem, then a call to solve it. Then ask the next approachable subproblem, then a call to solve it…
- **Primal-Dual**: Two calls asking the model to prove and disprove a statement. Then a call to a judge to decide which direction to take.
- **Routing**: Create different prompts to attempt to solve the problem via different techniques (e.g. geometry, algebra, analysis, probability, …) forcing the model to explore different directions. Then one call to get best possible directions.

Problem Solving Idea: Primal Dual

For the theorem below, start two separate agents, one trying to prove and one trying to disprove it. Each of them should produce a tex file with the conclusions. Then have a third agent to evaluate both and generate a final report.

Both the Prover and Disprover have finished their analyses!

- The **Prover** submitted a rigorous proof using Polya's and Handelman's Positivstellensatz, concluding the theorem is true.
- The **Disprover** submitted a counterexample for $n \geq 2$, arguing there is an algebraic contradiction involving the lowest degree homogeneous part of the denominator polynomial, concluding the theorem is false.

I have just launched the **Evaluator** subagent. It is currently reading both `.tex` files. It will evaluate the merits and flaws of both arguments and produce a `final_report.tex` with a definitive verdict.

I'll let you know as soon as the final report is ready!

The Final Report:

The comprehensive final evaluation report has been written and saved as a LaTeX file. You can view the final verdict here: `final_report.tex`

You can also check the individual conclusions from the Prover and Disprover:

- **Prover**: `proof_conclusions.tex`

For the theorem below, start two separate agents, one trying to prove and one trying to disprove it. Each of them should produce a tex file with the conclusions. Then have a third agent to evalute both and generate a final report.

```
Let $\mathbf{x} = (x_1, \dots, x_n) \in (0, 1)^n$. A \textbf{non-negative Bernstein monomial} is a function of the form:
\[
M_{\mathbf{b}, \mathbf{c}}(\mathbf{x}) = a \prod_{i=1}^n x_i^{b_i} (1-x_i)^{c_i}
\]
```

Worked for 47s

Explored

Thought for 19s

Defined math_prover subagent

Defined math_disprover subagent

Defined math_evaluator subagent

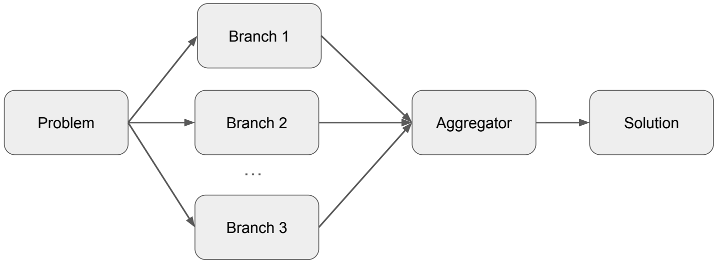
Thought for 10s

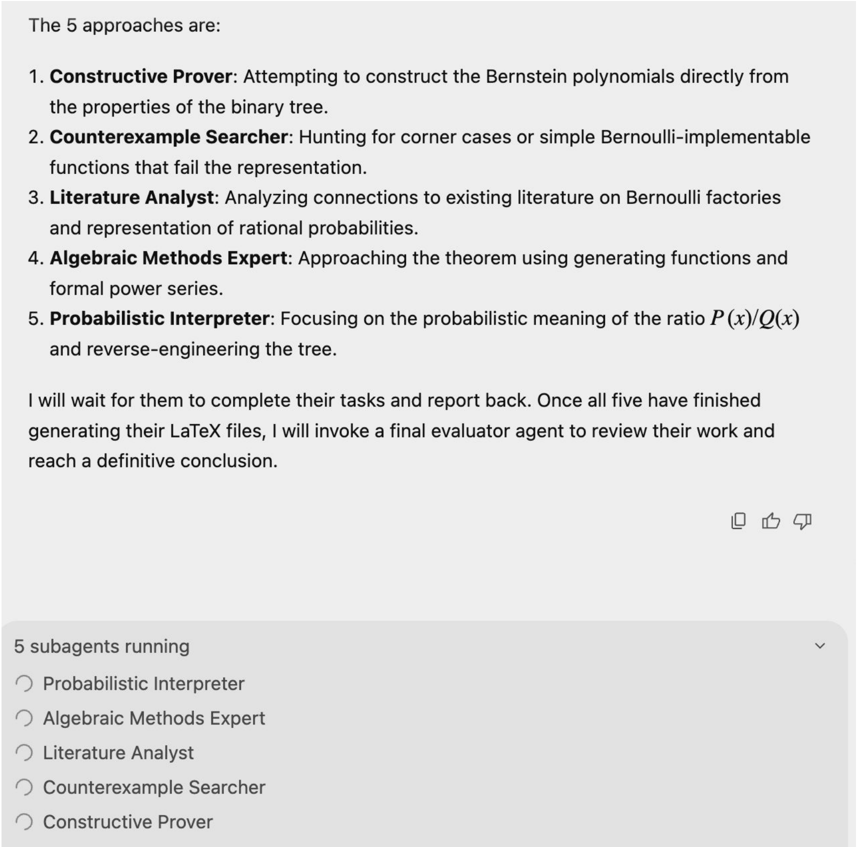
Theorem Prover
Invoked math_prover subagent

Theorem Disprover
Invoked math_disprover subagent

Problem Solving Idea: Parallelize and Aggregate

For the theorem below, start five separate subagents attacking the problem from different directions.. Each of them should produce a tex file with the conclusions. Then have another agent evaluating them and reaching a final conclusion.





The 5 approaches are:

1. **Constructive Prover**: Attempting to construct the Bernstein polynomials directly from the properties of the binary tree.
2. **Counterexample Searcher**: Hunting for corner cases or simple Bernoulli-implementable functions that fail the representation.
3. **Literature Analyst**: Analyzing connections to existing literature on Bernoulli factories and representation of rational probabilities.
4. **Algebraic Methods Expert**: Approaching the theorem using generating functions and formal power series.
5. **Probabilistic Interpreter**: Focusing on the probabilistic meaning of the ratio $P(x)/Q(x)$ and reverse-engineering the tree.

I will wait for them to complete their tasks and report back. Once all five have finished generating their LaTeX files, I will invoke a final evaluator agent to review their work and reach a definitive conclusion.

5 subagents running

- Probabilistic Interpreter
- Algebraic Methods Expert
- Literature Analyst
- Counterexample Searcher
- Constructive Prover

Encouraging the model to think longer

- You can encourage the model to think longer:
    - “`Don’t stop until you find a proof or a counter-example.`”
    - “`Provide an exhaustive response. Do not rush to a conclusion.`”
    - “`Take your time and explore all nuances ….`”
    - “`Don’t hesitate.`”

- With those, you need to be more mindful of token usage.

Create your own skills, workflows and agents

- **Create skills / workflows**: After a non-trivial operation that will be useful in the future, you can ask Antigravity to create a skill / workflow and save it somewhere
  - Modern version of keeping a prompt library
  - Example: Antigravity Arvix Skill

- **Create your own agents / orchestration patterns:** You can brainstorm with Antigravity how to create the best pattern / orchestration to solve your problem. In general, take the time to discuss with the model:
  - What is the right prompt to use
  - Iterate on a plan
  - Decide what is the right orchestration strategy

- Then start... For difficult tasks, if well specified, Antigravity can run for hours autonomously.

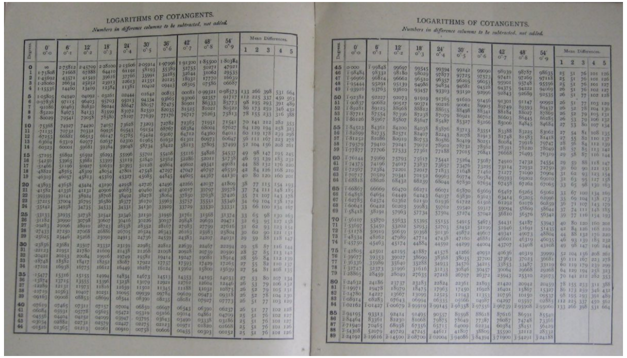
Final thoughts

- Different tasks require levels of “**human in the loop**”
  - **One end:** read all the reports and interact with AI as your collaborator
  - **In between:** advisor-student relationship, where AI runs autonomously for some time but you inspect and steer it from time to time
  - **Other end:** specify the problem and let the AI run for hours / days / …

- For more complex tasks, you may want to try an agentic IDE like Antigravity instead.

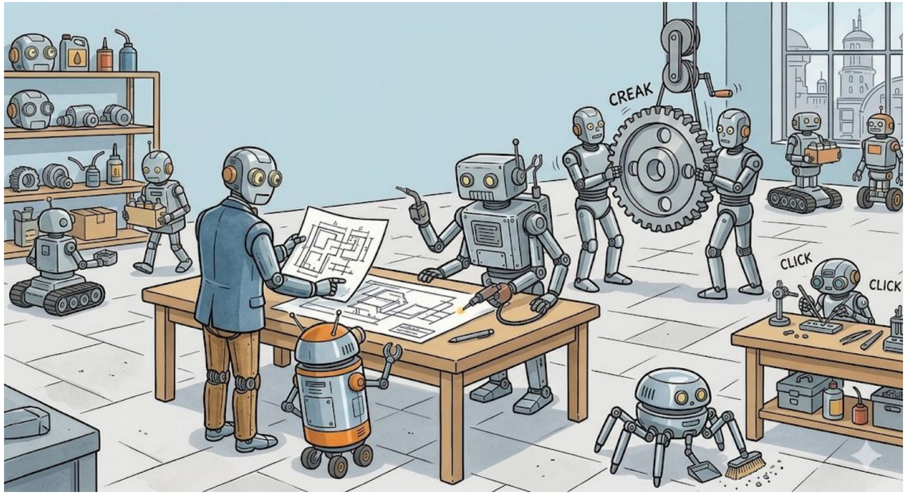
- Human ingenuity is still there, but we may be using different workflows

- In the past, a lot of engineer’s time was spent doing calculations and computers freed up time for us to spend in more complex tasks. We hope AI will help focus on creativity on tackling more difficult problems.



Final thoughts

- Don't assume the systems can't do something. Models are getting better very fast.
- But getting what you want may require many iterations (back-and-forth, corrections, restarts, novel agentic loops) → new kinds of difficulty
- Always check the work!
- How you ask matters a lot. Try different prompt variations and ask Gemini to refine prompt ideas. Try different systems.
- How the research community should evolve in the presence of AI is a big open question.



Thanks!

Feedback to {duetting, renatoppl, sbalseiro}@google.com
Slides are available at renatoppl.com/slides/ai-tutorial-columbia.pdf