Draft version September 23, 2026
Typeset using LaTeX **preprint** style in AASTeX631

ALMA Memo 631

Exploring LLMs as Tools for Testing and Developing ALMA Notebooks

Zaina Tarafder,[1,*] Daniel Western,[1,*] Adele Plunkett,[2] and Paul Torrey[1]

[1]*Department of Astronomy, University of Virginia, Charlottesville, VA 22903, USA*
[2]*National Radio Astronomy Observatory, Charlottesville, VA 22903, USA*

ABSTRACT

The steady increase of data volume and content available through major data repositories like the Atacama Large Millimeter/submillimeter Array (ALMA) Science Archive and NOIRLab's Astro Data Lab presents a unique opportunity to evaluate the role of Large Language Models (LLMs) as scientific copilots for data accessibility. While LLMs show promise in coding and text generation, their capabilities in multi-step reasoning, error resolution, and scientific generalization within code-based research workflows are variable due to their probabilistic nature and sensitivity to context framing. This study employs Gemini 2.5 Pro in an iterative, multi-stage methodology that uses the ALMA Science Archive (ASA) Jupyter notebooks as the primary information corpus, with ChatGPT (GPT-4o) mostly being used as support for creating effective prompts. Our approach benchmarks the models' ability to interpret code structure, generalize tasks to novel problems (re-creation), and execute complex multi-step analyses. We describe a cross-model prompting experiment to develop preliminary guidelines for effective AI-assisted research. Our findings reveal that LLMs, under structured guidance, show potential to act as functional co-developers: they generate novel notebooks for tasks like cross-referencing ALMA and JWST data, accurately produce astronomical data visualizations, and independently integrate robust features such as retry logic and archive mirror failovers. However, early challenges include significant failure rates due to context overload and poor multi-step reasoning. We conclude that effective LLM-assisted science is contingent upon active human supervision at inference time, requiring three critical user traits: contextual knowledge, specificity in prompting, and rigorous scrutiny of all generated outputs against scientific ground truth. This work establishes a foundation for continuously testing and improving public observatory data resources using AI-assisted workflows.

1. INTRODUCTION

The development and advancement of large language models (LLMs) such as OpenAI's ChatGPT and Google Gemini have ushered in new ways to explore, test, and develop standardized computational notebooks for conducting astronomical research. These models can serve as "copilots" to assist researchers in writing scripts for querying online databases, automating data reduction, and creating output figures. Recent benchmarking work shows that state-of-the-art LLMs are achieving near-top human performance on conceptual astronomy exams while still exhibiting notable gaps in multi-step reasoning and geometric/spatial visualization tasks (Carrit Delgado Pinheiro et al. 2025). At the

* Primary co-authors.

same time, some recent frameworks demonstrate how LLMs can be utilized to generate research ideas, write and execute code, visualize results, and draft manuscripts — suggesting feasible paths toward deeper use of LLM-based assistants in the scientific process (Lu et al. 2024). The integration of these tools into research workflows has the potential to lower barriers to accessing and correctly using complex, publicly available astronomical datasets, thereby supporting more reproducible analysis workflows.

With the rise in artificial intelligence (AI) usage, recent literature has explored the application of LLMs and AI systems with respect to specific scientific research tasks. In the PaperBench benchmark, Starace et al. (2025) demonstrated that modern AI agents can partially replicate complex machine learning research papers—reading, coding, executing, and evaluating experiments from scratch; however, it still falls short of expert human performance. This benchmark introduced hierarchical rubrics for measuring scientific replication, providing a framework to assess models' capabilities in executing long-horizon. Despite their ability to devise complex, multi-step plans, these models frequently fail to realize them in practice, reinforcing the necessity of human judgment and supervision.

In astronomy, efforts to evaluate LLMs on domain-specific reasoning tasks are beginning to emerge. The *AstroVisBench* project (Joseph et al. 2025) offers a large-scale attempt to measure how AI models interpret and analyze astronomical visualizations such as color–magnitude diagrams and galaxy survey plots. Their findings show that while modern models like GPT-4o and Gemini can recognize general patterns, they often misinterpret quantitative relationships or fail to connect visual trends with underlying astrophysical concepts.

The field of astronomy is abundant with reproducible analysis environments, particularly through the widespread use of Jupyter notebooks created by observatories and data archives such as the Atacama Large Millimeter/submillimeter Array (ALMA), the NSF National Optical-Infrared Astronomy Laboratory (NSF NOIRLab), and the Space Telescope Science Institute (STScI). These notebooks are publicly accessible and offer researchers a standardized framework for retrieving, calibrating, and visualizing data. As a result, the field of observational astronomy is a particularly promising one for exploring and evaluating techniques for harnessing AI in scientific research. Note that, while this study finds that maintaining these observatory-provided notebooks for compatibility across different and evolving software versions and Python environments presents a persistent, inherently difficult challenge, archive maintenance was seldom the source of error. This challenge, however, motivates the exploration of LLM adaptability with existing notebooks to different environments and use cases.

Situating our work within this context, we examine whether LLMs can engage not just with astronomical images, but with the computational processes that underpin modern research using ALMA notebooks. Using LLMs such as Gemini, we are able to generate several Jupyter notebooks that specifically address use cases outlined by the ALMA Science Archive (ASA) Working Group. Our findings illustrate both the potential and the current limitations of LLMs in supporting reproducible, code-based research, while also laying the groundwork for the development of AI-assisted notebooks that can tackle similar tasks to those demonstrated for the ASA.

2. METHODS

2.1. *Benchmark Construction*

To evaluate the capability of LLMs in scientific notebook execution and development, we adopted a benchmarking framework informed by the *AstroVisBench* methodology (Joseph et al. 2025). Specif-

ically, we applied its principle of comparing model-generated outputs against clearly defined ground truths, allowing for qualitative evaluation of how closely the results produced by Gemini and ChatGPT's prompting aligned with expected scientific outcomes. This benchmark approach further focuses on whether LLMs can reason through multi-step astronomical analyses, execute code accurately, and reproduce figures or data products consistent with reference notebooks. By adapting *AstroVisBench*'s human–AI comparison framework to computational notebooks, we created a practical means of assessing LLM performance in a context directly relevant to reproducible astronomy.

2.2. *Notebook Sourcing and Construction*

The corpus of notebooks used for conditioning Gemini was drawn from public repositories pertaining to major astronomical data archives and related software, which are listed and described here.

- **ASA** — ALMA provides a suite of ten Python-based Jupyter notebooks[1] that serve as step-by-step guides for programmatically querying metadata from the ASA[2], as well as selecting and downloading data products of interest. In total, ten notebooks were available at the time of this project; the notebooks are composed of cells of Python code, as well as Markdown scripts and comments that describe the code.
- **NOIRLab Astro Data Lab** — The Astro Data Lab science platform (Nikutta et al. 2020) houses a collection of Jupyter notebooks developed for optical and infrared survey data exploration, primarily supporting datasets from the Dark Energy Survey (DES), DESI Legacy Survey (LS), and other wide-field imaging programs. These notebooks demonstrate practical workflows for querying large public databases, performing cross-matching, creating image cutouts, and generating other visualizations (Juneau et al. 2021). They also emphasize reproducible data access via Python-based interfaces such as `astroquery` and Astro Data Lab's `queryClient`, serving as accessible templates for users analyzing NOIRLab's and the Data Lab's extensive survey archives. All of the notebooks used here were obtained from https://datalab.noirlab.edu/docs/manual/Appendices/Notebooks/Notebooks.html.
- **STScI Archive** — The Space Telescope Science Institute hosts repositories of Jupyter notebooks demonstrating use cases related to Hubble Space Telescope (HST), James Webb Space Telescope (JWST), and Mikulski Archive for Space Telescope (MAST) data. These notebooks exemplify high-level data querying, photometric catalog analysis, and FITS file visualization for several space telescope data sets.
- **CASA** – The online user guide for the Common Astronomy Software Applications (CASA) CASA Team et al. (2022)[3], which is the principal data processing software for ALMA, the Very Large Array (VLA), and other radio telescopes, incorporates Jupyter notebooks that be opened in Colab. These Jupyter notebooks include documentation and CASA commands on topics including general use cases, radio astronomy fundamentals, calibration, and imaging.

We examined these notebooks to understand their structure, data access methods, and levels of documentation. ASA was selected as the focus of this study due to the consistent formatting and

[1] https://almascience.org/alma-data/archive/archive-notebooks
[2] https://almascience.org/aq
[3] https://casadocs.readthedocs.io/

manageable scope of the Jupyter notebooks, as well as ample publicly accessible data products. Furthermore, the contribution of additional notebooks for accessing ASA data was determined to be a valuable potential outcome of our work. After reviewing the ten aforementioned ALMA notebooks, we identified a representative subset covering querying based on certain parameters and subsequent data analysis. These notebooks were then assessed for compatibility with LLM prompting, ensuring cell order and dependencies were maintained for accurate sequential execution. The other notebooks mentioned above are considered to be viable avenues for future applications of the methods we describe here.

2.3. *Iterative Evaluation With LLMs and Jupyter Notebook Development*

To investigate the effectiveness of LLMs in executing and developing scientific notebooks, we conducted interactive sessions primarily with **Google Gemini 2.5 Pro**[4] and, as a supplementary prompting model (see Figure 5), **OpenAI's ChatGPT (GPT-4o)**[5] between June and August 2025. It is important to note that the majority of these iterative processes were completed on Gemini due to its free student access to the 2.5 Pro version and its code-friendly user interface, and the iterative processes used are not the same as "training" in a machine learning sense. The models operated under guided prompting in the following four stages, which are also captured visually in Figure 1.

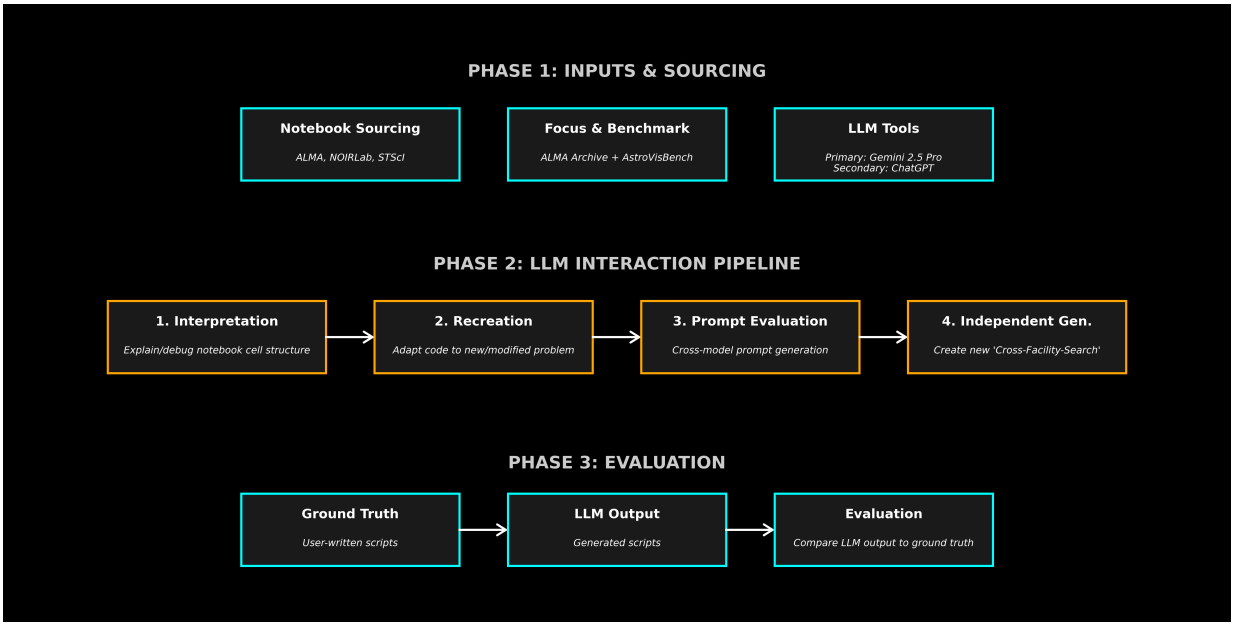


Figure 1. This flow chart captures the workflow of the LLM iterative evaluation and notebook development.

1. **Code Interpretation and Debugging** — Models were first asked to interpret the purpose and structure of each notebook cell, identify necessary packages, and explain how individual steps contributed to the overall analysis pipeline.

[4] https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-pro
[5] https://openai.com/index/hello-gpt-4o/

2. **Re-creation** — Rather than simply reproducing existing notebooks verbatim, we asked each model to *re-create* a selected notebook task by applying similar syntax, structure, and methodology to a new or modified problem.

   For example, a model might be shown an ALMA imaging notebook that queries a given dataset and then instructed to perform an analogous operation on a different target or frequency band while maintaining the same code logic and library use.

   This approach tested whether LLMs could generalize from provided examples to novel but related tasks, a key indicator of scientific adaptability.

3. **Prompt Evaluation** — To further investigate how LLMs interpret and respond to prompts, we implemented a two-stage "cross-model prompting" experiment:

   In the first stage, one model (e.g., ChatGPT) was asked to propose the *ideal prompt* that a researcher should use to recreate the given notebook efficiently (Appendix Fig. 5).

   In the second stage, that generated prompt was supplied to another model (e.g., Gemini) to evaluate how well it executed the task when prompted in this optimized manner.

   By comparing the performance of these cross-model pairs, we identified patterns in how LLMs conceptualize effective scientific instructions. This process informed a preliminary set of **prompt design guidelines** for future LLM-assisted research notebooks, emphasizing clarity, contextual guidance, and explicit task framing.

4. **Independent Notebook Generation** — Once familiar with the structure and logic of existing notebooks, each model was instructed to create a similar Jupyter Notebook, inspired by the ALMA examples, capable of tasks such as a "Cross-Facility-Search" notebook that can query, plot, and compare available datasets across varying public archives.

For data retrieval, both models employed Python-based querying through `astroquery.alma` and manual URL-based archive access, depending on their interpretation of the prompt (Ginsburg et al. 2019). Output analysis included evaluating generated figures, comparing numerical outputs to known reference values, and inspecting whether data parsing routines correctly handled metadata trees or hierarchical data structures.

All model interactions and outputs were recorded to enable direct comparison between the ground truth notebook results and LLM-generated outputs.

3. RESULTS

3.1. *Developing Notebooks with LLM Assistance*

Our experimentation with Google's Gemini 2.5 Pro model directly led to the development of example notebooks, which can be found in a repository at https://github.com/pre9mk/2025-Summer-CosmicAI-Notebooks/tree/main. Each notebook was developed using Gemini as a copilot, meaning the code was written initially by a human and later improved upon by the LLM. For the LLM to make these improvements, we provided it with the ten ALMA notebooks, as described in §2.2, and then followed the procedure described in §2.3.

3.1.1. *Challenges*

Initial testing of giving Gemini the ALMA notebooks exposed some weaknesses of the LLM when conducting research. Our first tests involved providing all ten notebooks at the same time, then asking Gemini to write code for a specific task (with some slight changes) chosen from any of the notebooks. Outputs from the LLM typically failed to accomplish the given task, and it would often cite an incorrect notebook or ignore the notebooks altogether when writing code. For example, after providing the notebooks and asking Gemini to access an observation of NGC 4874 from the ASA, it tried to completely write its own code rather than consulting any of the given notebooks. However, after we explicitly stated to refer to the notebooks, the LLM did so and returned mostly functional code, but sometimes missed the task and did something different instead. We believe this was due to an overload of context, since the AI would often "hallucinate" and get lost in the notebooks we were providing. The same issue would sometimes arise when providing each notebook individually and would get worse as more context was added. We found that after providing around four or five notebooks individually, Gemini would begin to hallucinate and return consistently incorrect results, so a new conversation would have to be started as a "reset." Getting Gemini to backtrack or ignore context once it started hallucinating proved to be difficult, so starting new conversations from scratch was necessary to continue experimentation. This also led us to give Gemini "validation" for correct results (i.e. telling the LLM we liked the output), which helped prevent hallucinations when adding more context. In addition to this, there were numerous small errors, such as using a search radius that was too small or using the wrong name for a variable. However, Gemini did note that the notebook we provided for this task was based on an old version of astroquery, which led to some errors when converting the Right Ascension (RA) and Declination (Dec) coordinates into decimal degrees.

3.1.2. *Successes*

We found success with Gemini by providing each ALMA notebook individually, then asking the LLM to read through the given notebook and return a summary of its contents. After giving it five of the ten notebooks, we conducted a "midterm" evaluation in which we asked Gemini to complete any two random tasks from the five notebooks we provided. We also gave it two tasks that we chose ourselves to make sure it understood the entirety of each notebook, not just one or two tasks from each. This midterm would be scored using a point-deduction system starting at 100 points with a "target score" of 80 points. We deducted five points for each critical error (code cannot compile, AI completes incorrect task, etc.), and 2.5 points for each minor error (mistakenly using an incorrect variable name, the code functions but returns 0 objects, etc.). Each task was worth 25 points out of 100, and we progressively deducted more points per error and increased the target score across multiple midterms until Gemini could complete the various tasks with minimal errors. These measures, as well as providing detailed feedback, consistently gave Gemini a better "understanding" of the necessary functions, imports, and general format of the notebooks, which led to fewer errors and confusion with context. Using the same prompt also yielded consistent results when less context was provided, though giving more context made Gemini's responses to the same prompt more variable across multiple trials. It should be clarified that while reliable consistency was produced with this method, scoring was done by a single reviewer and could vary between different individuals. This more rigid learning structure also caused the LLM to start making its own additions and improvements

to the code it wrote. Note that the purpose of this portion of experimentation was more so to gain a better understanding of Gemini's ability to handle context with the risk of hallucinations rather than develop a strict prompting regimen.

3.2. *LLM Additions*

After appropriately acclimating the LLMs to the form of typical ALMA notebooks (i.e. syntax, comment structure, necessary functions, etc.), they were able to produce notebooks that accurately completed various tasks, such as cross-referencing the results from querying both the ALMA and JWST data archives while also including improvements of their own. Figure 2 shows how Google Gemini included improvements to the code for cross-referencing observations of an object between the ALMA and JWST archives. The top box was suggested as a workaround for network issues when connecting to the North American ALMA archive, and completely bypasses the problem by using the European ALMA archive mirror. The bottom box was automatically added by the LLM to accommodate any connection errors/timeouts, but in the form of retry logic if the query failed at any point. More specifically, if there is an issue obtaining data from the archive, the script will try to connect multiple times, progressively waiting longer and longer after each attempt to account for factors such as server latency.

3.3. *Necessary User Traits*

Throughout the process of testing and building notebooks, we found that in the process of LLM-assisted science, it is important that the following traits are exemplified by the user.

3.3.1. *Contextual Knowledge*

While LLMs can write their own code and add improvements to existing code, it is crucial to check the accuracy of the work. We decided on three specific traits a user must have to effectively use LLMs as copilots when doing research: Contextual knowledge, specificity, and scrutiny. The first trait (contextual knowledge), indicates that the user must possess enough knowledge in the subject to provide the LLM with accurate information and descriptive prompts. For example, someone attempting to complete the cross-referencing task with no knowledge of astronomy will not know the necessary parameters for plotting the observations, like how the plots being generated will show positions of observations in right ascensions and declinations.

3.3.2. *Specificity*

The second trait (specificity), indicates that initial user prompts and the subsequent conversation with the LLM must be incredibly detailed to ensure exact, ideal results. For the cross-referencing script, the prompts specified that any plots generated should use right ascension and declination as the units for either axis, since, in this case, we are interested in the positions of each observation made and how close they are to the official, measured positions from NASA. "Specificity" also applies to any functions being used or necessary imports, and the LLM must be made aware of clear setup and import instructions, as well as how to use !pip to install required modules and packages.

```python
def query_alma(target_name, radius_arcsec, retries=3):
    print(f"\nQuerying ALMA archive for '{target_name}'...")
    try:
        # First, resolve the target name into coordinates for the cone search
        target_coords = SkyCoord.from_name(target_name)
        print(f"Resolved '{target_name}' to coordinates: {target_coords.to_string('hmsdms')}")

        # Get an instance of the Alma tool
        alma = Alma()

        # Switch to the European (ESO) mirror to bypass server errors
        print("Using the ESO (European) ALMA archive mirror.")
        alma.archive_mirror = 'ESO'

        # --- Retry logic for network errors ---
        for attempt in range(retries):
            try:
                # Perform a cone search using query_region with the resolved coordinates
                search_radius = radius_arcsec * u.arcsec
                alma_obs = alma.query_region(target_coords, radius=search_radius, public=True)

                if alma_obs and len(alma_obs) > 0:
                    print(f"\nFound {len(alma_obs)} public ALMA observations.")
                    return alma_obs
                else:
                    print("\nNo public ALMA observations found for this target in the specified radius.")
                    return None

            except (URLError, TimeoutError, ConnectionResetError) as e:
                print(f"Network error on attempt {attempt + 1}: {e}")
                if attempt < retries - 1:
                    wait_time = 5 * (attempt + 1) # Wait longer each time
                    print(f"Waiting {wait_time} seconds before retrying...")
                    time.sleep(wait_time)
                else:
                    print("Max retries reached. ALMA query failed.")
                    raise e # Re-raise the last exception

        return None # Should only be reached if all retries fail without raising

    except Exception as e:
        print(f"An unrecoverable error occurred during the ALMA query: {e}")
        return None
```

Figure 2. This figure demonstrates additions that the LLMs included on their own when asked to cross-reference ALMA and JWST results for an object. Switching to the European Southern Observatory (ESO) ALMA archive mirror and the retry logic are both additions by the LLM.

3.3.3. *Scrutiny*

The third trait (scrutiny) indicates that the user must be able to scrutinize the LLM's outputs and notice any errors or inaccuracies in both the script and the generated results. In Figure 3, we see that there are numerous scattered observations of M42 within a small area, but at this stage, it is unknown if the positions are accurate. One method of scrutinizing LLM outputs is to look at any known information that can be used as a "ground-truth." For example, according to NASA, the central coordinates of M42 are $05h35m17.0s$ RA and $-5°23'27.99"$ Dec. In decimal degrees,

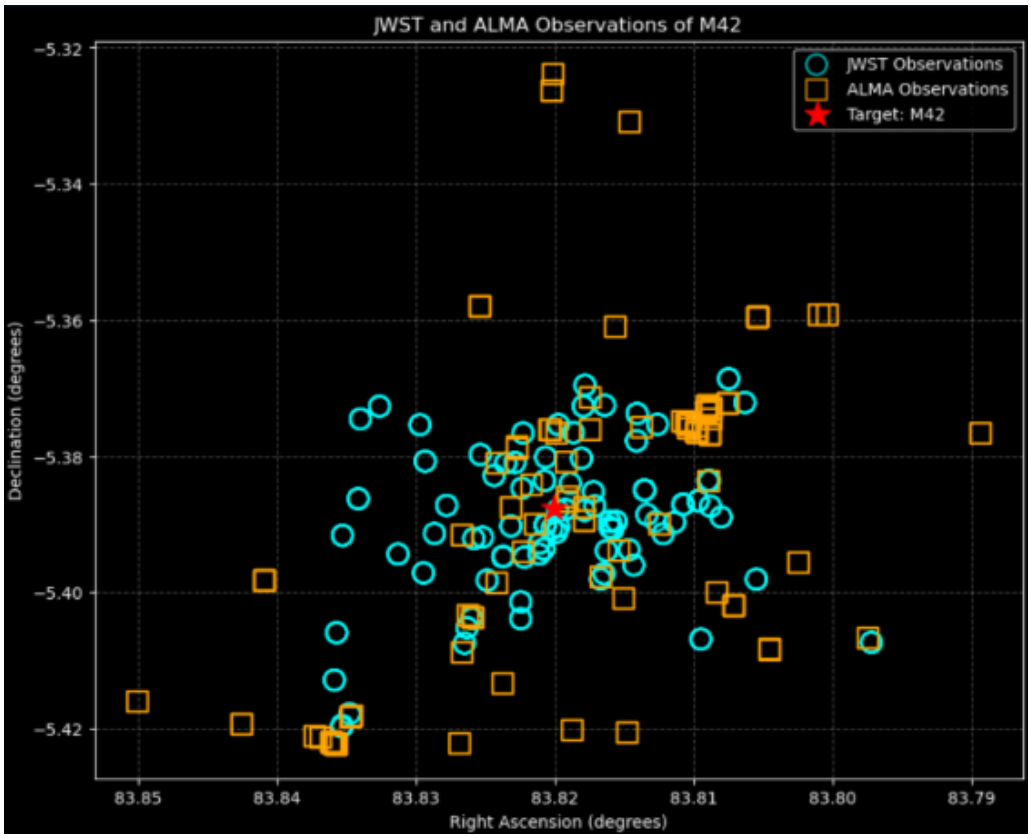


Figure 3. A plot generated by the LLM-assisted script showing observations of M42 from both the ALMA and JWST archives in right ascension and declination (in degrees). This output was validated by checking official NASA coordinates for M42, which coincided with those produced by Gemini.

this would be 83.82° RA and −5.391° Dec, which is what we see as the central coordinates of M42 on the generated plot. As another example, we look at Figure 4. The upper panel shows a plot generated from an entirely user-written script, while the lower panel was generated using an LLM-written script. We can see that both plots are identical by looking at the central coordinates of M33 and the positions of other observations. Since the LLM output matches the ground truth output, we know that the LLM has accurately scripted and plotted the desired task. It is best to use multiple methods of scrutiny, like the two examples above, in conjunction with each other for any single task to ensure the LLM-generated output is accurate.

4. CONCLUSIONS AND APPLICATIONS

In combining these methods, our study positioned LLMs not simply as coding assistants but as experimental agents for validating and developing reproducible astronomical workflows. By integrating a qualitative benchmarking approach, multi-source notebook analysis, and iterative LLM-notebook co-development, we established a foundation for assessing how AI systems might one day participate in the continuous testing and improvement of public observatory data resources. LLMs are useful tools as copilots, and with good data in the corpus of information (context), can accurately and efficiently obtain data, write entire scripts, troubleshoot errors, and generate complex plots to act as a "visualization assistant." However, limitations like hallucinations after taking in too much information should be noted when working with LLMs, and two to three notebooks per prompt will

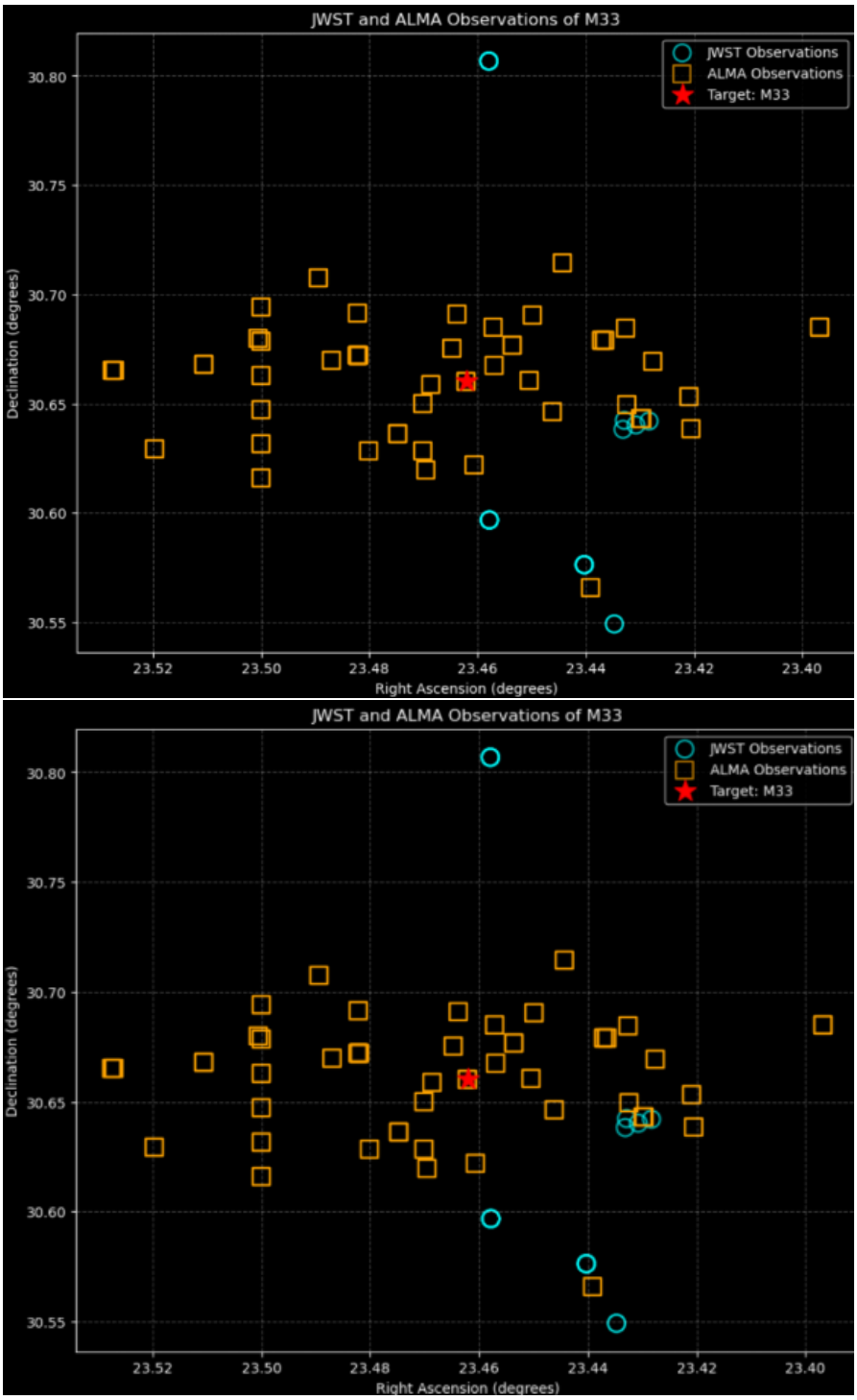


Figure 4. (upper) Ground truth output plot for observations of M33, stemming from a script entirely written by the user and validated with official values from NASA. (lower) The AI-generated output plot for observations of M33. We see that both plots, even though one was generated by us (ground truth) and one was generated by Gemini, appear to be identical. This indicates that the LLM accurately reproduced our plot using only the given context (notebooks).

reasonably avoid this most of the time. Future research would include finding ways to further reduce hallucinations and getting lost in context when completing scientific tasks, guiding an LLM to complete tasks with minimal user intervention, expanding into API based calls, automating the process to perform refined prompt engineering, and developing a more rigid, quantitative benchmark for the evaluation of LLMs. Depending on the needs of ASA and other astronomical archives, additional notebooks can be efficiently generated following the methodology we present here.

ACKNOWLEDGMENTS

We would like to extend our thanks to the CosmicAI Institute, funded by the National Science Foundation under Cooperative Agreement 2421782 and the Simons Foundation award MPS-AI-00010515. The authors would also like to thank Stephanie Juneau for helpful feedback on the memo. *This paper makes use of the ALMA Science Archive. ALMA is a partnership of ESO(representing its member states), NSF (USA) and NINS (Japan), together with NRC (Canada), NSTC and ASIAA (Taiwan), and KASI (Republic of Korea), in cooperation with the Republic of Chile. The Joint ALMA Observatory is operated by ESO, AUI/NRAO and NAOJ. The National Radio Astronomy Observatory is a facility of the National Science Foundation operated under cooperative agreement by Associated Universities, Inc.*

APPENDIX



This notebook should allow a user to query the ALMA science archive for data related to a **single astronomical source**, using **three methods**:

🔍 **1. Query by Coordinates**

- "The user inputs RA, Dec (ICRS), and a search radius (in degrees)."
- "The notebook should query the ALMA TAP service using ADQL via the PyVO library."
- "Use the `INTERSECTS()` ADQL function on the `ivoa.obscore` table with a `CIRCLE()` spatial region."
- "Return the results as a Pandas DataFrame."
- "Optionally sort results by angular distance from the input coordinates, using `DISTANCE()` in the query."

🗺️ **2. Query by Source Name (Name Resolver)**

- "The user inputs a known object name (e.g., "Cen A", "Orion KL")."
- "Resolve the object's sky coordinates using Astropy's `SkyCoord.from_name()` or Sesame."
- "Pass those coordinates to the coordinate query function above."

🔠 **3. Query by ALMA Target Name (Project Metadata)**

- "Allow the user to enter a string representing an ALMA target name (as recorded in project metadata)."
- "Provide two options:"
  - "Exact match (e.g., `target_name = 'Orion KL'`)"
  - "Substring match using SQL `LIKE '%...%'` pattern"

📦 **Technical Requirements:**

- "Use the TAP endpoint `https://almascience.eso.org/tap` (EU node)."
- "Return results as Pandas DataFrames."
- "Include basic exception handling for failed queries or empty results."
- "Use Markdown cells (or comments) to explain each step in the notebook."
- "Import standard packages: `pyvo`, `astropy`, `pandas`, `numpy`."

📊 **Output**

For each method:

- "Display the number of results found."
- "Show a table of selected fields: `target_name`, `s_ra`, `s_dec`, `band_list`, `obs_collection`, `access_url`, etc."
- "Bonus: Include a column that gives the angular distance from the original coordinate (if relevant)."

🧩 **Code Organization**

Structure the notebook with:

1. "A Setup section for package imports and TAP service initialization."
2. "A Function section for reusable query functions."
3. "An Examples / User Input section showing how each function works with real-world inputs."
4. "Optional markdown explanations before each section for clarity."

Please generate the entire notebook as Python code with Markdown blocks inline (i.e., in Jupyter Notebook format). I want this to be readable and reusable for scientific research or teaching.

Figure 5. Example prompt generated by ChatGPT to recreate ALMA notebook number two. The above image includes an "ideal" user-given prompt to accurately re-create a notebook. This prompt works well because of its deep level of detail. Prompting an LLM with something unspecific (for example, "Please replicate the code and plots in the second notebook") allows for much more variability as the LLM has less strict instructions for what exactly the user is looking for. The sectioned, bulleted requirements for the code provide Gemini with the exact steps and details necessary for creating a notebook, thus minimizing variability in the outputted results.

REFERENCES

Carrit Delgado Pinheiro, L., Chen, Z., Piazza, B. C., et al. 2025, Large Language Models Achieve Gold Medal Performance at the International Olympiad on Astronomy & Astrophysics (IOAA), arXiv:2510.05016, doi: 10.48550/arXiv.2510.05016

CASA Team, Bean, B., Bhatnagar, S., et al. 2022, PASP, 134, 114501, doi: 10.1088/1538-3873/ac9642

Ginsburg, A., Sipőcz, B. M., Brasseur, C. E., & Others. 2019, astroquery: An Astronomical Web-querying Package in Python, AJ, 157, 98, doi: 10.3847/1538-3881/aafc33

Joseph, S. A., Husain, S. M., Offner, S. S., et al. 2025, AstroVisBench: A Code Benchmark for Scientific Computing and Visualization in Astronomy, doi: 10.48550/arXiv.2505.20538

Juneau, S., Olsen, K., Nikutta, R., Jacques, A., & Bailey, S. 2021, Computing in Science and Engineering, 23, 15, doi: 10.1109/MCSE.2021.3057097

Lu, C., Lu, C., Lange, R. T., et al. 2024, The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery, arXiv:2408.06292, doi: 10.48550/arXiv.2408.06292

Nikutta, R., Fitzpatrick, M., Scott, A., & Weaver, B. A. 2020, Astronomy and Computing, 33, 100411, doi: 10.1016/j.ascom.2020.100411

Starace, G., Jaffe, O., Sherburn, D., et al. 2025, PaperBench: Evaluating AI's Ability to Replicate AI Research, doi: 10.48550/arXiv.2504.01848