

Language Model Powered Digital Biology

Joshua Pickard^{1*}, Marc Andrew Choi², Natalie Oliven¹, Cooper Stansbury¹,
Jillian Cwycyshyn³, Nicholas Galimoto¹, Alex Gorodetsky⁴,
Alvaro Velasquez⁵, and Indika Rajapakse^{1,2*}

¹Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI, USA

²Department of Mathematics, University of Michigan, Ann Arbor, MI, USA

³Department of Biomedical Engineering, University of Michigan, Ann Arbor, MI, USA

⁴Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, USA

⁵Department of Computer Science, University of Colorado Boulder, Boulder, CO, USA

*Corresponding authors: {jpic, indikar}@umich.edu

October 15, 2024

Abstract

Recent advancements in Large Language Models (LLMs) are transforming biology, computer science, and many other research fields, as well as impacting everyday life. While transformer-based technologies are currently being deployed in biology, no available agentic system has been developed to tackle bioinformatics workflows. We present a prototype Bioinformatics Retrieval Augmented Data (BRAD) digital assistant. BRAD is a chatbot and agentic system that integrates a suite of tools to handle bioinformatics tasks, from code execution to online search. We demonstrate its capabilities through (1) improved question-and-answering with retrieval augmented generation (RAG), (2) the ability to run complex software pipelines, and (3) the ability to organize and distribute tasks in agentic workflows. We use BRAD for automation, performing tasks ranging from gene enrichment and searching the archive to automatic code generation for running biomarker identification pipelines. BRAD is a step toward autonomous, self-driving labs for digital biology.

1 Introduction

Recent advances in artificial intelligence (AI), including Large Language Models (LLMs) and scientific Foundation Models (FM), show potential for optimizing research workflows and accelerating discovery. For instance, autonomous AI agents have been developed to automate tasks such as knowledge processing, experimental design and analysis, and automation in chemical engineering [1], aerospace research [2], medicine [3], and related fields. These multiagent systems augment classical modes of research, ranging from literature search and document review to laboratory automation, with the information processing and decision making capabilities of LLMs to form semi-autonomous systems.

Collaborative AI systems are now found in various sectors. These leverage Human-AI and AI-AI interactions to solve problems ranging from web search, question-answering (QA), game play, and more [4, 5, 6, 7, 8, 9]. The utility of ChatGPT, for instance, emerged not from a new transformer architecture, which is now six years old, but instead from coupling the LLM with a graphical user interface (GUI) that places a human in the loop [10, 11]. Similarly, the recently introduced OpenAI o1 chatbot responds to user queries with a mixture of LLMs and tools in an agentic workflow.

The construction of self-driving labs – collaborative, agentic systems for scientific discovery – aims to enhance reproducibility and increase the throughput of automated experiments. This framework has been employed in chemistry [12, 1, 13], materials science [14, 15], and biology [16, 17, 18]. Digital biology and the virtual cell program share similar objectives with self-driving labs, aiming to leverage *in silico* experimentation rather than automation [19, 20, 21, 22, 23]. However, challenges persist in areas of

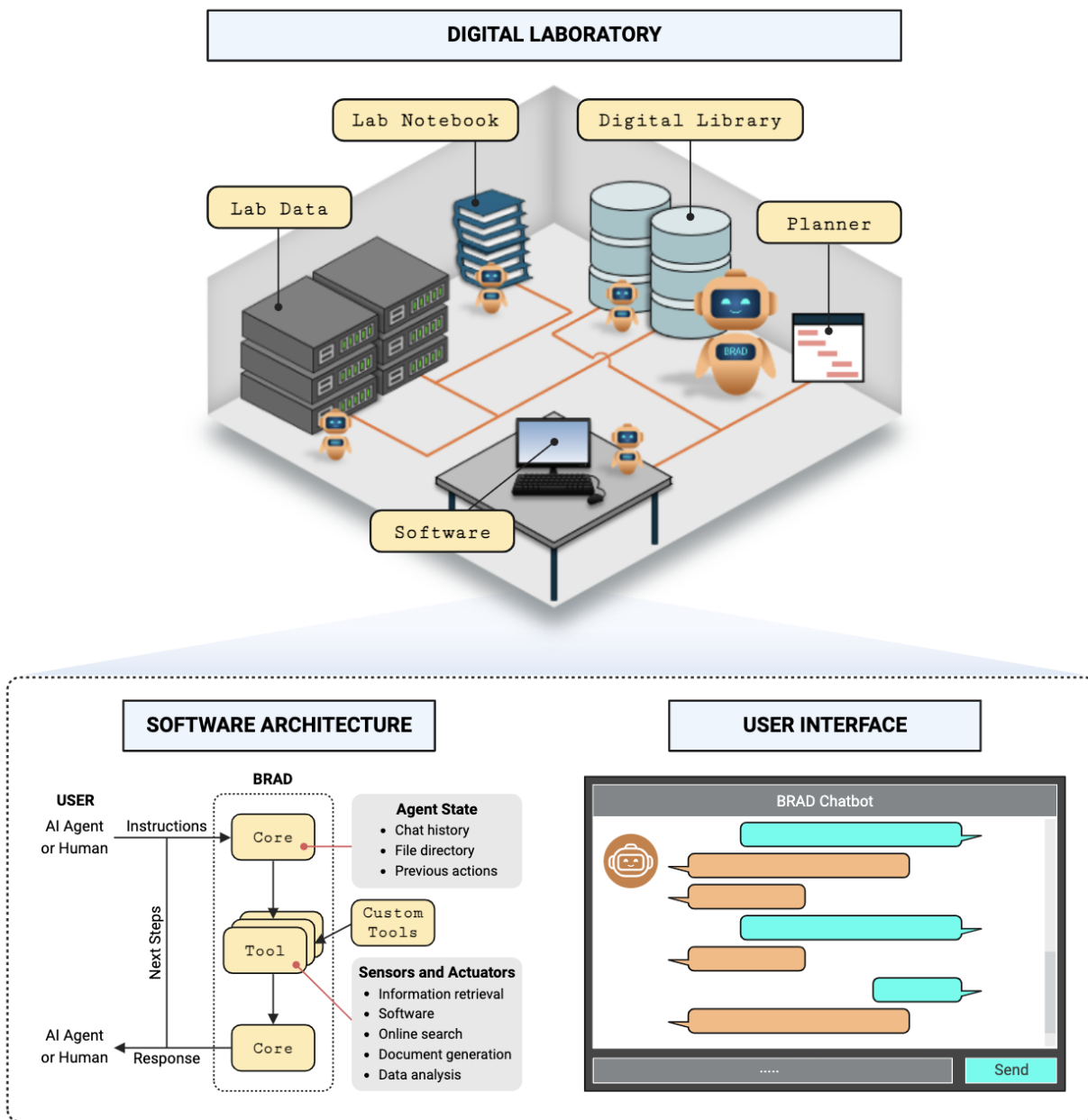


Fig 1. Software Framework. (**Digital Laboratory**) BRAD agents and tools interact with a human operator in a digital laboratory. (**Software Architecture**) BRAD’s modular design includes core and tool modules that enable flexible and customizable agentic workflows. (**User Interface**) A graphical user interface (GUI) allows users to interact with BRAD agents.

automated analysis and computational workflows, particularly regarding the integration of diverse data sources and protocols.

One challenge faced by self-driving labs mirrors that of traditional laboratories: since the Human Genome Project’s Bermuda Principles [24], the bioinformatics community has prioritized the rapid dissemination and storage of experimental data. However, ever-evolving experimental techniques, nomenclatures, and databases create obstacles for the knowledge processing problem.

Collaborative AI systems offer solutions for the knowledge processing challenge. For example, retrieving information from bioinformatics databases parallels LLM-based retrieval systems [25, 26]. In addition to retrieval, LLMs are well-suited for automating data processing pipelines and other repetitive tasks. While transformer-based models have found other use cases in digital biology [27, 28, 29, 30, 31, 32, 33, 34], these technologies remain underutilized for knowledge processing due to limited integration into existing bioinformatics databases, tools, and workflows.

This work presents a prototype digital assistant (BRAD), which integrates recent LLM technologies with traditional bioinformatics tools. BRAD is organized into modules, each powered by an LLM and responsible for a different task, such as literature search (**LAB NOTEBOOK** module), software generation and execution (**SOFTWARE** module), and database / web search (**DIGITAL LIBRARY** module). BRAD’s modules work together to solve more complex challenges such as database construction and biomarker identification. This collaboration between BRAD’s modules and multiple agents allows each agent to address tasks of greater complexity (see fig. 1).

2 Results

This section presents the architecture of BRAD followed by several application-oriented results.

2.1 BRAD Software Architecture

BRAD is an AI digital assistant that integrates bioinformatics and research tools through LLM and chatbot interfaces. It supports both locally-run LLMs, which can be fine-tuned to increase domain knowledge, and API connections to online models such as NVIDIA’s Nemotron and OpenAI’s offerings. BRAD is also compatible with LangChain and LangGraph, ensuring compatibility with standard frameworks [35].

A BRAD agent consists of tool and core modules. Core modules maintain the agents state and environment by managing resources, such as data files and LLM utilization. An output directory, chat log, and memory facilitate the information flow between the various modules. For queries demanding multiple modules, BRAD’s core **PLANNER** module breaks down the query into a series of tasks to be handled by individual tools, similar to a ReAct agent [36].

Tool modules connect the LLM agent to external databases and software. These modules are built upon the retrieval augmented generation (RAG) blueprint (see fig. 2A, fig. 3A), with each tool accessing information from a different source. BRAD contains three tool modules: the **LAB NOTEBOOK** for retrieving information from literature and documentation, the **DIGITAL LIBRARY** for searching online databases and archives, and the **SOFTWARE** module for running code. This modular architecture grants BRAD the flexibility to integrate diverse tools while affording users a framework to customize and expand BRAD’s capabilities. For further implementation details, see §SI 4.

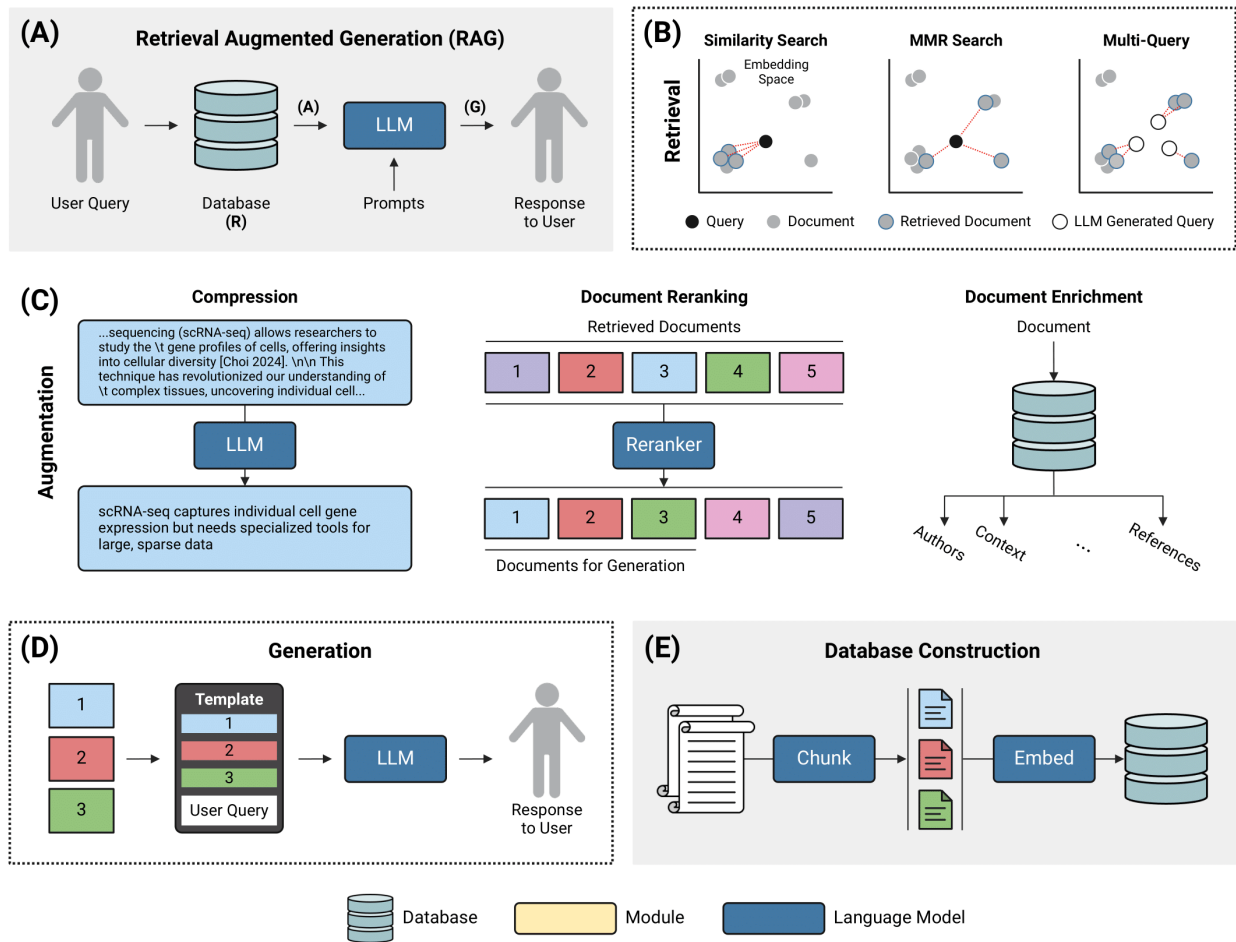


Fig 2. Overview of LAB NOTEBOOK, a tool module. **(A)** The LAB NOTEBOOK uses a RAG pipeline that access a document database to provide LLMs additional information for answering user queries. **(B)** Documents are retrieved from a database by searching using one of three methods: Similarity search selects the most relevant documents or those closest to the users query; MMR search selects relevant documents while maximize diversity among them; and multi-query search uses LLMs to generate alternative formulations of the user’s question, retrieving documents relevant to these variations. **(C)** Retrieved documents are augmented before answering the users query. Compression summarizes and enhances the clarity of each source. Reranking organizes the documents based on their relevance, helping to determine which ones to use for generation. Enrichment searches the database for additional information such as surrounding context, authors, or other metadata. **(D)** The augmented documents, along with the original user query, are provided to the LLM to generate a response. **(E)** The document database is constructed by parsing and splitting into chunks PDF documents and then generating vector embeddings of each chunk that can be used for retrieval.

2.1.1 Lab Notebook Module

The **LAB NOTEBOOK** retrieves information from documents, such as research articles, protocols, or any pdf file, to augment LLM generated responses using a RAG pipeline [25, 37] (see fig. 2A, §SI 4.2). This operates in three stages: (1) relevant documents are retrieved from the database using similarity-based, maximum marginal relevance (MMR) [38], and multi-query retrieval (see fig. 2B); (2) these documents are augmented with contextual compression [39] or enrichment and their importance is reranked [2] (see fig. 2C); (3) finally, the documents and user query are input to a LLM that responds to the user with additional context (see fig. 2D). The **LAB NOTEBOOK** is initialized either with user-supplied documents, or BRAD can dynamically build a RAG database from online search (see fig. 2E).

2.1.2 Software Module

The **SOFTWARE** module enables BRAD to interact with, analyze, and run custom data processing workflows (see §SI 4.3). Documentation of external code, whitelisted by the user, are retrieved and used, in conjunction with the user query, to prompt the LLM. The LLM selects and produces code to run the appropriate software. The LLM generated code is validated and checked for common errors, such as misconfigured input and output paths, and then executed (see SI fig. 6). Because many workflows are repetitive and the set of software to run is constrained by the user, the LLM can reliably produce correct codes.

Errors identified during code validation are sent back to the LLM along with the original query for refinement. This iterative process of code generation and validation continues until the code executes without error. Once all issues are resolved, the code is run.

Designed for bioinformatics workflows, where data is often read and written to files during each stage, the use of whitelisted software simplifies validation of the correct input/output paths as well as many other arguments of LLM generated code. This facilitates running codes with specific and isolated purposes, such as differential expression or pseudotime analysis.

2.1.3 Digital Library Module

The **DIGITAL LIBRARY** module retrieves information from online bioinformatics databases and literature archives (see §SI 4.4). BRAD accesses Enrichr and Gene Ontology, two commonly-used databases in bioinformatics workflows, through their application programming interfaces (APIs) [40, 41, 42]. BRAD can search terms extracted from data, such as spreadsheets, or from user queries, and download both structured and unstructured results.

BRAD also integrates the arXiv, bioRxiv, and PubMed literature databases. Search results, including recent articles and abstracts can be displayed to the user, and BRAD can retrieve and download the articles. When articles are downloaded, the **DIGITAL LIBRARY** can place the new content into the RAG database of the **LAB NOTEBOOK** so that the user can query directly from the new texts.

2.1.4 Planning and Routing Modules

The **PLANNER** and **ROUTER** modules orchestrate multi-module, agentic workflows for a BRAD agent (see §SI 4.5, §SI 4.6). The **PLANNER** generates a blueprint in the form of a directed graph, linking tool modules and defining the execution order and information flow between them (see fig. 3C). Users can interactively revise and edit the workflow and save or reuse previously generated plans from a library.

The **ROUTER** module provides two levels of coordination for a BRAD agent. For an individual query, the **ROUTER** uses semantic routers to select the appropriate tool module (see fig. 3B). For an agentic workflow designed by the **PLANNER**, the **ROUTER** enables branching and cycles within the plan by having the LLM evaluate the history and determine the next module (see fig. 3D).

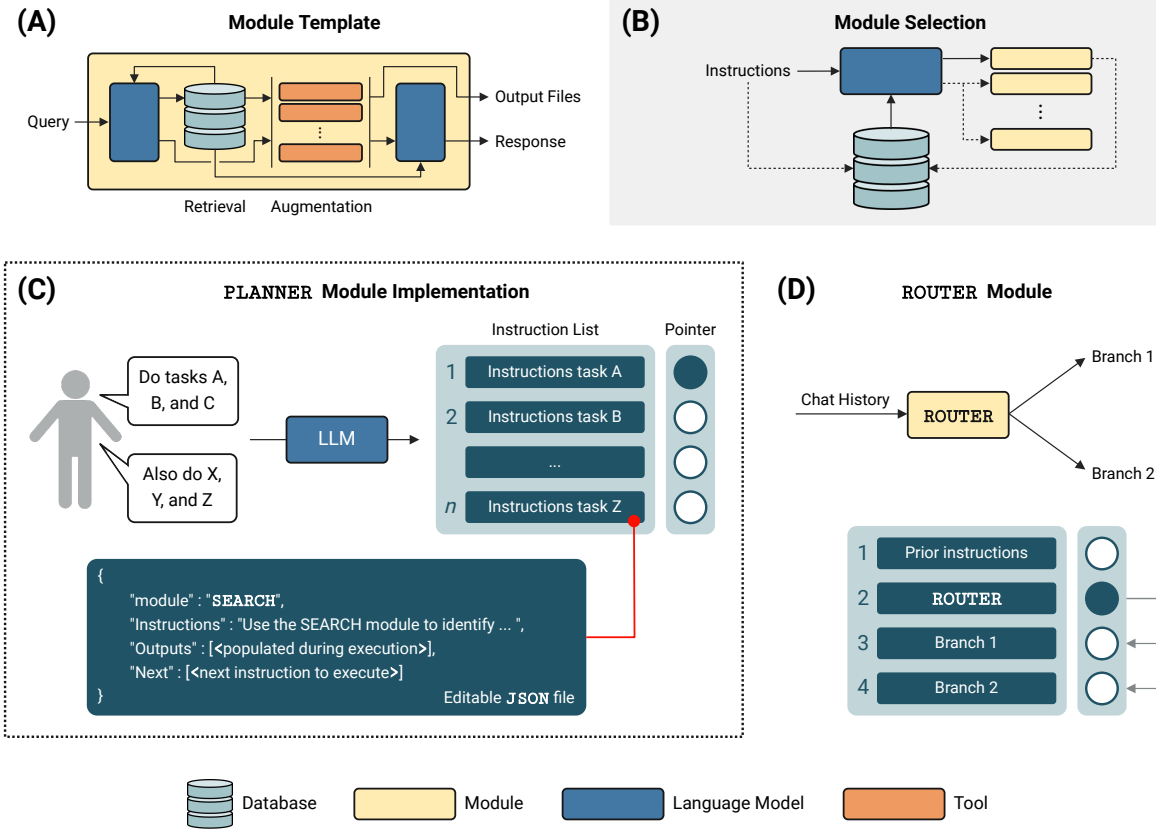


Fig 3. Overview of agentic workflow core modules. **(A)** The module template serves as a wrapper for connecting tools or databases with LLMs to respond to user inquiries, much like RAG. **(B)** Based on a dynamically updated database of previously encountered prompts, the **ROUTER** selects the most fitting tool module for each input directed to a BRAD agent. **(C)** The **PLANNER** module converts user queries into a set of instructions to be answered with different tool modules. These are saved to an editable **JSON** file and can be modified interactively. **(D)** The **ROUTER** module also determines the control flow of multi-module agentic workflows.

	LAB NOTEBOOK	DIGITAL LIBRARY	SOFTWARE	Internal	External
QA with the vanilla LLM	✓				✓
QA with the standard RAG	✓				✓
QA with the RAG using contextual compression	✓				✓
Contextual compression	✓			✓	
Database selection		✓		✓	
Load data from a file for database search		✓		✓	
Software selection			✓	✓	
Software execution			✓		✓

Table 1. Use cases of the LLM across various modules. Some LLM interactions are intended for internal processing, while others generate user-facing responses.

2.2 Digital Laboratory

We evaluated BRAD on a range of tasks to both benchmark the performance of individual modules and demonstrate the utility of agentic workflows and multi-agent systems.

2.2.1 Benchmarking Tool Modules

To evaluate BRAD’s tool modules, we designed a set of prompts targeted at the LAB NOTEBOOK, DIGITAL LIBRARY, and SOFTWARE tool modules and profiled the cost of executing these modules (see §SI 5.1). The cost associated with of LLM applications occurs in the form of time, size, and potentially API costs depending on where the LLM is running. We used OpenAI’s `gpt-4o` and `gpt-3.5-turbo-0125` models, which like other providers, charge per input/output token. Eight use cases of the LLM in BRAD were profiled (see table 1).

BRAD’s response time varies across module, with an average LLM query taking less than 2 seconds making it fast enough for interactive use (see SI fig. 9, SI table 2). There is a relatively narrow distribution for the costs of non-QA tasks, likely due to BRAD’s prompt template library, which uses tailored prompts that only vary due from the user input when querying the LLM. The increased variability in cost of QA is attributed to both the increased number and variability in input tokens supplied to both versions of the RAG as well as the increased cost of using a vanilla LLM. While the inputs tokens to a vanilla LLM are a function only of the user’s query, the LLM without the RAG produces significantly more output tokens, which increases its cost.

2.2.2 RAG Assesment

As an example, consider Q1: What are the coordinates of the genome? Figure 4A, compares answers from `o1-preview` and `gpt-4o` with the LAB NOTEBOOK. `o1-preview`, a recently released agentic chatbot with both the latest LLM technology and access to unknown computation and search tools, answered Q1 from the perspective of a bioinformatician: providing a surface level response regarding the measurement coordinates used to represent sequencing data. `gpt-4o` with LAB NOTEBOOK responds like a developmental biologist: discussing the role of the physical and geometric structure in shaping cell identity. While both answers are scientifically valid, the `o1-preview`’s response characterizes the current sentiment toward Q1 (`o1-preview`’s response is similar to a google search of the same question), but neglects the perspective that once dominated [43]. Retrieving contextually relevant information can (1) reduce the bias toward popular or prevailing perspectives in the model’s training data, and (2) provide transparency about the tools and data used to generate the response.

To evaluate the quality of the responses from the LAB NOTEBOOK’s RAG pipeline, a series of research questions were generated based on documents in the database (see §SI 5.2.3). The database was built from over 500 documents, totaling more than 18,000 pages (see fig. 4C).¹

¹The set of documents contained in our database is available here: <https://rajapakse.lab.medicine.umich.edu/papers/digital-library>

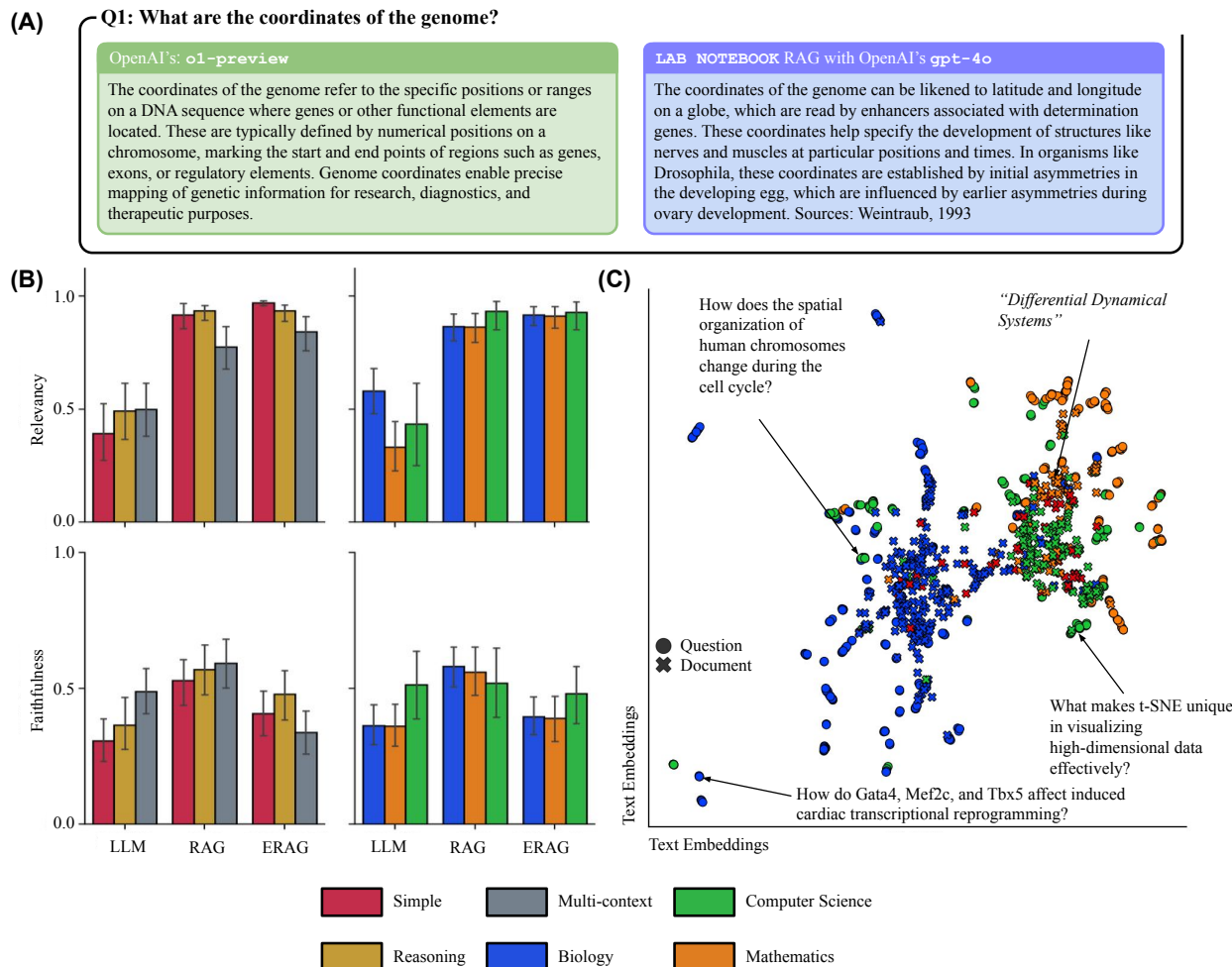


Fig 4. Profiling of LAB NOTEBOOK. **(A)** RAGAs metrics evaluate QA capabilities utilizing various configurations of the LAB NOTEBOOK. **(B)** Evaluation questions and document titles from the RAG database are displayed in UMAP embeddings. While biology is separated, there is overlap between computer science and mathematics texts. There appears a wider spread of questions relative to document titles. This embedding space is used for document retrieval.

RAG Assessment (RAGAs) was used to generate and evaluate responses to these questions [44]. The questions were classified into three types: those requiring a direct, *simple* response, those requiring *reasoning*, and those that required synthesizing information from *multiple contexts*. Human annotators further categorized the questions by domain, labeling them as belonging to Biology, Mathematics, or Computer Science. These questions were evaluated against three different configurations of BRAD: (1) a LLM without RAG (Vanilla), (2) a standard RAG pipeline (RAG), and (3) an enhanced RAG with multi-query retrieval, compression, and document reranking (ERAG).

The RAG systems demonstrated improved faithfulness and relevancy when answering the question dataset (see fig. 4B). To measure the quality of BRAD's responses, verifiable statements of the generated answers were compared against the text used to generate each question (see §SI 5.2.2). Faithfulness, for instance, measures how well the LLM generated answer aligns with verifiable information retrieved from the document database. Relevancy, on the other hand, measures the similarity of the LLM's response to the question or user query. While these metrics illustrate the potential utility of RAG, it remains an open challenge to quantify the utility of a LLM's response to unknown and open ended research questions.

2.2.3 Enhanced Search

Multi-module agentic workflows improve the depth and quality of BRAD’s responses. Consider Q2: What role does PCNA play in the cell cycle and imaging? BRAD utilizes the **SOFTWARE** module for network analysis and gene association, performs pathway enrichment with the **DIGITAL LIBRARY** by querying PCNA-related genes, and searches the literature via the **LAB NOTEBOOK** for relevant information (see fig. 5B). Along with a detailed text response, this process also generates images, spreadsheets, and other documentation to provide additional context for the user (see §SI 5.5).

Q2: What role does PCNA play in the cell cycle and imaging?

BRAD Output

PCNA’s Role in the Cell Cycle: PCNA (Proliferating Cell Nuclear Antigen) plays a crucial role in the cell cycle, specifically in the S phase. It is loaded onto DNA during S phase, and its interaction with the minimal human Cdt1 PIP degron triggers the degradation of a cell cycle reporter, providing a direct physical readout of DNA replication.

PCNA’s Role in Cell Cycle Imaging: PCNA is a DNA replication sensor that can be used to track the cell cycle. It appears as a diffuse nuclear pattern in G1 phase, punctate in S phase, diffuse again in G2 phase, and diffuses throughout the whole cell in M phase. It is a high-contrast S/G2 transition marker, but requires high-resolution images and regular nuclear morphology and monolayer culture.

...

See section 5.5 for the full output and generated documents.

2.2.4 Building a RAG Database

BRAD was assigned the task of building a database from the ground up to answer specific research questions. Alternating between the **LAB NOTEBOOK** and **DIGITAL LIBRARY** (see fig. 5E), BRAD searched online archives to build a RAG database and answer user queries.

Consider the question Q3: What are single cell foundation models? The **ROUTER**, which recognizes insufficient answer produced by the **LAB NOTEBOOK** in iteration 1, directs BRAD to use the **DIGITAL LIBRARY**. After searching online and incorporating new articles into the RAG database, the **LAB NOTEBOOK** responds to the original user query, and the response is again evaluated by the **ROUTER**. This process is repeated until either BRAD’s core modules detect infinite looping behavior or the **ROUTER** detects a high quality response and exits the cycle. BRAD’s final response to Q3, in iteration 2, provides a detailed overview of single-cell foundation models, including several notorious examples and use cases. See §SI 5.3 for the intermediary responses.

Q3: What is a single cell foundation model?

RAG: Iteration 1 (empty database)

Based on the provided source and content, the initial database is empty, and I couldn't find any information regarding "single cell foundation models." I am confident that I cannot provide an accurate answer to this question with the current information available.

Search Terms: (1) single cell foundation models, (2) machine learning for single cell data analysis, (3) single cell machine learning, and (4) deep learning for single cell analysis

RAG: Iteration 2 (populated database)

Single cell foundation models are artificial intelligence models designed to analyze and understand data from individual cells. They are constructed using single-cell transcriptomic data, which measures the expression levels of genes in individual cells. These models utilize machine learning algorithms, such as transformers, to learn gene-level and cell-level representation. They can accurately categorize known cell types and have the potential to uncover new biological insights in single-cell RNA sequencing data. Single cell foundation models are used in genomics and multi-omics research, showing promise in various downstream tasks, such as identifying cell types and states, discovering novel cell populations, inferring gene regulation networks, and more. Examples of single cell foundation models include scGPT, Genefomer, and scFoundation. They are a growing field of study, with ongoing advancements expected as single-cell technologies and omics sciences continue to develop.

2.2.5 Biomarker Identification and Consensus

Biomarker identification – finding genes that indicate a biological state – remains an active and open research problem with both research and healthcare applications [45, 46]. Searching for new biomarkers often follows a two step process of (1) modeling a system and (2) analyzing observability properties of different genes or chemical species [47]. We used BRAD to execute a workflow for biomarker identification and evaluation (see fig. 5D).

Fibroblast time series expression data were obtained from [48, 49], and BRAD used the provided software to construct linear models of gene expression, ranking genes as biomarkers by observability analysis (see §SI 5.4). To verify the accuracy of this workflow execution, the selected biomarkers were compared with the results in [50], where this particular biomarker workflow was developed. To validate the utility of the biomarkers, BRAD trained a feed-forward neural network (FNN) to predict cell cycle phases based on biomarker expression. Single-cell RNAseq data from paired with cell cycle phases derived from imaging from [51], were used to train and test the FNN.

Although this workflow uses established methods and widely used models, executing it with a collaborative AI offers several advantages. First, collaborating with an AI allows the user to specify key parameters, while the burden of managing standard configurations falls to the AI. Second, the autonomy of the AI allows the workflow to run more times with a wider range of parameters and inputs. Third, each stage of the pipeline can be paired with other tool modules to perform tasks such as biomarker enrichment or searching for similar results in literature. Finally, while BRAD executes the workflow semi-autonomously, the full code to manually execute the workflow is provided to the user to maintain reproducibility and transparency.

Compared to `o1-preview`, using BRAD agents powered by `gpt-3.5` to execute code provides several key advantages over the newer model (see §SI 5.4.2). Specifically, granting the LLM direct access to data and software greatly enhances both the utility and structure of its responses, particularly for users aiming to apply the identified biomarkers.

Despite the success of running biomarker workflow with BRAD, there remain several practical challenges in biomarker identification. Questions such as: how many biomarkers? do biomarkers work at all times? do biomarkers work across cell types? and more, remain unanswered. Specifically, while [48] and [49] study the same cell line, different experimental conditions are used, and while [47] and [50] proposes biomarker selection protocols, these provide choices for observability metrics and can be configured for static or dynamics biomarkers. For a multi-agent, collaborative AI system to work in practice, it must form consensus from the diverse set of possibly selected biomarkers.

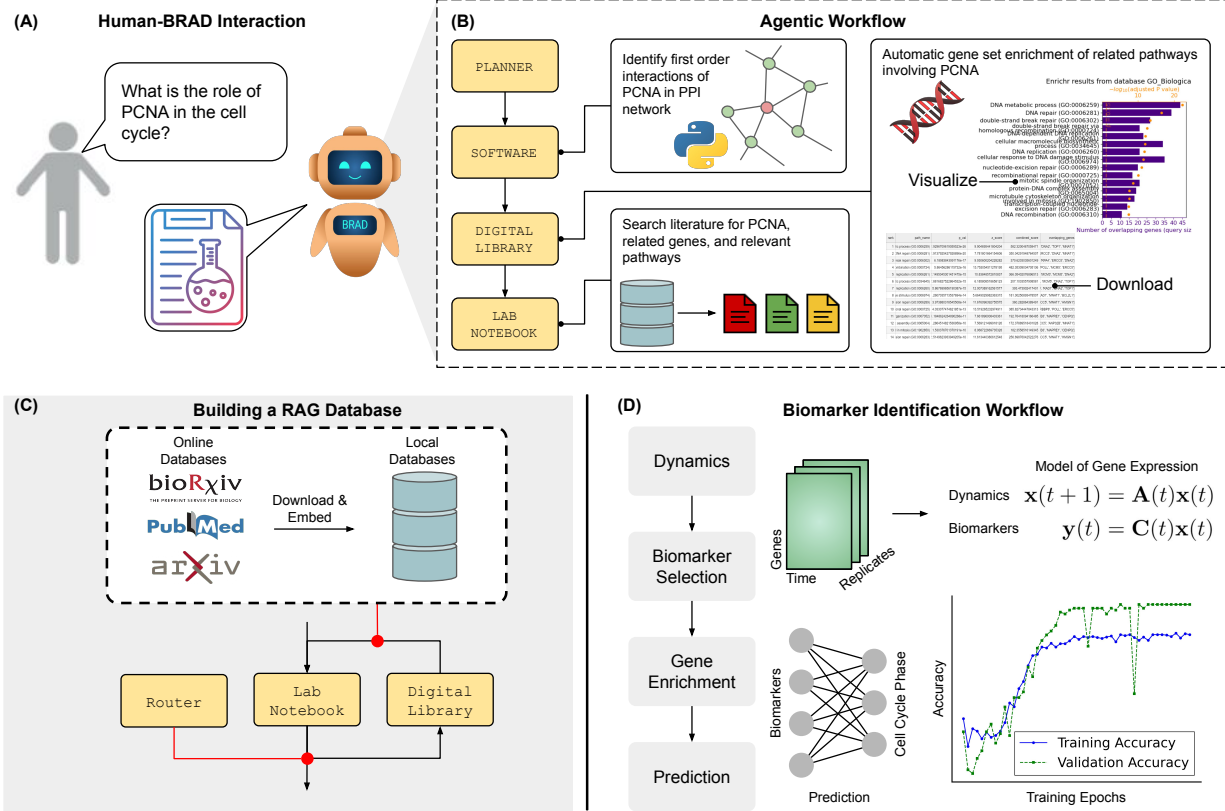


Fig 5. Agentic Workflows with BRAD. **(A)** User’s pass BRAD complex queries that can be answered with multiple modules. **(B)** BRAD uses three modules to answer Q2 and assembles a final report. **(C)** Combining the LAB NOTEBOOK, DIGITAL LIBRARY, and ROUTER modules allows BRAD to build a custom RAG database with online retrieval. **(D)** Combining several SOFTWARE modules, BRAD performs a biomarker identification workflow.

To evaluate BRADs ability to achieve consensus, a team of multiple agents were provided biomarkers and instructed to reach a consensus as to which markers are best. In this setting, agents send information to one another until either agreement is achieved or time runs out. In some, but not all cases, the system does achieve agreement; however, as with experiment design, consensus is not always achieved.

3 Discussion

This work presents a prototype AI research assistant. Integrating an array of data repositories, software, and literature with LLMs creates capabilities that exceed the sum of each individual tool. The modular design of BRAD will facilitate the emergence of improved research capabilities as new tools and LLMs are integrated into BRAD.

The ability of LLMs to quickly make reliable, verifiable decisions for routine research tasks is a strength of agentic systems. The modular architecture of BRAD offers a blueprint for scaling up the number of agents that collaborate while still allowing humans control over the different aspects of the pipeline.

Current design of such systems must address two significant challenges. First, human skepticism of agentic systems is pervasive, and necessitates the development of interpretable, transparent agents. Although LLMs themselves may lack transparency due to unknown training data and opaque internal mechanisms, the surrounding chat interface and agentic systems should provide transparency by clearly showing the data, references, and tools used in generating responses.

Second, a central challenge to the design of agentic systems is maintaining pace with the advancement of

LLM technology. While current integrations with tools like Gene Ontology, arXiv, or RAG improve the utility of today’s LLMs, they risk becoming obsolete as future models improve in reasoning and problem-solving capabilities. This raises a critical question: how do we build systems today that are flexible enough to adapt to tomorrow’s LLMs? If agentic systems are too closely tied to a specific LLM, they could quickly become outdated. However, creating a truly LLM-agnostic agentic framework remains an open challenge. The modularity of BRAD offers one way to address this, enabling adaptability to future advances without dependence on any single model.

In summary, this work highlights the advantages and select use cases of a collaborative AI research assistants. Despite the persisting challenges, there currently are opportunities to use LLMs for well-defined and repetitive tasks, such as running software pipelines or performing enrichment. BRAD, as a tool, demonstrates these capabilities. Moving forward, we aim to enhance the planning and coordination between modules and multiple BRAD agents, with the goal of advancing towards a fully autonomous digital laboratory.

Acknowledgments

We thank Adam Lord of NVIDIA, Santosh Srivastava of General Electric, and members of the Rajapakse Lab for helpful and inspiring discussions. We also thank Vivan Nyati, a student of Greenhills, Ann Arbor, and Ram Prakash of University of Michigan for their valuable contributions. This work was supported by the Defense Advanced Research Projects Agency award number HR00112490472 (IR), the Air Force Office of Scientific Research (AFOSR) award number FA9550-22-1- 0215 (IR), support from NVIDIA (IR), and NIGMS GM150581 (JP). BioRender was used in the assembly of figures 1, 2, and 3.

Software Availability

The source code for BRAD and its documentation are attached via links:

- Source Code: <https://github.com/Jpickard1/BRAD>
- Software Manual: https://brad-bioinformatics-retrieval-augmented-data.readthedocs.io/_/downloads/en/latest/pdf/

BRAD is distributed in several formats, including a `python` package, a command-line tool, and a containerized graphical user interface. All distributions are based on the `python` package, but contain access to varying features at this time.

References

1. Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.
2. Vansh Sharma and Venkat Raman. A reliable knowledge processing framework for combustion science using foundation models. *Energy and AI*, 16:100365, 2024.
3. Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.
4. Tamer Abuelsaad, Deepak Akkil, Prasenjit Dey, Ashish Jagmohan, Aditya Vempaty, and Ravi Kokku. Agent-e: From autonomous web navigation to foundational design principles in agentic systems. *arXiv preprint arXiv:2407.13032*, 2024.
5. Saaket Agashe, Yue Fan, and Xin Eric Wang. Evaluating multi-agent coordination abilities in large language models. *arXiv preprint arXiv:2310.03903*, 2023.

6. Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023.
7. Pengying Wu, Yao Mu, Kangjie Zhou, Ji Ma, Juntao Chen, and Chang Liu. Camon: Cooperative agents for multi-object navigation with llm-based conversations. *arXiv preprint arXiv:2407.00632*, 2024.
8. Xudong Guo, Kaixuan Huang, Jiale Liu, Wenhui Fan, Natalia Vélez, Qingyun Wu, Huazheng Wang, Thomas L Griffiths, and Mengdi Wang. Embodied llm agents learn to cooperate in organized teams. *arXiv preprint arXiv:2403.12482*, 2024.
9. Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008, 2023.
10. Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.
11. Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
12. Jeffrey A Bennett and Milad Abolhasani. Autonomous chemical science and engineering enabled by self-driving laboratories. *Current Opinion in Chemical Engineering*, 36:100831, 2022.
13. Simon D Rihm, Yong Ren Tan, Wilson Ang, Markus Hofmeister, Xinhong Deng, Michael Teguh Laksana, Hou Yee Quek, Jiaru Bai, Laura Pascasio, Sim Chun Siong, et al. The digital lab manager: Automating research support. *SLAS technology*, 29(3):100135, 2024.
14. Milad Abolhasani and Eugenia Kumacheva. The rise of self-driving labs in chemical and materials sciences. *Nature Synthesis*, 2(6):483–492, 2023.
15. Nathan J Szymanski, Bernardus Rendy, Yuxing Fei, Rishi E Kumar, Tanjin He, David Milsted, Matthew J McDermott, Max Gallant, Ekin Dogus Cubuk, Amil Merchant, et al. An autonomous laboratory for the accelerated synthesis of novel materials. *Nature*, 624(7990):86–91, 2023.
16. Hector G Martin, Tijana Radivojevic, Jeremy Zucker, Kristofer Bouchard, Jess Sustarich, Sean Peisert, Dan Arnold, Nathan Hillson, Gyorgy Babnigg, Jose M Marti, Christopher J Mungall, Gregg T Beckham, Lucas Waldburger, James Carothers, ShivShankar Sundaram, Deb Agarwal, Blake A Simmons, Tyler Backman, Deepanwita Banerjee, Deepti Tanjore, Lavanya Ramakrishnan, and Anup Singh. Perspectives for self-driving labs in synthetic biology. *Current Opinion in Biotechnology*, 79:102881, 2023.
17. Jillian Cwycyshyn, Cooper Stansbury, Walter Meixner, James B Hoying, Lindsey A Muir, and Indika Rajapakse. Automated in vitro wound healing assay. *bioRxiv*, pages 2023–12, 2023.
18. Deyber Arley Vargas Medina, Edvaldo Vasconcelos Soares Maciel, and Fernando Mauro Lanças. Modern automated sample preparation for the determination of organic compounds: A review on robotic and on-flow systems. *TrAC Trends in Analytical Chemistry*, page 117171, 2023.
19. Charlotte Bunne, Yusuf Roohani, Yanay Rosen, Ankit Gupta, Xikun Zhang, Marcel Roed, Theo Alexandrov, Mohammed AlQuraishi, Patricia Brennan, Daniel B Burkhardt, et al. How to build the virtual cell with artificial intelligence: Priorities and opportunities. *arXiv preprint arXiv:2409.11654*, 2024.
20. Aneil Mallavarapu, Matthew Thomson, Benjamin Ullian, and Jeremy Gunawardena. Programming with models: modularity and abstraction provide powerful capabilities for systems biology. *Journal of The Royal Society Interface*, 6(32):257–270, 2009.

21. Hanwen Wang, Theinmozhi Arulraj, Alberto Ippolito, and Aleksander S Popel. From virtual patients to digital twins in immuno-oncology: lessons learned from mechanistic quantitative systems pharmacology modeling. *NPJ Digital Medicine*, 7(1):189, 2024.
22. Arsen Osipov, Ognjen Nikolic, Arkadiusz Gertych, Sarah Parker, Andrew Hendifar, Pranav Singh, Darya Filippova, Grant Dagliyan, Cristina R Ferrone, Lei Zheng, et al. The molecular twin artificial-intelligence platform integrates multi-omic data to predict outcomes for pancreatic adenocarcinoma patients. *Nature cancer*, 5(2):299–314, 2024.
23. Hugo Chenel, Malvina Maruk, Tim James, Andrei Zinovyev, and Vera Pancaldi. Multilayer network approaches to ‘omics data integration in digital twins for cancer research. *arXiv:2410.07252*, 2024.
24. Human Genome Organisation. Summary of the report of the second international strategy meeting on human genome sequencing. *Bermuda: Human Genome Organisation*, 1997.
25. Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.
26. Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zhicheng Dou, and Ji-Rong Wen. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*, 2023.
27. Fan Yang, Wenchuan Wang, Fang Wang, Yuan Fang, Duyu Tang, Junzhou Huang, Hui Lu, and Jianhua Yao. scbert as a large-scale pretrained deep language model for cell type annotation of single-cell rna-seq data. *Nature Machine Intelligence*, 4(10):852–866, 2022.
28. Christina V Theodoris, Ling Xiao, Anant Chopra, Mark D Chaffin, Zeina R Al Sayed, Matthew C Hill, Helene Mantineo, Elizabeth M Brydon, Zexian Zeng, X Shirley Liu, et al. Transfer learning enables predictions in network biology. *Nature*, 618(7965):616–624, 2023.
29. Frederikke Isa Marin, Felix Teufel, Marc Horlacher, Dennis Madsen, Dennis Pultz, Ole Winther, and Wouter Boomsma. Bend: Benchmarking dna language models on biologically meaningful tasks. In *The Twelfth International Conference on Learning Representations*, 2023.
30. Mengzhou Hu, Sahar Alkhairy, Ingoo Lee, Rudolf T Pillich, Dylan Fong, Kevin Smith, Robin Bachelder, Trey Ideker, and Dexter Pratt. Evaluation of large language models for discovery of gene set function. *arXiv:2309.04019v2*, 2023.
31. Hongyu Zhao, Tianyu Liu, Kexing Li, Yuge Wang, and Hongyu Li. Evaluating the utilities of large language models in single-cell data analysis. 2023.
32. Haotian Cui, Chloe Wang, Hassaan Maan, Kuan Pang, Fengning Luo, Nan Duan, and Bo Wang. scgpt: toward building a foundation model for single-cell multi-omics using generative ai. *Nature Methods*, pages 1–11, 2024.
33. Jiajia Liu, Mengyuan Yang, Yankai Yu, Haixia Xu, Kang Li, and Xiaobo Zhou. Large language models in bioinformatics: applications and perspectives. *arXiv:2401.04155v1*, 2024.
34. Wenpin Hou and Zhicheng Ji. Assessing gpt-4 for cell type annotation in single-cell rna-seq analysis. *Nature Methods*, pages 1–4, 2024.
35. Ben Affar. *Generative AI with LangChain: Build Large Language Model (LLM) Apps with Python, ChatGPT, and Other LLMs*. Packt Publishing Ltd., Birmingham, UK, 2023.
36. Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

37. Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*, 2021.
38. Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries, 1998.
39. Griffin Adams, Alexander Fabbri, Faisal Ladhak, Eric Lehman, and Noémie Elhadad. From sparse to dense: Gpt-4 summarization with chain of density prompting. *arXiv preprint arXiv:2309.04269*, 2023.
40. Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.
41. Edward Y Chen, Christopher M Tan, Yan Kou, Qiaonan Duan, Zichen Wang, Gabriela Vaz Meirelles, Neil R Clark, and Avi Ma’ayan. Enrichr: interactive and collaborative html5 gene list enrichment analysis tool. *BMC bioinformatics*, 14:1–14, 2013.
42. Suzi A Aleksander, James Balhoff, Seth Carbon, J Michael Cherry, Harold J Drabkin, Dustin Ebert, Marc Feuermann, Pascale Gaudet, Nomi L Harris, et al. The gene ontology knowledgebase in 2023. *Genetics*, 224(1):iyad031, 2023.
43. H Weintraub. Summary: genetic tinkering—local problems, local solutions. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume 58, pages 819–836. Cold Spring Harbor Laboratory Press, 1993.
44. Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*, 2023.
45. Lee Hartwell, David Mankoff, Amanda Paulovich, Scott Ramsey, and Elizabeth Swisher. Cancer biomarkers: a systems approach. *Nature biotechnology*, 24(8):905–908, 2006.
46. Robert M Califf. Biomarker definitions and their applications. *Experimental biology and medicine*, 243(3):213–221, 2018.
47. Aqib Hasnain, Shara Balakrishnan, Dennis M Joshy, Jen Smith, Steven B Haase, and Enoch Yeung. Learning perturbation-inducible cell states from observability analysis of transcriptome dynamics. *Nature Communications*, 14(1):3148, 2023.
48. Haiming Chen, Jie Chen, Lindsey A Muir, Scott Ronquist, Walter Meixner, Mats Ljungman, Thomas Ried, Stephen Smale, and Indika Rajapakse. Functional organization of the human 4d nucleome. *Proceedings of the National Academy of Sciences*, 112(26):8002–8007, 2015.
49. Sijia Liu, Haiming Chen, Scott Ronquist, Laura Seaman, Nicholas Ceglia, Walter Meixner, Pin-Yu Chen, Gerald Higgins, Pierre Baldi, Steve Smale, et al. Genome architecture mediates transcriptional control of human myogenic reprogramming. *Isience*, 6:232–246, 2018.
50. Joshua Pickard, Cooper Stansbury, Amit Surana, Anthony Bloch, and Indika Rajapakse. Biomarker selection for adaptive systems. *arXiv preprint arXiv:2405.09809*, 2024.
51. Lenno Krenning, Stijn Sonneveld, and Marvin E Tanenbaum. Time-resolved single-cell sequencing identifies multiple waves of mrna decay during the mitosis-to-g1 phase transition. *Elife*, 11:e71356, 2022.
52. Maxim V Kuleshov, Matthew R Jones, Andrew D Rouillard, Nicolas F Fernandez, Qiaonan Duan, Zichen Wang, Simon Koplev, Sherry L Jenkins, Kathleen M Jagodnik, Alexander Lachmann, et al. Enrichr: a comprehensive gene set enrichment analysis web server 2016 update. *Nucleic acids research*, 44(W1):W90–W97, 2016.

53. Zhuorui Xie, Allison Bailey, Maxim V Kuleshov, Daniel JB Clarke, John E Evangelista, Sherry L Jenkins, Alexander Lachmann, Megan L Wojciechowicz, Eryk Kropiwnicki, Kathleen M Jagodnik, et al. Gene set knowledge discovery with enrichr. *Current protocols*, 1(3):e90, 2021.
54. Peter J Schmid. Dynamic mode decomposition and its variants. *Annual Review of Fluid Mechanics*, 54(1):225–254, 2022.
55. Scott Ronquist, Geoff Patterson, Lindsey A Muir, Stephen Lindsly, Haiming Chen, Markus Brown, Max S Wicha, Anthony Bloch, Roger Brockett, and Indika Rajapakse. Algorithm for cellular reprogramming. *Proceedings of the National Academy of Sciences*, 114(45):11832–11837, 2017.
56. Christian von Mering, Martijn Huynen, Daniel Jaeggi, Steffen Schmidt, Peer Bork, and Berend Snel. String: a database of predicted functional associations between proteins. *Nucleic acids research*, 31(1):258–261, 2003.
57. Katja Luck, Dae-Kyum Kim, Luke Lambourne, Kerstin Spirohn, Bridget E Begg, Wenting Bian, Ruth Brignall, Tiziana Cafarelli, Francisco J Campos-Laborie, Benoit Charloteaux, et al. A reference map of the human binary protein interactome. *Nature*, 580(7803):402–408, 2020.

Supporting Information

Contents

1	Introduction	1
2	Results	3
2.1	BRAD Software Architecture	3
2.1.1	Lab Notebook Module	5
2.1.2	Software Module	5
2.1.3	Digital Library Module	5
2.1.4	Planning and Routing Modules	5
2.2	Digital Laboratory	7
2.2.1	Benchmarking Tool Modules	7
2.2.2	RAG Assesment	7
2.2.3	Enhanced Search	9
2.2.4	Building a RAG Database	9
2.2.5	Biomarker Identification and Consensus	10
3	Discussion	11
4	Software Architecture Extended	18
4.1	Base Architecture	18
4.2	Lab Notebook Module	20
4.3	Software Module	20
4.4	Digital Library	22
4.5	Planner Module	24
4.6	Module Selection and Control Flow	24
4.7	Prompt Templates	24
5	Results and Examples	27
5.1	Benchmarking Tool Modules	27
5.2	LAB NOTEBOOK Evaluation	28
5.2.1	Qualitative Example	28
5.2.2	RAG Assessment (RAGAs) evaluation metrics	32
5.2.3	Evaluation Question Dataset	34
5.3	Building a Database	39
5.4	Biomarker Identification Workflow	42
5.4.1	Biomarker Selection from Data using BRAD	42
5.4.2	Biomarker Selection from o1-preview	42
5.5	Advanced Search	46

4 Software Architecture Extended

This section provides an in-depth overview of BRAD’s software architecture. Readers interested in implementation details may consult the BRAD software manual linked: [here](#).

4.1 Base Architecture

Behind each module of BRAD is a framework that standardizes information flow through BRAD, memory, and external resources. The configuration parameters for each module are specified in `config.json`. These default settings can be overridden by supplemental configuration files when instantiating individual BRAD instances. Module specific parameters are further detailed in their respective sections, and see Listing 1, on the next page, for an example set of configurations.

Language Model Integration. The LangChain interface is used for querying LLMs in BRAD. This supports a wide array of LLMs that could be integrated into a BRAD chatbot. For ease of use, BRAD provides a simplified interface to run LLMs hosted by NVIDIA and OpenAI, and also supports locally hosted LlamaCPP.

Output Directory. An output directory is created for each BRAD agent. These output directories are the location where any downloaded database results, new literature, images, or other main results will be placed. Within `config.json`, the `output-directory` parameter allows users to specify where these directories should be created. New directories are placed in the appropriate location and time stamped. The session state is saved within the output directory and can be used to turn on a BRAD agent from the same session or starting point in the future.

Logging. A detailed log is provided within each output directory. The log is a JSON formatted file that tracks the state of BRAD as it responds to each query. It records user inputs, outputs, and the specific actions taken, such as LLM calls, file operations, or the use of other module-specific tools. There are three primary uses for these log files:

- (1) **Clarity:** The log files record every database query, executed software, referenced literature, and other resources used. Users can refer to the log file to understand how BRAD processes an input.
- (2) **Documentation:** BRAD’s documentation generation features, including both code and PDF generation, utilize the log files to detail the process after completion.
- (3) **Debugging:** These logs are useful for identifying errors.

Separate from the log files but also useful for understanding how BRAD responds to a user query, a `debug` flag can be set in `config.json` to have BRAD display each of the steps taken along the way.

Utilities. A large array of utilities have been developed to standardize the interface between the LLM and each module. This includes parsing LLM responses, reading data from files, managing the output directory, and more.

Listing 1. Example Configuration Settings in JSON

```
1 {
2     "log_path": "<path/to/output-directories>/BRAD", // output location
3     "image-path-extension": "images"                // output location for
4     images
5     "RAG": { // LAB NOTEBOOK
6         "num_articles_retrieved": 3, // number of articles during retrieval
7         "multiquery": false,         // retrieval with single user query or
8         // multiple llm generated multiqueries
9         "contextual_compression": false, // using contextual compression or raw
10        // chunks during generation
11        "rerank": false, // reranking documents after retrieval
12        "similarity": true, // search algorithm for selecting document
13        // during retrieval
14        "mmr": true, // one of similarity or mmr must be true
15        ...
16    },
17    "SCRAPE": { // DIGITAL LIBRARY: web scraping from arXiv, bioRxiv, and PubMed
18        "add_from_scrape": true, // add documents to the LAB NOTEBOOK database
19        "max_search_terms": 10 // number of terms to search on the archives
20        "max_articles_download": 10, // how many articles to download
21        ...
22    },
23    "DATABASE": { // DIGITAL LIBRARY: searching enrichr, gene ontology, etc.
24        "max_search_terms": 100 // maximum number of search terms (genes or
25        // other) to query at once
26        "max_enrichr_pval": 0.5, // database specific parameters for search
27        ...
28    },
29    "SOFTWARE": { // SOFTWARE: path to find available software for BRAD to run
30        "py-path": "/home/jpic/BRAD-Tools/",
31    },
32    "display": { // General display parameters
33        "num_df_rows_display": 3,
34        "dpi": 300,
35        "figsize": [ 5, 3 ],
36        "colormap": "viridis",
37        ...
38    },
39 }
```

4.2 Lab Notebook Module

Database Construction and Chunking. Chunking breaks documents into smaller pieces or chunks of texts that can later be used to help an LLM respond to a user query. It is important to divide documents to create more focused text that RAG models can use to generate better responses. There are many ways to split up texts; BRAD employs two methods:

- **Recursive chunking** divides text into smaller chunks based on size and a set of separators, such as periods and line breaks.
- **Semantic chunking** encodes sentences into an embedding space and then applies similarity metrics to determine which sentences should be grouped into the same chunk

Similarity Search v. MMR. Given a user query, BRAD has two mechanisms for retrieving the initial text chunks: Similarity Search and Maximum Marginal Relevance (MMR). In both methods, the user query is embedded as a vector so that similarity metrics, primarily cosine distance, quantify the relevance of chunks of text to the user query. Similarity search compares chunks to the prompt and determines which chunks are most similar to the prompt. This is a straight forward approach to chunk retrieval but suffers from possible redundancies in the chunks which may hamper BRAD’s response. MMR retrieves chunks by finding chunks that are maximize similarities to the prompt and to other chunks based on a parameter λ . The MMR based retrieval selects a more diverse set of chunks, mitigating the redundancy issue.

Multi-query Retrieval. A challenge of retrieval is questions or queries with similar semantic meaning can be worded differently. To mitigate this, multi-query retrieval takes the original user input and uses an LLM to create several different ways of expressing the same query. For each new expression as well as the original user query, the aforementioned retrieval methods are applied to find chunks relevant to the user query.

Compression. Since the retrieved chunks can be unorganized, we use contextual compression to summarize the contexts of each of the selected chunks. This method passes each chunk one at a time, along with the user query, to an LLM with instructions to summarize the retrieved text, highlight the pieces of text relevant to the user’s query, and to remove information not relevant to the user’s query. Since the summaries are often shorter than the chunks themselves, this method can be effective when the context of an LLM is a limiting factor of the generation stage of RAG. While this is the most computationally expensive feature of BRAD’s RAG system, since an LLM call is used for each retrieved chunk, this feature, produced the greatest improvements in the quality of BRAD’s responses.

Reranking. Reranking is a common feature in RAG systems and provides a secondary validation of the relevance of retrieved chunks of text, prior to the generation stage. To maximize relevance and diversity of chunks given to the LLM during generation, we implemented a PageRank based reranker. From the retrieved chunks, we create a weighted graph where the nodes are the chunks of text and the edges are the similarity between the chunks. Then, the PageRank algorithm is used to select the most relevant chunks to be used in the generation phase. By having this secondary filter, an increased number of text chunks can be retrieved from the RAG database without necessarily passing duplicated or irrelevant information to the LLM.

Scalability of Database and Runtime. Prior to runtime of the RAG, creating the vectorized text database can take several seconds to hours, depending on the number of files a user includes. For reference, our database consisting of more than 500 papers and books that consist of roughly 18000 pages was constructed in about 8 hours using 1 CPU and 100 GB of RAM. Although creating the vectorized database takes time, database construction is a one time cost to the user. Querying the database is fast, even for large databases.

4.3 Software Module

BRAD’s SOFTWARE module provides a multistage pipeline for running codes that uses LLMs in conjunction with custom documentation to analyze data and improve the response to a user’s query (see fig. 6).

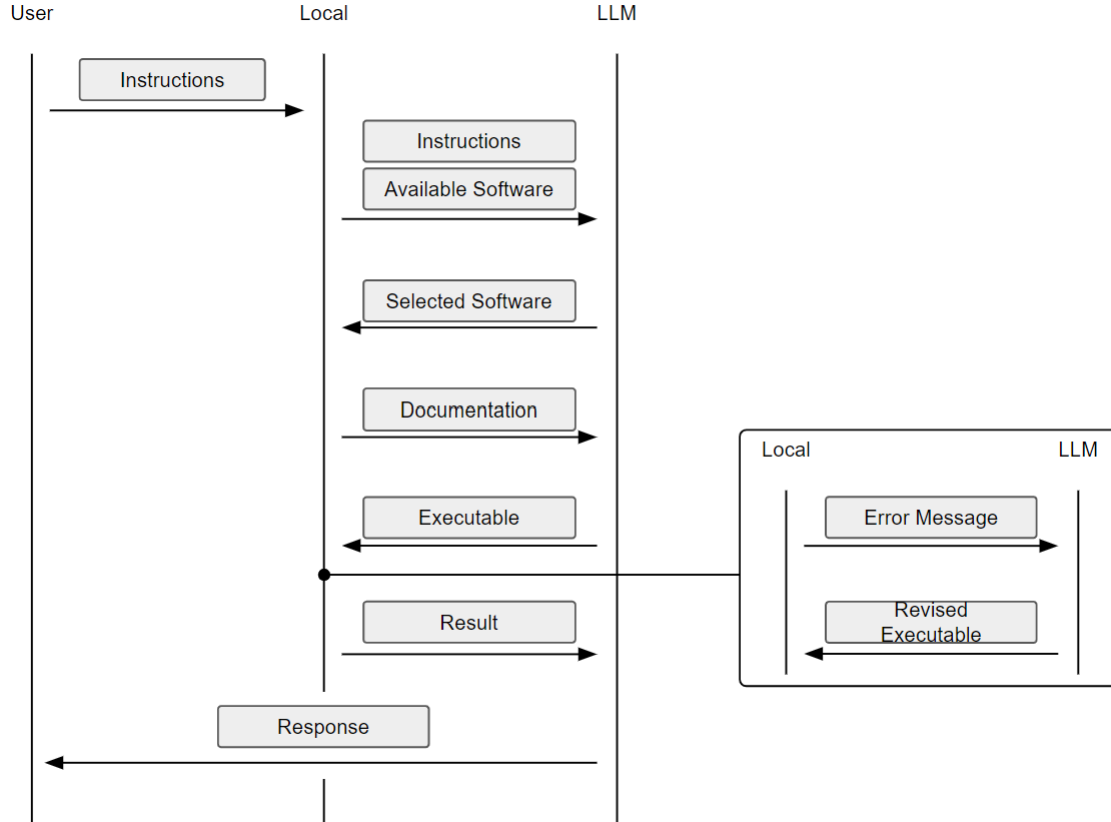


Fig 6. Sequence diagram of SOFTWARE module. (1) Instructions or a query from the user are used to identify that the SOFTWARE module should be run. (2) Summary documentation of the available code are passed along with the user instructions to a LLM. (3) A specific piece of code is selected to run. (4) Detailed documentation of the code is sent to the LLM. (5) The LLM generates a line of code to run the selected software. (6) Error messages and revised LLM generated code are passed back and forth until the code will execute. (7) The printed results of the code are passed to the LLM. (8) The LLM generates a response to be output to the user.

Software Specification. In order for BRAD to run code, the following requirements are enforced:

- (1) **Location:** All code that a user wants BRAD to run must be included in a BRAD specific path.
- (2) **Documentation:** Any code to be run must be well documented, enabling BRAD to understand the software’s purpose and execution process.
- (3) **Output Arguments:** Any code that writes information to disk should accept at least one argument where BRAD can indicate the correct output directory for new files.

These minimal requirements ensure that BRAD can effectively manage, understand, and execute the code while maintaining a consistent and organized workflow. Additionally, these three requirements are intended to allow for ease of adoption of previously generated code.

Software Limitations and Considerations. While useful and effective, this framework for LLM powered execution of code has several limitations and vulnerabilities. It expects bug-free software from the user, which can be challenging to ensure. It can be susceptible to malicious code, if the scripts executed by BRAD are not well curated.

4.4 Digital Library

There is a growing collection of online databases common to bioinformatics workflows that are integrated into BRAD’s DIGITAL LIBRARY module. Here, we outline the integration of several databases and highlight that our architecture allows for the seamless addition of new databases; see fig. 7.

Enrichr. Enrichr is a comprehensive gene set enrichment tool commonly used in many bioinformatics pipelines [41, 52, 53]. We integrated Enrichr into BRAD to leverage gene set annotations, curated pathways, and ontology knowledge to enhance BRAD’s responses to user queries. This allows for enrichment of literature mined gene sets as well as those found in users data directly within the chatbot’s interface.

Gene Ontology. The Gene Ontology (GO) database is a bioinformatics initiative to provide a uniform approach and vocabulary to studying genes and gene products across different species [40, 42]. This database aims to provide a consistent, curated vocabulary and annotations for genes across different species, making it an excellent resource to integrate into BRAD. When gene names, GO terms, or other keywords are identified in BRADs responses to the user, BRAD has the ability to define these GO terms and download charts that diagram their relationships with other genes. Moreover, the GO database provides papers associated with different terms, and these papers can be integrated via the webscraping codes.

PubMed, arXiv, and bioRxiv. While RAG allows users to supplement LLMs with additional information from a text database, the response quality is constrained by the relevance of the literature in the initial database. However, databases may sometimes fail to provide satisfactory answers due to containing irrelevant information or lacking the latest research.

To address this, BRAD incorporates the ability to update its literature database. We implemented searching of arXiv, bioRxiv, and PubMed, but could broaden this process to include additional databases. BRAD can both display search results to the user, link articles, summarize recent abstracts, download and display papers, and embed the new papers into its database. This enables BRAD to form a closes, human in the loop system. User feedback – either in the form of detected poor responses or in the form of explicitly telling BRAD to update – allow BRAD to improve its retrieval knowledge base over time.

When retrieving new documents on PubMed, BRAD only gathers publicly accessible papers. This ultimately limits the documents BRAD has access to, excluding key information and notable findings. To combat this problem, users may directly add their own documents to databases to ensure their favorite findings are accounted for in BRAD’s responses.

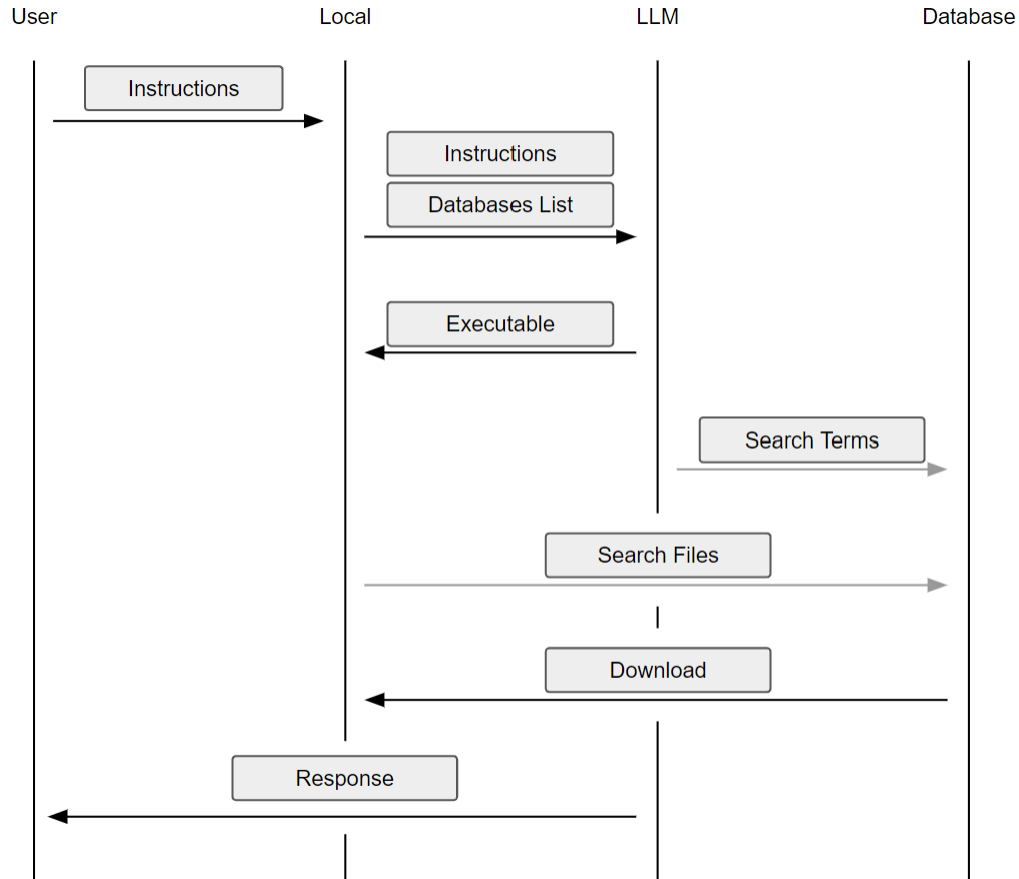


Fig 7. Sequence Diagram of querying the Digital Library. (1) Instructions or a query from the user are used to identify that the **Digital Library** module should be run. (2) The user instructions and a list of available databases are sent to the LLM. (3) The LLM sends executable instructions outlining how to search the database. (4a) The LLM may return a list of search terms, found in the users instruction, chat history, or otherwise generated that can be queried in the database. (4b) The LLM may return a file or field of data within a file that can be queried in the database. (5) Using the API of the selected database and selected search terms, information is downloaded from the database. (6) The LLM generates a response to explain what happened.

4.5 Planner Module

Unlike other modules of BRAD that respond to individual user queries, the **PLANNER** module can handle complex queries that require multiple tools. Given a user input, the scheduler determines a series of smaller tasks that can be executed using one module at a time whose collective output will address the user’s query. These tasks are shown to and edited by a human, after which point the set of tasks are automatically executed; see fig. 8 for a high level schematic of a workflow involving the **PLANNER**.

4.6 Module Selection and Control Flow

A series of mechanisms are utilized by Brad to determine the appropriate module to execute specific tasks or orchestrate complex processes involving multiple modules.

Module Selection with Semantic Routing. BRAD’s direction of a query to the RAG or any other module is a standard control flow problem. To route a users query to the appropriate module or function, BRAD uses a series of semantic routers. To manage the control flow and determine which databases or modules to use in response to a user query, we employ semantic routers to make fast, real time decisions. Semantic routers are prediction or classification models that process user queries to determine the appropriate path for the prompt to flow through the chatbot. At a high level, BRAD uses semantic routing to determine if a query requires fetching data online, access to private data or custom software, or a standard RAG and LLM response. At a lower level, these routers can be used to identify specific function calls and parameters from queries to BRAD.

Adaptive Routing through Reinforcement Learning. Routers are initialized with a series of prompts to guide BRAD’s control flow, and these prompts are refined over time via a simple reinforcement learning procedure. When interacting with BRAD, semantic routers direct user prompts to use different modules, perform online retrieval, or query the text database. Users can force a query down a path or indicate if a prior query was misrouted. As users interact with BRAD, the set of prompts to direct down different paths is recorded. This feedback is incorporated into the router, enabling similar queries to follow the appropriate route in the future.

Control Flow. The router can manage more intricate tasks schedules beyond simple linear execution. For example, the scheduler might first execute RAG to gather initial information and then, based on the quality of the answer, proceed to search the web for additional literature. The **PLANNER**’s combination of an instruction pointer and an instruction list allows for sophisticated control flow patterns, including loops, if/else statements, and other logic.

For instance, consider the scenario where BRAD receives a query addressed at the RAG but has no text database for retrieval. In this case, the scheduler might construct a three stage loop consisting of: (1) download relevant literature to the user’s query and include it in the database; (2) propose an answer based on RAG to the user’s query; and (3) determine if the answer in stage (2) is satisfactory, in which case the chain ends, or if the answer is unsatisfactory, the instruction pointer will return to step (1). In Step (3), Brad uses the LLM at the core of BRAD to evaluate complex control flow decisions and determine if previous outputs in the chain satisfy particular conditions set by the scheduler.

4.7 Prompt Templates

A core component of BRAD is a prompt template library designed to interface with various modules and generate responses for the users. This library of templates contains two distinct classes and is based on recent LLM prompting techniques.

Input v. Output Prompts. We developed two classes of prompts corresponding to the primary uses of an LLM within BRAD. The first class of prompts is designed to transform a user’s query into actions, leveraging the LLM to generate or execute code. This includes tasks such as data or software selection, generating or editing code to execute, or selecting databases and online search terms, among other tasks. In

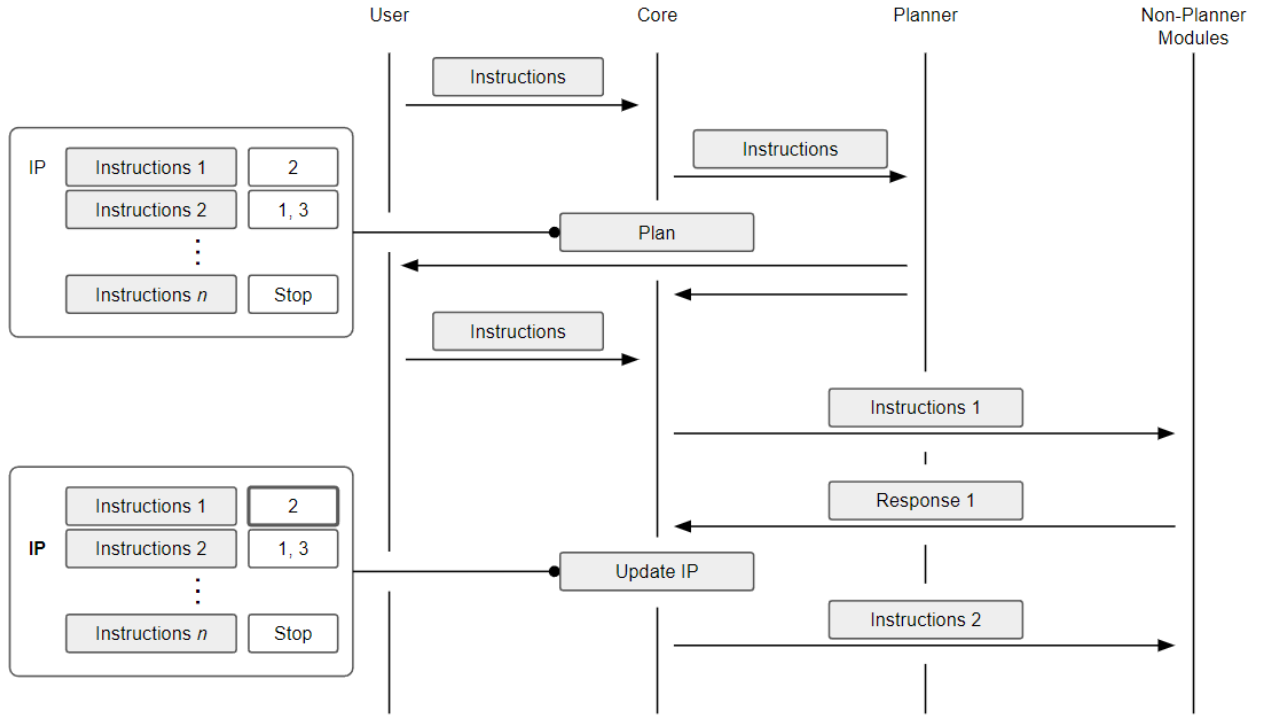


Fig 8. Sequence Diagram for Multi-module Plans with BRAD. Information is shared between four areas: the *user* is a human; the *core* refers to the BRAD's core modules and agent state; the **PLANNER** module; and BRAD's tool modules, which execute the various tasks. (1) A user sends instructions to BRAD. (2) These instructions are routed to the Planner module. (3) The Planner sends back a plan that is shown to the user and saved to BRAD's core/main framework. The plan, shown on the left, consists of a list of instructions that can be executed by each module. Each instruction has a list of possible instructions to execute next, and an Instruction Pointer (IP) is used to manage what to do next. (4) After viewing the plan, the user provides new instructions including (a) returning to step (1) and editing the plan or (b) allowing the code to execute the plan. (5) Instruction 1, starting at the top of the list, is sent to the appropriate module. (6) The module sends back a response. (7) Based on the modules response and the instruction list, the IP is updated. In this case, the IP moves to Instruction 2. (7) Instruction 2 is sent to the appropriate module. This process will continue until the IP is told to Stop.

these prompts, a series of techniques for improving the responses via structured output, few shot prompting, and other methods are used.

The second class of prompts is used to generate output responses for the user. These include the generation stage of RAG, where pieces of text are formatted into a prompt template to improve user responses, prompts used to summarize data resources and code utilized, and prompts for generating output PDF documentation. These prompts also convey specific instructions and often are templated to insert information and data retrieved while executing various modules.

Few Shot Prompting. Few-shot prompting is a technique to provide LLM responses by including a set of example outputs as input to the model. For instance, to execute code with BRAD a set of prompts have been constructed that contain examples on how to construct an appropriate piece of code in either MATLAB or python. These few examples are both built into BRAD's prompt library and can additionally be provided in the documentation of software being run. By providing examples, the variability in response structure as well as the quality of the LLM's output are improved.

Chain of Thought Prompting. Chain of thought prompting is a technique that uses reasoning through examples as input to an LLM to demonstrate how to reason through complex or multistep queries. When asking an LLM to solve a problem, the user could specify in the prompt to solve the problem using a series of steps. In BRAD's code execution program, a series of such prompts are provided. When formulating a function call to execute a script, the LLM is asked to generate an appropriate piece of code based on the user's input, software documentation, and few shot examples of running the code. The output of the prompt is structured to require the LLM to explain its chain of thought and reasoning behind the constructed code.

5 Results and Examples

This section presents extended results and implementation details related to the use cases of BRAD discussed in the paper.

5.1 Benchmarking Tool Modules

This section profiles the use of the LLM in different modules of BRAD. The major cost to the user occurs in terms of (1) runtime and (2) API fees. There is no substantial memory requirements for this system. A user could also elect to run the LLM inference locally, in which case there would be no fees and the time requirements depend on the user’s hardware. The following metrics are evaluated:

- Total Time: the time required to respond to the users query by a single module.
- LLM Time: the time required for the LLM to perform its task within the module.
- Total Cost: the total cost of LLM API fees within a single module. This can represent multiple LLM utilizations.
- LLM Cost: the cost of LLM API fees for the LLM to perform its task within a module.
- Input Tokens: the number of tokens input for a single LLM query.
- Output Tokens: the number of tokens output for a single LLM query.
- Total Tokens: the sum of input and output tokens for a single LLM query.

See fig. 9 and table 2.

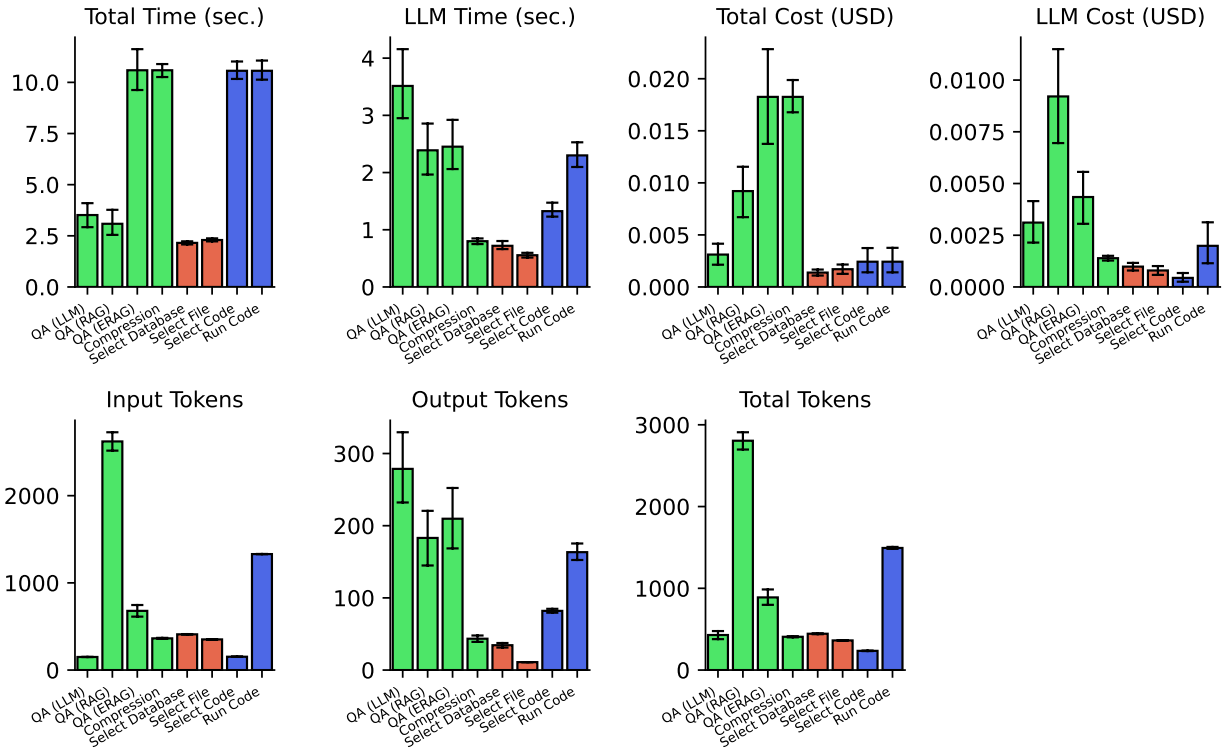


Fig 9. Costs associated with the use of LLM in each module of BRAD are profiled. Each task is color-coded according to its respective module: LAB NOTEBOOK is represented in green, SOFTWARE in red, and DIGITAL LIBRARY in blue.

Table 2. Summary of cost of different tasks performed by LLMs.

Task	Model	Total Tokens	LLM Time (sec.)	Total Cost (USD, $\times 10^{-4}$)
QA (LLM)	gpt-3.5-turbo-0125	303.14 ± 57.22	1.95 ± 0.67	0.0 ± 0.0
	gpt-4o	680.18 ± 146.8	6.65 ± 1.71	0.009 ± 0.002
QA (RAG)	gpt-3.5-turbo-0125	2746.5 ± 340.04	1.67 ± 0.87	0.001 ± 0.0
	gpt-4o	2863.77 ± 367.62	3.11 ± 1.58	0.017 ± 0.003
QA (ERAG)	gpt-3.5-turbo-0125	743.64 ± 202.12	1.5 ± 0.57	0.003 ± 0.0
	gpt-4o	1033.73 ± 372.96	3.41 ± 1.35	0.034 ± 0.007
Compression	gpt-3.5-turbo-0125	403.45 ± 73.0	0.73 ± 0.49	0.003 ± 0.0
	gpt-4o	410.78 ± 77.89	0.87 ± 0.56	0.034 ± 0.007
Call code	gpt-3.5-turbo-0125	1488.21 ± 29.26	2.17 ± 0.52	0.001 ± 0.0
	gpt-4o	1530.75 ± 51.25	3.22 ± 0.74	0.012 ± 0.001
Select code	gpt-3.5-turbo-0125	233.86 ± 7.0	1.25 ± 0.16	0.001 ± 0.0
	gpt-4o	243.5 ± 14.15	1.88 ± 0.71	0.012 ± 0.001
Select database	gpt-3.5-turbo-0125	448.38 ± 29.29	0.78 ± 0.4	0.0 ± 0.0
	gpt-4o	436.19 ± 34.02	0.59 ± 0.23	0.004 ± 0.001
Select file	gpt-3.5-turbo-0125	363.35 ± 1.69	0.47 ± 0.13	0.0 ± 0.0
	gpt-4o	361.25 ± 1.65	0.71 ± 0.05	0.004 ± 0.0
Average	gpt-3.5-turbo-0125	575.45	1.02	0.002
	gpt-4o	631.17	1.53	0.026

5.2 LAB NOTEBOOK Evaluation

5.2.1 Qualitative Example

We provide further qualitative comparisons of responses to open-ended research questions, both with and without the RAG pipeline. We consider the question Q: How can we view Pore-C as a hypergraph?

Using OpenAI’s GPT-4.o mini as the LLM, we equipped BRAD with three different configurations of the LAB NOTEBOOK to test BRAD’s QA capabilities: *vanilla* LLM, a *standard* RAG with no enhancements (i.e. only MMR retrieval and generation), and an *enhanced* RAG with additional features, including MMR, multi-query retrieval, reranking and contextual compression. When BRAD employs a RAG to improve its response, it is supplemented with ten retrieved chunks of text. In our view, the *enhanced* RAG best answers this question.

Despite the fact that this question has a clear cut meaning, it is still quite open ended, allowing the BRAD chatbots freedom to dive into the specifics as much as possible. We deem *vanilla* LLM’s response to be the worst; its discussion of higher-order interactions is repetitive, parroting its own previously generated text. Additionally, although it claims to give perspectives into how to analysis and visualize the data, it completely omits a discussion of visualization techniques.

The *standard* RAG is also repetitive, as seen through repeating information in the first paragraph and the enumerated list of concepts. However, its discussion about hyperedges is more insightful. Even though it also fails to give pointed recommendations about visualization tools, it comments about the capability rather than omitting the discussion altogether.

We believe the *enhanced* RAG best answers the question. It fixes many of the issues of the *vanilla* LLM and *standard* RAG. Specifically, its sections on higher-order interactions and analysis and algorithms add new insights compared to both *vanilla* LLM and *standard* RAG.

Q1: How can we view Pore-C as a hypergraph?

Enhanced RAG

Pore-C is a technique used to study the three-dimensional organization of the genome, particularly in the context of chromatin interactions. To view Pore-C as a hypergraph, we can consider the following aspects:

1. **Vertices and Edges:** In a hypergraph, vertices represent entities, and hyperedges connect multiple vertices. In the context of Pore-C, the vertices can represent genomic loci (e.g., genes, regulatory elements, or specific regions of the genome). The hyperedges can represent the interactions between these loci, where a single hyperedge can connect multiple loci that interact with each other within a specific spatial context.
2. **Higher-Order Interactions:** Unlike traditional graphs where edges connect pairs of vertices, hypergraphs allow for the representation of higher-order interactions. In Pore-C, certain genomic regions may interact with several other regions simultaneously, which can be effectively captured by hyperedges. This is particularly relevant in understanding complex regulatory networks and chromatin architecture.
3. **Data Representation:** When analyzing Pore-C data, one can construct a hypergraph where each hyperedge corresponds to a set of loci that are in close spatial proximity or show significant interaction frequencies. This representation can help in identifying clusters of interacting regions and understanding the functional implications of these interactions.
4. **Analysis and Algorithms:** Viewing Pore-C data as a hypergraph opens up various analytical approaches, such as community detection, clustering, and network analysis, which can provide insights into the organization and function of the genome. Algorithms designed for hypergraphs can be applied to identify key regulatory regions, interaction hubs, and other biologically relevant structures.
5. **Visualization:** Hypergraphs can be visualized using specialized software that can handle higher-order relationships, allowing researchers to explore the complex interactions revealed by Pore-C data in a more intuitive manner.

In summary, conceptualizing Pore-C data as a hypergraph allows for a more nuanced understanding of genomic interactions, accommodating the complexity of higher-order relationships that are inherent in chromatin organization.

Q1: How can we view Pore-C as a hypergraph?

Standard RAG

Pore-C is a technique used to study the three-dimensional organization of the genome, particularly in the context of chromatin interactions. Viewing Pore-C data as a hypergraph can provide a more nuanced understanding of the complex relationships between genomic elements. In a hypergraph, nodes represent individual genomic features (such as genes, regulatory elements, or chromatin domains), and hyperedges represent interactions or relationships that can connect multiple nodes simultaneously. This is particularly relevant for Pore-C data, where interactions are not limited to pairs of genomic elements but can involve multiple elements at once.

Here's how you can conceptualize Pore-C as a hypergraph:

1. **Nodes:** Each node in the hypergraph can represent a specific genomic feature, such as a gene, enhancer, or other regulatory elements.
2. **Hyperedges:** A hyperedge connects multiple nodes that interact with each other. In the context of Pore-C, a hyperedge could represent a chromatin interaction that involves several genomic regions simultaneously. For example, if a gene interacts with multiple enhancers, this relationship can be captured as a single hyperedge connecting the gene and all relevant enhancers.
3. **Multiplicity of Interactions:** Pore-C data often reveals that genomic regions can interact with many others, which is a key characteristic of hypergraphs. This allows for a more comprehensive representation of the complex interactions that occur within the nucleus.
4. **Analysis:** By treating Pore-C data as a hypergraph, researchers can apply various graph-theoretical approaches to analyze the structure and dynamics of chromatin interactions. This can include identifying clusters of interacting regions, understanding the role of specific genomic features in the overall architecture, and exploring how these interactions change under different conditions.
5. **Visualization:** Hypergraphs can be visualized using specialized software that allows for the representation of multi-way interactions, making it easier to interpret the complex relationships inherent in Pore-C data.

In summary, viewing Pore-C as a hypergraph allows for a richer representation of genomic interactions, capturing the complexity of chromatin organization and facilitating advanced analyses of genomic architecture.

Q1: How can we view Pore-C as a hypergraph?

C. GPT-4.o

Pore-C is a technique used to study the three-dimensional organization of the genome, particularly focusing on chromatin interactions. To view Pore-C as a hypergraph, we can consider the following aspects:

1. **Vertices and Edges:** In a hypergraph, vertices represent entities, and hyperedges connect multiple vertices. In the context of Pore-C, the vertices can represent genomic loci (e.g., genes, regulatory elements, or specific regions of the genome). The hyperedges can represent the interactions between these loci, where a single hyperedge can connect multiple loci that interact with each other.
2. **Higher-Order Interactions:** Unlike traditional graphs where edges connect pairs of vertices, hypergraphs allow for the representation of higher-order interactions. In Pore-C data, a single interaction event may involve multiple genomic regions simultaneously, which can be effectively captured by hyperedges.
3. **Complexity of Genomic Interactions:** The genome is not organized in a linear fashion; rather, it exhibits complex interactions that can involve multiple regions. By modeling these interactions as a hypergraph, we can better represent the multifaceted nature of chromatin architecture and the relationships between different genomic elements.
4. **Analysis and Visualization:** Viewing Pore-C data as a hypergraph can facilitate advanced analytical techniques, such as community detection, clustering, and network analysis, which can help identify functional modules or regulatory networks within the genome.
5. **Applications:** This hypergraph representation can be useful for understanding how different genomic regions collaborate in processes such as gene regulation, transcription, and the overall spatial organization of the genome.

In summary, conceptualizing Pore-C data as a hypergraph allows for a more nuanced understanding of the complex interactions within the genome, enabling researchers to explore the functional implications of these interactions in a more comprehensive manner.

5.2.2 RAG Assessment (RAGAs) evaluation metrics

Five metrics were used to evaluate the utility of BRAD’s RAG pipeline [44]:

- **Faithfulness:** This quantifies the consistency between BRAD’s answer and the literature.
- **Answer Relevance:** This evaluates if the response is applicable to the particular question.
- **Context Precision:** This quantifies the ranking of the chunks BRAD uses to formulate its response relative to the ground truth.
- **Context Recall:** This measures the use of context in BRAD’s response.
- **Answer Correctness:** This is the F1 score of claims that can be verified from the ground truth.

Four components are used to calculate each of these metrics: question, response, context, and ground truth. For an individual question, the context used to construct the question and the answer produced by BRAD’s RAG are split into small, verifiable claims. Then, statements from BRAD’s answer can be compared in a one for one fashion to find claims indicated by BRAD substantiated or not substantiated in context of the literature. This approach allows us to construct a standard confusion matrix where:

- **TP:** True Positives are statements that are present in both the ground truth and the generated response.
- **FP:** False Positives are statements that are present in the generated response but not the ground truth.
- **TN:** True Negatives are statements that are present in the ground truth but not in the generated response.
- **FN:** False Negatives are statements that are present in the ground truth but not in the generated response.

Faithfulness. Faithfulness refers to the factual consistency of the generated response to a question against the given context. Since this metric is entirely based on the context given, if the context given is faulty, then a factually incorrect answer can still be considered faithful. Each of the generated claims is then checked against the given context to determine if the claim can be inferred from context.

$$\text{Faithfulness} = \frac{\text{Number of Claims from Response inferred from Context}}{\text{Number of Claims from Response}}.$$

Answer Relevance. Answer Relevance is a measure of how closely the generated response and given prompt are related. Answers that are incomplete or contain redundant information are typically assigned lower scores. We use the following formula to calculate answer relevance

$$\text{Answer Relevance} = \frac{1}{N} \left(\sum_{k=1}^N \frac{e_{g_k} \cdot e_o}{\|e_{g_k}\| \|e_o\|} \right),$$

where N is the number of generated questions, e_{g_k} is the vectorized embedding of the k th generated question from a generated response, and e_o is the vectorized embedding of the original question.

Context Precision. Context Precision measures how highly ranked ground truth relevant chunks are compared to other chunks in the context of the response. We use the following formula to calculate context precision

$$\text{Context Precision} = \frac{1}{\text{Number of Relevant Chunks}} \left(\frac{\sum_{k=1}^N \text{Number of Relevant Chunks up to position } k \cdot \delta_k}{k} \right),$$

where δ_k is 1 if the chunk is relevant and 0 otherwise. Although this does not directly prove anything inherently about the generated response, it does serve as a metric to demonstrate the effectiveness of page rank reranking.

Context Recall. Context Recall measures how related the retrieved context compares to the ground truth. It is calculated similar to faithfulness.

$$\text{Context Recall} = \frac{\text{Number of Ground Truth claims that can be linked to the context}}{\text{Number of Ground Truth claims}}.$$

Answer Correctness. Answer Correctness measures how the generated response compares to the ground truth. It is calculated by averaging the semantic similarity between the generated response and the ground truth and factual similarity. The semantic similarity is calculated by embedding both the ground truth and the generated response into the same embedding space and calculating the cosine similarity between the two vectors (similar to answer relevance calculations). The factual similarity is calculated through the F1 score

$$\text{F1} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})}.$$

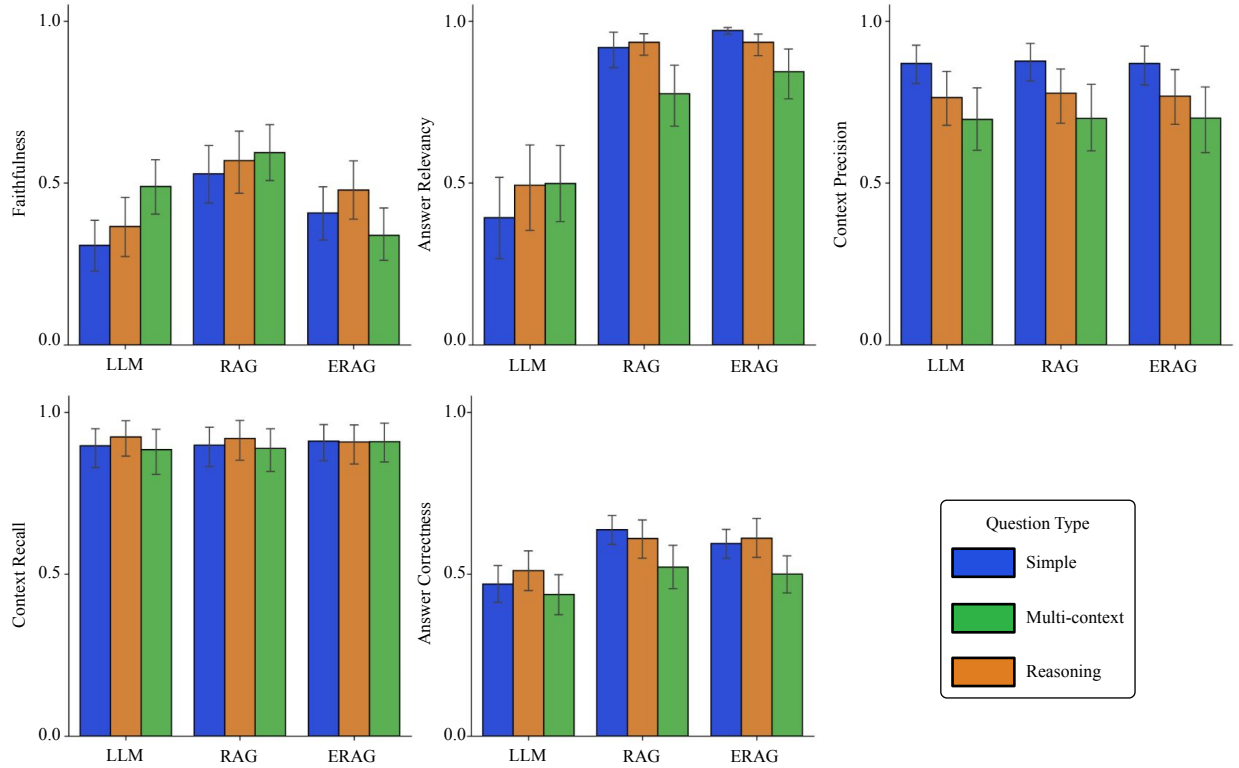


Fig 10. Visualizations of the RAGAs metrics using BRAD. The base LLM in all three BRAD bots is OpenAI GPT 4.o-mini, and the enhancements used in ERAG are Multiquery, Contextual Compression and Reranking.

5.2.3 Evaluation Question Dataset

The following questions were used to evaluate our LAB NOTEBOOK module:

1. How do B cells undergo affinity maturation during the germinal center reaction?
2. How is the timely clearance of neutrophils critical for the resolution of inflammation?
3. How did Levinson and Smith use the method of proving the divergence integrated along any limit cycle to establish uniqueness in their study of limit cycles?
4. What steps were involved in the PacBio library preparation for the American Gut Project samples?
5. How does Morse theory relate to the number of critical points in a Morse function on a manifold?
6. How do ectopically expressed transcription factors interact with endogenous components of recipient cells' transcriptional network in induced lineage reprogramming?
7. How are the kinetic parameters for each gene estimated using nonlinear least-squares methods in single-cell kinetics data?
8. How has in vivo reprogramming with viral vectors been used as a strategy for regeneration of the central nervous system in mice?
9. How does Batch Normalization apply a linear transformation to the output mean in the context of OrthDNNs?
10. How is protein identification confirmed in the antibody characterization process?
11. How does the concept of the unit interval relate to the Kraft inequality in coding theory?
12. How is the mean radius of gyration (R_g) of chromosome 12 TADs calculated and presented in the data analysis?
13. How many critical points does the height function on the n -sphere have?
14. How do B cells gain access to the GC reaction and what role does inter-clonal competition play in this process?
15. What are some challenges in studying cell-cycle dynamics, particularly in terms of protein levels and post-translational modifications?
16. How do the highly non-linear dynamics of cooperative and competitive regulation of transcription factors impact the differentiation efficiency in cell reprogramming experiments?
17. What is the significance of periodic network connectivity in the analysis of the discrete-time consensus algorithm?
18. How did the recognition of the power of computational techniques contribute to the growth of network analysis and network science?
19. How can physicians claim CME credits for reading content from the Journal of Investigative Dermatology?
20. How does the inclusion of temporal information in the system impact the complexity and effectiveness of the detector-generation process in immune-inspired network IDS?
21. How is β -catenin involved in the promitogenic effect of TGF- β ?
22. How is IRF2 editing associated with keratinocyte differentiation and skin barrier formation?
23. How did the use of a "physiological" buffer impact the activity of RNA polymerases in cells during the experiment?
24. How does the spatial organization of human chromosomes change during the cell cycle?
25. How does the substitution of serines with glutamate in cyclin B1 affect its rate of nuclear import in *Xenopus* oocytes?
26. How is nuclei isolation performed for skeletal muscle tissue samples?
27. How can Hodge Decomposition be used to identify tie strength with high accuracy?
28. How does CTCF binding factor play a role in creating metastable states related to DNA methylation?
29. How did Lou Pecora's research interests shift from solid-state physics to chaos theory while working at the U.S. Naval Research Laboratory in Washington?
30. How are nucleoids released from *Escherichia coli* and what is their structure?
31. How does Cyclin E control S phase progression during *Drosophila* embryogenesis?
32. How does the experimental efficiency impact the quality of in silico contact maps in Hi-C, SPRITE, and GAM experiments?
33. What are some of the mathematical problems discussed by David Hilbert in his lecture at the Second International Congress of Mathematics in 1900?
34. How can sequence count data be used in differential expression analysis?
35. How were peaks called in the ChIP-seq data analysis, and what role did input controls play in this process?
36. How is the level of nucleosome occupancy at the TSS related to accessibility during mitosis?
37. How can the parameters of an affine function be obtained by evaluating the function at specific vectors?
38. How does over-fitting affect the RMS prediction error in data fitting models?
39. How does DEC205 targeting affect the interactions between GC BCs and TFH cells in the context of antigen presentation?

40. How do nearby trajectories behave close to the chaotic attractor in the Lorenz system?
41. How do deep neural networks contribute to the mapping of gene expression levels from WSIs in the study of tumor heterogeneity and survival outcomes in breast and lung cancer?
42. How does the spectral dimension of a simplicial complex affect the possibility of frustrated synchronization in a network?
43. What is the significance of transforming a matrix A into real canonical form B in the context of solving differential equations?
44. What is the significance of matrix inverses in solving linear equations and least squares problems?
45. How does supervised learning typically approach binary classification tasks?
46. How can data-driven algorithmic approaches help in predicting transcription factors for directed cell conversion?
47. Why were researchers shocked to find that the longest REM episodes occurred near the beginning of sleep, rather than near the end?
48. What was explored in the study regarding the temporal program of gene expression in human fibroblasts in response to serum?
49. How are homomorphism numbers calculated in the context of randomly weighted graphs?
50. How does the CalCB method correct for known Hi-C biases and CNV in the estimation of Hi-C contact matrices?
51. How can weights be optimized to minimize downside risk for a given target annualized return?
52. How can a higher-order tensor be regarded as a formal representation of a linear mapping between different spaces, such as matrix and vector spaces, based on the concept of an isomorphic link?
53. How is haplotype imputation utilized in single-cell chromatin conformation capture?
54. How are vector fields defined and approximated in the context of integrability?
55. How are local connectivity patterns represented in hypergraphs?
56. How does Edge PageRank differentiate itself from traditional centrality measures in networks?
57. How does T2T-CHM13 improve variant calls compared to GRCh38 in the context of the 1KGP datasets?
58. How has the direct conversion of various types of cells been successfully achieved and what are the potential applications of this method in medical treatment and research?
59. How does the local track-correction module improve tracking performance in EllipTrack?
60. How do binary relationships affect complex topologies in network science?
61. What happens to most cells in *Drosophila melanogaster* embryos after entering G1 phase?
62. What causes two Hopf bifurcation points to merge in degenerate cases?
63. How does a monotone subsequence in (x_n) in \mathbb{R} imply a convergent subsequence for every bounded sequence in \mathbb{R} ?
64. How does a spiral scaffold affect chromosome loop formation during cell division, especially in relation to inner and outer loop sizes in various models?
65. How are eigenvalues of a square matrix affected by adding a matrix with all entries as 1?
66. How do dominant-negative mutations contribute to genetic dominance in cancer development despite other initiation mechanisms?
67. How do germinal centers aid immune response coordination and high-affinity antibody generation, considering TFH cell movement, GC reuse, and GC B cell clonal restriction?
68. How do Gata4, Mef2c, and Tbx5 affect induced cardiac transcriptional reprogramming?
69. What concept explains cascades of period doubling bifurcations in mathematical ecology models like the logistic map and Ricker model, discovered by Feigenbaum in discrete-time dynamical systems?
70. How does a monotone subsequence in a bounded sequence in \mathbb{R} relate to the Bolzano-Weierstrass Theorem?
71. How do manifolds contribute to proving saddle equilibrium states in dynamical systems?
72. What controls cell cycles in *Drosophila* embryogenesis, coordinating phases with growth and G1 arrest until larva hatches?
73. What role does Myc activity play in apoptosis and oncogenic risk in Burkitt's lymphoma?
74. How does a magnetic field affect the supercurrent behavior observed by experimentalists studying tunneling supercurrents?
75. How has math influenced immunology, especially with help from physicists?
76. How does the requirement for adjacent edges to share two nodes in the random walk process affect its representation and Markovian nature, changing the use of traditional graph and matrix techniques for analysis?
77. How do plasma cells differ from effector cells in humoral immunity roles, given the complex regulation mechanisms at cellular and molecular levels?
78. How can feature mappings enhance regression models with dimension reduction and standardized features?
79. How can protein expression in cells be controlled through growth factors availability or production in other cell types?

80. How do post-translational modifications affect E2F1 stability and function in response to DNA damage like cisplatin?
81. How do positive eigenvalues in a matrix connect to Sylvester's Law of Inertia and the span of eigenvectors for positive eigenvalues in the transformed matrix?
82. How did Gemma Frisius propose longitude could be determined, as pursued by Huygens and others?
83. How to calculate minimal state variables for a given transfer function in linear dynamical systems?
84. How can campaign strategies be influenced by clustered voter representatives, and what is the role of k ?
85. How does finding a compatible similarity measure benefit workload and efficiency in stochastic consensus clustering?
86. How does the spindle assembly checkpoint help ensure correct sister-chromatid alignment during mitosis?
87. How does V(D)J recombination impact allelic exclusion in B-cell development?
88. How do transcription factors affect cardiac reprogramming through co-occupied binding sites and synergistic effects of Tbx5 and Gata4?
89. What is the initial mechanism that holds sister chromatids together by intertwining duplicated DNA molecules at adjacent replication forks?
90. What technological innovation enabled sound and image transmission in the 20th century, leading to advancements in electronics?
91. How does Ime2 stimulate Ndt80 in yeast for meiosis I and G1/S progression?
92. Which locus in the $(\operatorname{Re} \lambda, \operatorname{Im} \lambda)$ -plane indicates the Neimark-Sacker bifurcation with a stable closed invariant curve outside the circle?
93. How does Mengerian property connect to hypergraphs in graph theory?
94. How does AEP relate to entropy in information theory when dividing sequences into typical and nontypical sets based on sample entropy?
95. What kind of trajectories in the restricted three-body problem return close to their initial state?
96. How is Collagenase IV used in iPS cell dissociation for culture before transferring to a new dish on SNL feeder cells?
97. How was the "global" category of smooth manifolds and maps created from the "local" category of open sets in Euclidean space and smooth maps?
98. What makes t-SNE unique in visualizing high-dimensional data effectively?
99. How does Ig gene rearrangement in B-cell development relate to Ig gene allelic exclusion regulation?
100. How is Tucker decomposition connected to higher-order PCA using factor matrices and a core tensor?
101. How does RIG-I activation by RN7SL1 impact cGAS-STING signaling in metastatic cells?
102. How is the preselection productive HCDR3 length distribution approximated to calculate the percentage of initial productive IgH rearrangements removed during B cell development?
103. What stress-response system in mammalian cells prevents excessive cell proliferation by activating p53 in response to overproduction of key mitogenic signaling proteins?
104. How does RIG-I activation by RN7SL1 impact cGAS-STING signaling in cancer cells?
105. How does serum stimulation affect DNA replication timing and gene expression in fibroblasts?
106. How many common neighbors do $(1,1)$ and $(2,2)$ have in a SRG?
107. What was discussed by Laemmli about metaphase chromosome bands?
108. What spatial changes occur in Hi-C maps as cells progress from prometaphase to metaphase in mitosis?
109. What's the condition for a map's derivative to be an immersion between smooth manifolds?
110. What concept represents non-stationary, periodic solutions in dynamical systems and why are they significant in nonlinear dynamics?
111. How does community modularity change with τ values in time-evolving analysis, shown by a decreasing trend from 0.70 to 0.36?
112. How do high-order interactions affect diversity in ecological communities in simulations with different interaction strengths and species numbers?
113. What was the AS change in FOXP1 gene at day 10 post neural induction and its impact on exon 18 inclusion in H9 hESCs?
114. How does aligning subjective and objective probability distributions address model misspecification concerns in rational expectations models?
115. How do miRNAs affect E2F1 protein levels in cells with 3'UTR reporter due to c-myc and E2F1, considering transcript expression in senescent populations?
116. How do pigmented cells and chromatin affect skin patterns in zebrafish?
117. How can vectors model cash flow movement in finance?

118. How do p53 activators affect CDK2 activity before mitosis in relation to p21 expression?
119. How do Fos and Jun proteins interact in wound healing, and what role do HIF factors play in the hypoxic response to tissue injury?
120. How can genetic analysis help prioritize therapy for cancer patients without actionable oncogene mutations, considering gene signature overlap with specific cell types in different body regions?
121. What roles do cdcgenes play in cell cycle control, especially in mitosis entry and S phase replication origin activation?
122. What role do homoclinic orbits play in bifurcation diagrams in curve dynamics with limit cycles and saddle points?
123. What components form the inner kinetochore in higher eukaryotic cells and how do they aid in chromosome attachment and tension during cell division?
124. What are the implications of recent research on converting glial cells, like astrocytes, into neurons for brain repair therapies?
125. What's the BSC capacity with crossover probability for a coding scheme with two 3-length codewords and an average error probability of 0.285?
126. How does Morse theorem affect density of Morse functions in space of smooth functions on a manifold?
127. What defines a true somatic stem cell in terms of self-renewal and progeny generation, considering differentiation and lineage commitment in Waddington's model?
128. How does the cell cycle affect epigenetic marks for memory, considering mitosis and S-phase impact on gene expression and chromatin?
129. How do Wnts affect somite patterning in myogenesis?
130. How do Wnt proteins affect somite cell commitment?
131. How do cells in an epithelium maintain adhesive contacts with neighbors during division, and how does this relate to junction formation and cell cleavage in cytokinesis?
132. How is transversality defined in dynamical systems with periodic orbits and hypersurfaces?
133. How do basic fibroblast growth factor-impregnated gelatin microspheres affect fibroblast migration in 3D wound healing?
134. How is the adjacency matrix used in link prediction algorithms in different fields?
135. How do Gata4, Mef2c, and Tbx5 affect induced cardiac myocyte reprogramming and heart repair?
136. Which company provided the artificial dura for the primate BMI task?
137. How to simplify integro-differential equations using real canonical form for an $n \times n$ matrix?
138. Can you provide examples of research studies on dynamical processes in complex networks beyond pairwise interactions?
139. What is Herman's theorem's role in analyzing Morse-Smale vector fields with the Poincare-Bendixson theorem?
140. How do algorithms for integer multiplication use polynomial rings and tensors?
141. How does LZ algorithm efficiency impact file compression in terms of bits for pointers and match lengths in LZ77?
142. How do tangent vectors and constants characterize an expanding map on a compact Riemannian manifold, and how does this relate to stability in topological spaces?
143. How are external circuit parameters chosen to place complex-conjugate poles slightly right of $j\omega$ -axis in a multiterminal device circuit?
144. How do V, D, and J segment rearrangements impact antibody and T cell receptor encoding in the immune system?
145. How can a cell create lasting memory of a response using transcription and DNA replication?
146. How does TF over-expression affect network dominance in the omnigenic model, considering simplified regulatory networks and TFs' role in cellular reprogramming?
147. How does a graphon relate to the node set $[0,1]$ in terms of structure and connectivity?
148. What is the role of the Nbs1 subunit in activating ATM through the MRN complex in response to double-strand breaks?
149. What are the key features and practical uses of the Watts model in social contagions on networks, considering its relationship with Granovetter's threshold models and the incorporation of simplicial complexes in modeling transmission dynamics?
150. What happens to the fixed points of period-seven cycles as r changes in one or two-parameter bifurcations?
151. How does vertex-transitivity demonstrate the relationship between a graph's structure and its automorphisms, especially in Cayley graphs and the lexicographic product?
152. How does Cyclin B1-Cdk1 complexes affect CDK2 activity post-mitosis in proliferating cells?
153. How is the two-parameter bifurcation diagram constructed for a system with codim 1 and codim 2 bifurcations of equilibria, limit cycles, and homoclinic orbits?
154. How to reconstitute functional modules in vitro using gene expression and morphology from various cell types?
155. How do centrosomes help with sister chromatid bi-orientation during spindle assembly?

156. How does scRNA-seq compare to snRNA-seq in studying transcriptional reprogramming and chromatin interactions?
157. How does removing ECB sequences affect Start synchronization in yeast cells, considering growth rate differences?
158. Which fellowship organization supported the research at the Yale Stem Cell Center Genomics Core Facility?
159. How do factors contribute to hiHeps induction for drug metabolism?
160. How do aged HSCs compare to younger HSCs in terms of regenerative potential, proliferation, and lineage tracing data?
161. How is the Edge PageRank equilibrium point represented in a dynamical system, considering node centrality and random walks?
162. How can S_H interference in V_H - J_H interval prevent secondary recombination events on the same Ig allele?
163. How can periodic solutions be shown under Hopf conditions with Lyapunov-Schmidt reduction and Poincaré normalization?
164. How does lateral inhibition affect feather primordia spacing in chicks, considering growth and regeneration in pattern formation?
165. How does the lasso penalty promote sparsity in regression for neuroimaging data?
166. How is the Laplace operator shown in TT-format for tensors in linear operators and elliptic PDE solutions?
167. What advantage does CD9 offer in isolating murine hematopoietic stem cells?
168. How is a rewriting event in GNA models defined in relation to node mapping and Kronecker matrix sampling?
169. How is fresh frozen tissue fixed with Formaldehyde for immunofluorescence staining in mouse embryo tissue?
170. What causes unexplained stock market fluctuations attributed to decentralized networks and herd behavior?
171. What causes motor neuron degeneration in ALS and how do interactions with glia contribute?

5.3 Building a Database

Typically, we use our prior knowledge and publications of interest to construct a literature database for BRAD. However, the web scraping module makes it easy for BRAD to construct a new database aimed at a particular topic area or question.

To demonstrate the utility of the RAG and web scraping features from the LAB NOTEBOOK and DIGITAL LIBRARY modules respectively, we tested BRAD's ability to respond to questions without a pre-existing database. BRAD was initially provided an empty database. Through alternating between RAG and web scraping, following fig. 11, BRAD was able to adequately answer complex research questions by retrieving relevant information online.

For instance, consider the question: "What are single cell foundation models?" Initially, BRAD has no information on this topic. Responding with only the RAG pipeline, BRAD's core ROUTER module correctly identified a lack of information on this topic and triggered a web search. BRAD then proceeded to perform a web search to gather recent research papers and articles relevant to single cell analysis and foundation models. Equipped with the new information, BRAD's answer to the initial question is improved and is able to discuss this research area, providing the names of several notorious single cell FMs and a discussion of their use cases.

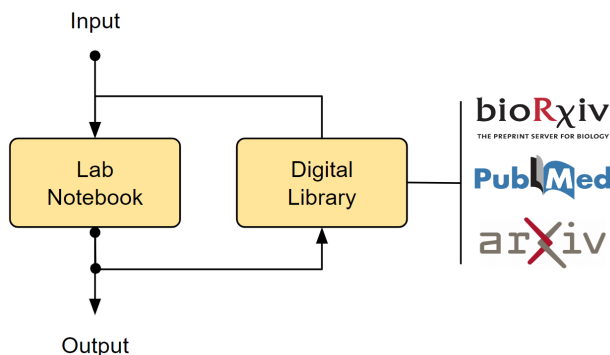


Fig 11. To answer a question with an empty or poorly aligned database, BRAD can iteratively use the Web Search and RAG modules. If the RAG output is assessed as inadequate, the Web Search module retrieves relevant research literature to enhance the answer. This loop may be repeated until the answer to the user's question improves.

Question: What is a single cell foundation model?

RAG: Iteration 1 (empty database)

Based on the provided source and content, the initial database is empty, and I couldn't find any information regarding "single cell foundation models." I am confident that I cannot provide an accurate answer to this question with the current information available.

Web Search

arXiv search terms: "single cell foundation models", "machine learning for single cell data analysis"

RAG: Iteration 1 (populated database)

Single cell foundation models are large-scale pretrained deep language models used for analyzing and annotating single-cell RNA sequencing (scRNA-seq) data. They can help improve the accuracy and speed of cell type annotation and identification in various research areas. Examples of these models include scBERT and a method for ultra-fast cell-type identification using specific marker combinations from single-cell transcriptomic data. These models are used to facilitate queries and analysis of large cell atlas databases, such as the Tabular Muris and Tabular Sapiens, which contain single cell transcriptomic data for mice and humans, respectively.

Additionally, single cell foundation models can utilize transfer learning to enable predictions in network biology and gene regulation, making them valuable tools in understanding cell behavior and interactions in various biological processes. However, these models should be used to aid in basic scientific research and generating or validating mechanistic hypotheses, rather than providing definitive answers.

Web Search

arXiv search terms: "single cell machine learning", "deep learning for single cell analysis"

RAG: Iteration 2 (populated database)

Single cell foundation models are artificial intelligence models designed to analyze and understand data from individual cells. They are constructed using single-cell transcriptomic data, which measures the expression levels of genes in individual cells. These models utilize machine learning algorithms, such as transformers, to learn gene-level and cell-level representation. They can accurately categorize known cell types and have the potential to uncover new biological insights in single-cell RNA sequencing data. Single cell foundation models are used in genomics and multi-omics research, showing promise in various downstream tasks, such as identifying cell types and states, discovering novel cell populations, inferring gene regulation networks, and more. Examples of single cell foundation models include scGPT, Geneformer, and scFoundation. They are a growing field of study, with ongoing advancements expected as single-cell technologies and omics sciences continue to develop.

Question: What is the role of MYOD in Weintraub's cellular reprogramming experiment?

RAG: Iteration 1 (empty database)

Based on the information provided in the initial database, I don't have any specific data about Weintraub's cellular reprogramming experiment or the role of MYOD in that experiment. After searching the literature and documents from a text database, I found that Weintraub's experiment aimed to reprogram fibroblasts into myoblasts using the MyoD gene. MyoD is a myogenic regulatory factor that plays a critical role in muscle cell differentiation and development. It reprograms fibroblasts to a muscle cell fate by binding to specific muscle-specific DNA elements and regulating the expression of muscle-specific genes. However, I couldn't find detailed information on the exact role of MYOD in this experiment. It's recommended to consult the original research article for more comprehensive insights.

Web Search

PubMed search terms: "MYOD", "Weintraub", "cellular reprogramming", "transcription factor", "myogenic differentiation", "muscle cell development", ...

RAG: Iteration 2 (populated database)

MYOD is a transcription factor that plays a crucial role in Weintraub's cellular reprogramming experiment, where it successfully reprogrammed fibroblasts into skeletal muscle cells when expressed alone. MYOD initiates cell reprogramming by contacting E-boxes within closed chromatin and interacting with various regulatory layers such as nucleosome density, histone post-translational modifications, and chromatin folding. Additionally, MYOD can interact with other factors, such as the basal transcription machinery and the transcription elongation factor b, to co-regulate gene expression.

In the context of Weintraub's experiment, MYOD's co-activation involves interactions with the epigenetic machinery. MYOD's binding and activity can be modulated by other factors, including those indirectly recruited by MYOD to its target sites. Furthermore, MYOD makes use of long noncoding RNAs (lncRNAs) in cellular reprogramming. For instance, LncMyoD plays a role in the first step of MYOD-mediated trans-differentiation by interacting with MYOD across the genome and increasing chromatin accessibility at the co-occupied sites. LncMyoD is essential for the differentiation of muscle precursors and the myogenic conversion of fibroblast cells, but it is not needed for the differentiation of already committed myoblasts. These interactions and regulatory roles of MYOD highlight the complexity of the mechanisms underlying cellular reprogramming and the importance of understanding the molecular dynamics of transcription factors and their interacting partners in this process.

5.4 Biomarker Identification Workflow

5.4.1 Biomarker Selection from Data using BRAD

The workflow for biomarker identification, designed to select key genes from time series genomics data as detailed in [50, 47], consists of two steps:

- (1) **Model Fitting:** A dynamic model is fit to time series data.
- (2) **Sensor Selection:** Observability analysis is used to identify the best sensors, which are designated as biomarkers.

In accordance with [50], BRAD fit dynamic models utilizing external software for Dynamic Mode Decomposition (DMD) [54] and Data Guided Control (DGC) [55]. These models of dynamics are written as

$$\begin{aligned}\mathbf{x}(t+1) &= \mathbf{A}(t)\mathbf{x}(t) \\ \mathbf{y}(t) &= \mathbf{C}(t)\mathbf{x}(t),\end{aligned}$$

where $\mathbf{x}(t)$ is the gene expression at time t , $\mathbf{A}(t)$ is a linearization of the dynamics at time t learned from DMD or DGC, and $\mathbf{y}(t)$ are the biomarkers, and $\mathbf{C}(t)$ is the measurement matrix that selects the biomarkers. The goal of the observability analysis is to design $\mathbf{C}(t)$ that selects optimal biomarkers $\mathbf{y}(t)$ at time t .

BRAD then used software to solve the following observability optimization problems:

$$(1) \quad \max_{\mathbf{C}(t)} \sum_t \mathbf{y}(t)^\top \mathbf{y}(t) \quad \text{and} \quad (2) \quad \min_{\mathbf{C}(t)} J(\mathbf{G}),$$

where in (1), the aim is to maximize the output energy, and in (2) \mathbf{G} is the observability Gramian matrix and $J(\cdot)$ can be several different measures of observability. Each optimization problem is formulated to select unique biomarkers for each time point; however, in practice, it can be desirable to select fixed biomarkers throughout time.

5.4.2 Biomarker Selection from o1-preview

To compare the utility of BRAD versus the recently introduced **o1-preview** chatbot from OpenAI, we queried the **o1-preview** model, asking it to select biomarkers on Human foreskin fibroblasts, the cells studied in [48, 49].

The principal challenge for biomarker selection with **o1-preview**, as opposed to BRAD, is finding an appropriate method to format the input data to the agent. Below, we are three ways we queried the **o1-preview** model to select biomarkers:

- (P1) What are the most important biomarkers for: Human foreskin fibroblasts?
- (P2) What are the most important gene biomarkers for: Human foreskin fibroblasts?
- (P3) We are studying a Human foreskin fibroblast cell line from a previously published paper. Here is the abstract of the paper: [insert abstract from [49]]. What are the most important biomarker genes for this cell line?

Responses to these three prompts are shown on the following pages.

These responses are well suited for a reader unfamiliar with human fibroblasts or the problem of biomarker selection. For instance, all three prompts provide the user with information about Vimentin and Fibronectin, two well known markers. Similarly, each response places the list of biomarkers in context of how human foreskin fibroblasts are commonly studied or details of the research paper.

However, these results are not particularly useful to the practitioner interested in using the biomarkers in an experiment, either *in silico* or in the lab. Much of the provided information in **o1-preview**'s responses is likely known to researcher studying this cell line. Moreover, the responses are not based on either (1) the particular data available to the user or (2) transparent methods for why these markers are selected. Additionally, practical considerations such as the availability of particular genes or chemical species being measured in the current dataset or possibility to measure these biomarkers is not considered in **o1-preview**'s responses. The biomarkers selected by BRAD, according to the methods outlined in §SI section 5.4.1, are able to mitigate these issues by providing transparency in the selection approach and results tailored to the users data, while still using capabilities of the LLM.

o1-preview

Human foreskin fibroblasts (HFFs) are a type of primary cell derived from the dermal layer of human foreskin. These cells are widely used in scientific research, particularly in studies related to cell biology, wound healing, aging, cancer, and tissue engineering. Biomarkers for these cells are important to identify their cellular characteristics, stage of development, and health status. Here are some of the most important biomarkers associated with human foreskin fibroblasts:

1. Fibroblast-Specific Protein Markers:

- **Vimentin:** A type of intermediate filament protein that is widely expressed in mesenchymal cells, including fibroblasts. It helps to maintain cell integrity and is used as a general marker for fibroblast identification.
- **Fibroblast-Specific Protein 1 (FSP1 or S100A4):** FSP1 is another key marker for fibroblasts, used to distinguish them from other cell types.
- **Procollagen I:** Since fibroblasts are known for producing extracellular matrix proteins like collagen, procollagen I is often used as a marker to assess their function.

2. Extracellular Matrix (ECM) Proteins:

- **Collagen Type I and III:** Collagen is the most abundant protein in the extracellular matrix of connective tissues. HFFs synthesize and secrete collagen types I and III, making them important biomarkers of fibroblast activity and ECM production.
- **Fibronectin:** Another key ECM protein, fibronectin is involved in wound healing and tissue repair. It is a marker for fibroblasts engaged in matrix formation.
- **Elastin:** Although expressed in lower quantities compared to collagen, elastin production is important in fibroblasts for providing elasticity to tissues.

3. Cell Proliferation and Senescence Markers:

- **Ki-67:** A nuclear protein that is expressed in actively proliferating cells. Ki-67 is often used to evaluate the proliferative capacity of HFFs in culture.
- **p16 and p21:** These cyclin-dependent kinase inhibitors are biomarkers for cellular senescence. As fibroblasts age, their expression of p16 and p21 increases, marking the transition from proliferation to a senescent state.
- **β -Galactosidase (SA- β -gal):** This enzyme is commonly used to detect senescence in cultured fibroblasts, indicating their aging or damage.

4. Surface Markers and Adhesion Molecules:

- **CD44:** A cell surface glycoprotein involved in cell-cell and cell-matrix interactions. It plays a role in cell migration and wound healing, making it an important marker for fibroblasts in tissue regeneration.
- **Integrins (e.g., Integrin β 1):** These transmembrane receptors mediate adhesion between fibroblasts and the extracellular matrix. They are critical for fibroblast motility and tissue repair.

5. Stress and Inflammation Markers:

- **α -Smooth Muscle Actin (α -SMA):** Fibroblasts can differentiate into myofibroblasts, especially under conditions of tissue repair or fibrosis. α -SMA is a marker of myofibroblast differentiation.
- **TGF- β (Transforming Growth Factor-beta):** A cytokine that induces fibroblast differentiation into myofibroblasts and regulates ECM production. Increased TGF- β signaling is often associated with wound healing and fibrosis.
- **IL-6 and IL-8 (Interleukins):** These pro-inflammatory cytokines are commonly secreted by fibroblasts during inflammation or stress responses, making them key biomarkers for studying cellular responses to damage or aging.

6. Stemness and Plasticity Markers (for Multipotent Fibroblasts): ...

7. Metabolic Markers: ...

8. Applications of Biomarker Identification: ...

These biomarkers help characterize fibroblasts in various physiological and pathological states, enabling more precise study of their roles in human health and disease.

P2: What are the most important gene biomarkers for: Human foreskin fibroblasts?

o1-preview

When identifying gene biomarkers for human foreskin fibroblasts (HFFs), researchers focus on genes that regulate key cellular functions such as fibroblast identity, extracellular matrix production, wound healing, proliferation, senescence, and responses to environmental stimuli. These genes are critical in understanding how HFFs behave in both normal physiological processes and disease conditions like aging, fibrosis, and cancer. Here are some of the most important gene biomarkers for human foreskin fibroblasts:

1. Fibroblast Identity and Function Genes:

- **VIM (Vimentin):** This gene encodes vimentin, a structural protein critical for the cytoskeletal integrity of fibroblasts. It is one of the most widely used gene markers for identifying fibroblasts.
- **S100A4 (FSP1, Fibroblast-Specific Protein 1):** This gene is important for fibroblast motility and is used to identify fibroblast cells. It's associated with fibroblast activation and tissue remodeling.
- **COL1A1 / COL1A2 (Collagen Type I Alpha 1 and Alpha 2 Chains):** These genes encode type I collagen, a primary component of the extracellular matrix produced by fibroblasts. Collagen gene expression is crucial for studying fibroblast function in tissue formation and wound healing.
- **COL3A1 (Collagen Type III Alpha 1 Chain):** Another collagen gene expressed in fibroblasts, particularly during wound healing and tissue regeneration.
- **FN1 (Fibronectin 1):** This gene encodes fibronectin, an extracellular matrix glycoprotein involved in cell adhesion and migration, and is a key marker for fibroblast function in tissue repair.

2. Extracellular Matrix Production and Remodeling:

- **MMP2 (Matrix Metalloproteinase 2):** MMP2 plays a role in the degradation of the extracellular matrix, especially collagen, which is crucial for tissue remodeling, wound healing, and fibrosis.
- **MMP9 (Matrix Metalloproteinase 9):** Like MMP2, MMP9 is involved in extracellular matrix remodeling and wound healing, making it another important gene for fibroblast activity.
- **TIMP1 / TIMP2 (Tissue Inhibitors of Metalloproteinases 1 and 2):** These genes encode proteins that inhibit matrix metalloproteinases, balancing ECM degradation and synthesis, which is essential for normal fibroblast function and tissue homeostasis.

3. Cell Proliferation and Senescence:

- **CDKN1A (p21):** This gene is a critical marker for cellular senescence, regulating cell cycle arrest. It's upregulated in senescent fibroblasts and plays a role in aging studies.
- **CDKN2A (p16):** Another gene involved in cell cycle regulation, p16 is commonly used to identify senescent fibroblasts. Its expression increases as cells age or undergo stress.
- **TP53 (p53):** Known as the "guardian of the genome," TP53 plays a role in DNA repair and cell cycle arrest, and its expression is often linked to senescence and stress responses in fibroblasts.
- **PCNA (Proliferating Cell Nuclear Antigen):** This gene is used to assess the proliferative status of fibroblasts. PCNA is upregulated in actively dividing cells, making it a good marker for studying cell proliferation in fibroblasts.

4. Cytokines and Growth Factors (Involvement in Wound Healing and Inflammation): ...

5. Differentiation and Myofibroblast Activation: ...

Summary of Key Gene Biomarkers:

- Fibroblast Identity: **VIM, S100A4, COL1A1, FN1**
- Extracellular Matrix and Remodeling: **COL3A1, MMP2, MMP9, TIMP1**
- Proliferation and Senescence: **CDKN1A (p21), CDKN2A (p16), TP53**
- Inflammation and Growth Factors: **TGFB1, IL6, IL8, PDGF**
- Myofibroblast Differentiation: **ACTA2 (α -SMA), SMAD3**
- Stress Response: **BCL2, HMOX1, SOD2**

These gene markers help in characterizing the state and function of human foreskin fibroblasts in various research contexts, including wound healing, aging, fibrosis, and cellular reprogramming.

P3: We are studying a Human foreskin fibroblast cell line from a previously published paper. Here is the abstract of the paper: [insert abstract from [49]]. What are the most important biomarker genes for this cell line?

o1-preview

The abstract you provided describes a study focused on transcription factor (TF)-mediated reprogramming of human fibroblasts, particularly looking at the genome architecture and how it reorganizes during the process of transitioning these cells into a myogenic (muscle-related) identity. The key finding is that genome architecture reorganization precedes transcriptional activation of the myogenic program. This suggests that the study investigates how changes in the 3D structure of chromatin influence gene expression during cellular reprogramming.

Based on the study's focus, the most important biomarker genes for this human foreskin fibroblast cell line would be associated with:

- **Fibroblast Identity:** Baseline markers to confirm the fibroblast nature of the starting cells.
- **Myogenic Reprogramming:** Markers that identify the activation of myogenic transcriptional programs.
- **Genome Architecture and Transcription Factors:** Genes involved in chromatin remodeling and transcriptional regulation, specifically those that play a role in reprogramming.

Here are the key categories and corresponding genes likely to be important as biomarkers for this cell line:

1. Fibroblast Identity Biomarkers

These genes are important to confirm the starting identity of the human foreskin fibroblasts before reprogramming begins:

- **VIM (Vimentin):** A marker for fibroblast identity. It is expressed in mesenchymal cells and is a fundamental marker to confirm the fibroblast origin.
- **S100A4 (Fibroblast-Specific Protein 1, FSP1):** Specific to fibroblasts and used to ensure that the starting cell line retains fibroblast characteristics.
- **COL1A1 / COL1A2 (Collagen Type I Alpha 1 and 2):** These genes encode the major components of the extracellular matrix in fibroblasts, essential for confirming fibroblast functionality.
- **FN1 (Fibronectin 1):** This extracellular matrix protein gene is a hallmark of fibroblast function, particularly in wound healing and tissue remodeling.

2. Myogenic Program Biomarkers

The study focuses on myogenic reprogramming, so genes involved in myogenic lineage development are critical:

- **MYOD1 (Myogenic Differentiation 1):** This master regulator is a key transcription factor involved in initiating myogenic differentiation. It plays a critical role in reprogramming fibroblasts into muscle cells.
- **MYOG (Myogenin):** Another essential transcription factor, MYOG, is involved in the terminal differentiation of myoblasts into muscle cells and is crucial for the commitment to a myogenic fate.
- **MYF5 (Myogenic Factor 5):** A gene involved in the early stages of myogenic lineage commitment, working alongside MYOD1 to initiate muscle-specific gene expression.
- ...

3. Transcriptional Regulators and Genome Architecture Biomarkers ...

4. Chromatin Dynamics and Genome Topology Markers ...

5. Rhythmic and Biological Rhythms-Related Genes ...

Summary of Key Biomarker Genes for This Cell Line

- **Fibroblast Identity:** VIM, S100A4, COL1A1, FN1
- **Myogenic Reprogramming:** MYOD1, MYOG, MYF5, PAX7, DES
- **Chromatin and Transcriptional Regulation:** CTCF, SMARCA4, EP300, EZH2, GATA4/6
- **Genome Architecture and Chromatin Dynamics:** RAD21, SCC1, TP53
- **Rhythmic Genes:** CLOCK, ARNTL

These genes collectively serve as crucial biomarkers for tracking fibroblast identity, myogenic reprogramming, chromatin remodeling, and the interplay between genome architecture and transcriptional programs in this cell line.

5.5 Advanced Search

We show the full output produced by BRAD in the process of answering Q2: What role does PCNA play in the cell cycle and imaging? To answer this, BRAD’s **PLANNER** constructed a pipeline consisting of (1) running the **SOFTWARE** module to find first order interactions of PCNA in the **STRING** and **HuRI** databases [56, 57], (2) performing gene enrichment with the **DIGITAL LIBRARY** module, and (3) using the **RAG** system in the **LAB NOTEBOOK** to find other relevant information. After completing these tasks, the **WRITER** module was used to produce a report summarizing the entire process.

The **WRITER** module, although not a focal point of the paper, is a module designed to process the chat log and output directory to produce a single, **PDF** document that describes BRAD’s analysis and workflow. The goal is to ensure BRAD’s work is transparent, well-documented, and clear, even for users unfamiliar with AI, so they can easily understand what occurred. The **WRITER** is implemented by using LLMs to generate **L^AT_EX** code, according to a user specified **L^AT_EX** template, configured similarly to whitelisted software, to explain the results of each module.

In the below example, which begins on the following page, BRAD specifies an extended user prompt, characteristic of inputs that could initiate this pipeline. Then, a summary of the full pipeline is provided. The main section of the document is a summary of the outputs, in which BRAD lists the main information retrieved from the **LAB NOTEBOOK** module as well as a table and chart generated by the **DIGITAL LIBRARY** module, as well as references to the used literature.

The report is generated by BRAD entirely in **L^AT_EX** and then automatically compiled. The formatting, writing, and style of the report is not always consistent or ideal, but we find that the reports often summarize the pipelines well. That said, BRAD’s **WRITER** modules also provides the **L^AT_EX** used to generate the report, from which a user could make edits.

The remaining pages are a report automatically generated by BRAD in response to Q2.

PCNA's Role in Cell Cycle and Imaging

BRAD

June 21, 2024

User Prompt

write a 3 step pipeline to 1. perform a lit search of PCNA with the RAG. Use a prompt such as, "What role does PCNA play in the cell cycle and cell cycle imaging?", then 2. run the hwg analysis pipeline to find first order interactions of PCNA loaded from step 2. Next, (3.) Load the output file of the hwg codes (step 2) and perform gene enrichment and identify the pathways PCNA is involved in with enrichr. Finally, write a small report summarizing each of these findings in Step 4

Summary of Process

The bioinformatics analysis pipeline was executed by the chatbot, which utilized its various modules to complete the following steps. Initially, the Retrieval Augmented Generation (RAG) method was employed to search literature and documents, providing a summary of PCNA's role in the cell cycle and cell cycle imaging. Next, the chatbot executed the HWG analysis pipeline to identify first-order interactions of PCNA, using the information gathered from the previous step. Subsequently, gene set enrichment analysis was performed on the output file using the Enrichr database, identifying the pathways PCNA is involved in. Finally, the chatbot synthesized the information from the previous steps to generate a concise report summarizing PCNA's role in the cell cycle and cell cycle imaging, its first-order interactions, and the pathways it is involved in.

Summary of Outputs

PCNA and Cell Cycle Phases PCNA (Proliferating Cell Nuclear Antigen) plays a crucial role in identifying cell cycle phases. The formation of PCNA foci is tightly correlated with the S phase onset, with an offset of approximately 5 minutes. PCNA loading occurs before PIP-mVenus degradation, and measuring PCNA variance can accurately identify cell cycle phases.

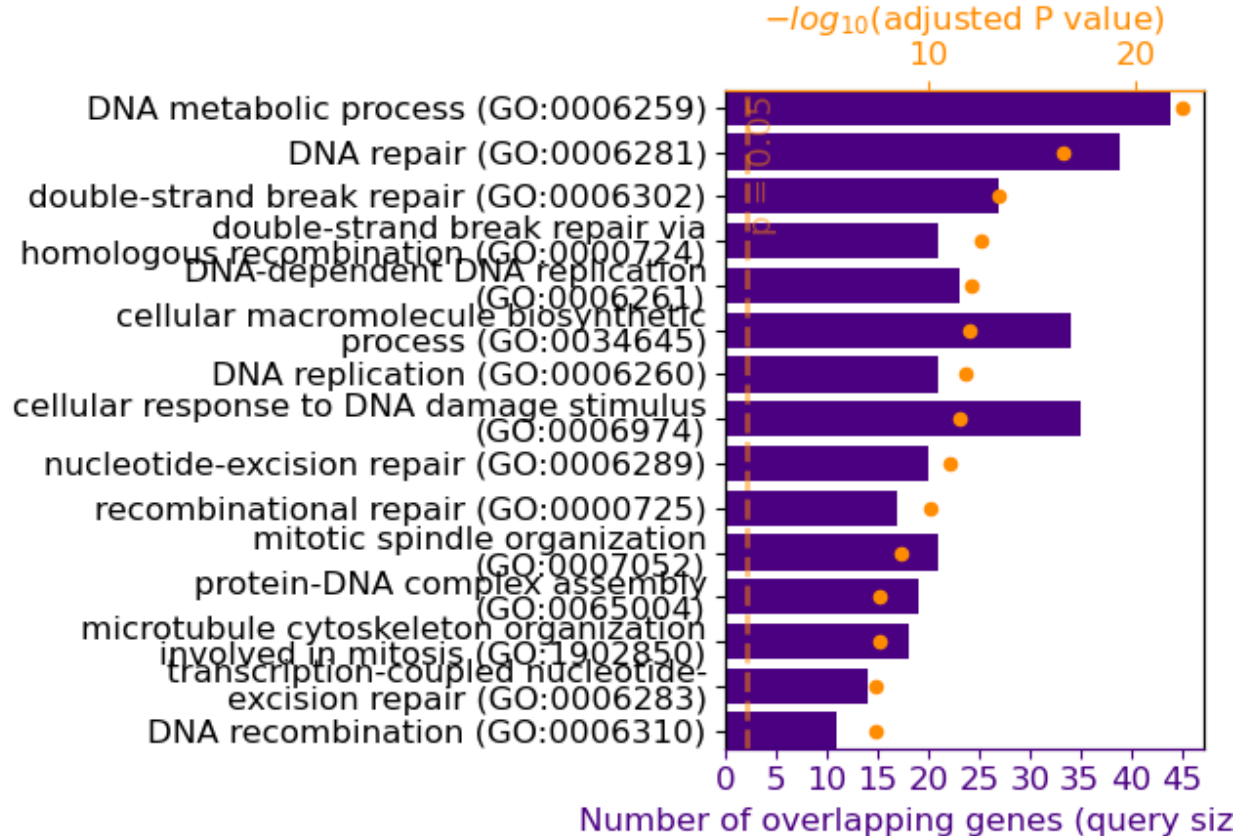
PCNA and S Phase The onset of PCNA foci formation is a reliable indicator of the S phase onset. Additionally, PCNA forms puncta and is involved in the S phase of the cell cycle. The accumulation of Citrine-Gem 1-110, a reporter protein, coincides with the onset of S phase, and PCNA is present during this phase. PCNA is also loaded onto DNA in S phase, and its interaction with the minimal human Cdt1 PIP degenon leads to the degradation of a reporter, which is a direct physical readout of DNA replication.

PCNA-based Cell Cycle Reporter Strategy This degradation is rapid and relies solely on the interaction with PCNA, allowing for a simple intensity-based cell cycle reporter strategy. Furthermore, the total PCNA signal variance can be used to analyze cell cycle transitions at lower magnification and light intensity images.

Let me know if you'd like me to make any adjustments!The execution of the High-Throughput Genome (HWG) analysis pipeline and the loading of the Representative Aspect Gene (RAG) output from Step 1 remain unclear. The provided context lacks explicit instructions on how to accomplish these tasks.

The context does provide some information on the utility of Proliferating Cell Nuclear Antigen (PCNA) as a reporter to mark subphases of S phase and mitosis, and how PCNA localization can offer additional insights

Enrichr results from database GO_Biological



into progression within S phase. However, it falls short in providing detailed instructions or guidelines on how to execute the HWG analysis pipeline or load the RAG output.

Let me know if you'd like me to make any adjustments!

Code Called

The Enrichr database was searched using the HWG output file from Step 2 to perform gene set enrichment analysis, specifically querying the gene list against the GO_Biological_Process_2021 dataset. This query returned a table with 1545 results, which are presented below.

References

- Accurate delineation of cell cycle phase transitions in living cells with PIP Fucci.pdf
- Mahdessian et al. - 2021 - Spatiotemporal dissection of the cell cycle with s.pdf
- Mapping transcriptomic vector fields of single cells.pdf
- Qiu et al. - 2022 - Mapping transcriptomic vector fields of single cel.pdf
- Accurate delineation of cell cycle phase transitions in living cells with PIP Fucci.pdf
- Mahdessian et al. - 2021 - Spatiotemporal dissection of the cell cycle with s.pdf

rank	path_name	p_val	z_score	combined_score	adj_p_val
1	DNA metabolic process (GO:0006259)...	2.92967e-26	9.9047	582.32	5.69528e-23
2	DNA repair (GO:0006281)...	2.91379e-20	7.7818	350.043	2.83221e-17
3	double-strand break repair (GO:0006...	6.18984e-17	9.9306	370.62	4.01102e-14
4	double-strand break repair via homo...	5.86456e-16	13.7531	482.353	2.85018e-13
5	DNA-dependent DNA replication (GO:0...	2.1495e-15	10.8385	366.054	8.35727e-13
6	cellular macromolecule biosynthetic...	2.88168e-15	6.18581	207.103	9.33665e-13
7	DNA replication (GO:0006260)...	5.8677e-15	12.0074	393.474	1.62954e-12
8	cellular response to DNA damage sti...	1.2607e-14	5.6649	181.303	3.06351e-12
9	nucleotide-excision repair (GO:0006...	3.97388e-14	11.6761	360.283	8.58358e-12
10	recombinational repair (GO:0000725)...	4.03308e-13	13.5193	385.827	7.8403e-11
11	mitotic spindle organization (GO:00...	1.18486e-11	7.6619	192.764	2.09398e-09
12	protein-DNA complex assembly (GO:00...	1.28645e-10	7.56912	172.379	2.08405e-08
13	microtubule cytoskeleton organizati...	1.53038e-10	8.06872	182.356	2.2885e-08
14	transcription-coupled nucleotide-ex...	2.51496e-10	11.6134	256.699	3.29441e-08
15	DNA recombination (GO:0006310)...	2.54198e-10	19.1142	422.287	3.29441e-08
16	regulation of cyclin-dependent prot...	1.26331e-09	10.0717	206.365	1.53492e-07
17	DNA integrity checkpoint signaling ...	4.71091e-09	16.7348	320.863	5.38706e-07
18	mitotic recombination (GO:0006312)...	8.49057e-09	42.1961	784.186	9.16982e-07
19	mitotic cell cycle phase transition...	1.43372e-08	5.22315	94.3323	1.46692e-06
20	telomere maintenance (GO:0000723)...	1.64778e-08	11.8829	212.957	1.60165e-06