

Bioinformatics Retrieval Augmentation Data (BRAD) Digital Assistant

Joshua Pickard^{1*}, Marc Andrew Choi², Natalie Oliven¹, Cooper Stansbury¹,
Jillian Cwycyshyn³, Nicholas Galioto¹, 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

⁵Defense Advanced Research Projects Agency (DARPA), USA

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

September 5, 2024

Abstract

We present a prototype for a Bioinformatics Retrieval Augmentation Data (BRAD) digital assistant. BRAD integrates a suite of tools to handle a wide range of bioinformatics tasks, from code execution to online search. We demonstrate BRAD’s capabilities through (1) improved question-and-answering with retrieval augmented generation (RAG), (2) BRAD’s ability to run and write complex software pipelines, and (3) BRAD’s ability to organize and distribute tasks across individual and teams of agents. We use BRAD for automation of bioinformatics workflows, performing tasks ranging from gene enrichment and searching the archive to automatic code generation and running biomarker identification pipelines. BRAD is a step toward the ultimate goal to develop a digital twin of laboratories driven by self-contained loops for hypothesis generation and testing of digital biology experiments.

Introduction

Large language models (LLMs) and scientific foundation models (FMs), have expanded the potential applications of artificial intelligence [35, 12, 7]. Originally designed for machine translation, LLMs have now found applications in areas such as question-and-answering, code generation, and data analysis. These models are driving innovation across disciplines, advancing research in aerospace [31], chemical engineering [6], and bioinformatics [39, 33, 34, 11, 21]. Allowing models to function as independent agents capable of analyzing data and making decisions, or as collaborative teams where agents share instructions and results, offers a promising avenue for enhancing human-in-the-loop supervision in AI-driven discovery [36, 17]. By enhancing scientific understanding and accelerating discovery, these technologies offer unprecedented opportunities for research.

Although a single LLM can accomplish a myriad of tasks, Multi-Agent LLM systems are increasing in popularity [17]. By delegating tasks to an array of individual LLM-powered agents, these agents can collaborate to solve various tasks, such as software development [13] and biomedical discovery [16]. In bioinformatics, Multi-Agents can power autonomous labs equipped with data analysis, enhancing scientific knowledge and accelerate discovery [23].

Our ultimate goal is the design of a self-driving, digital biology laboratory. Self-driving labs integrate both experimental and analytical processes for a lab and vastly accelerate scientific progress. Although they are still in their infancy [29], closed loop systems have already been implemented in other domains such as chemistry [5] and synthetic biology [26]. Bioinformatics Retrieval Augmentation Data (BRAD) digital

assistant integrates recent LLM and FM technologies with traditional bioinformatics tools, creating an advanced LLM-based agent, capable to be chained together as a Multi-Agent System that can conduct complex decision-making and execute digital biology experiments.

The prototype BRAD research assistant utilizes a LLM to operate an array of modules. The **Lab Notebook** Module integrates documentation, protocols, and literature using Retrieval Augmented Generation (RAG) [20], while the **Digital Library** Module enables searches across online data and literature repositories. Additionally, the **Software** Module automates code generation and execution. Together, these main features—supported by a robust software framework—allow BRAD to provide in depth responses to research grade user queries by performing its own independent research.

We demonstrate BRAD’s ability to execute standard research workflows efficiently. Individually, a BRAD bot can handle tasks like literature searches, gene set enrichment from databases such as Enrichr or Gene Ontology, and generation and execution of Python and MATLAB software. When organized effectively, BRAD bots can work in tandem to solve more complex challenges such as biomarker identification, network analysis, or exploratory data analysis. This collaboration between BRADs allows each agent to tackle tasks of greater complexity, producing results that surpass the sum of their individual efforts (see fig. 1). The remainder of this article first presents the anatomy (or software architecture) of BRAD and then presents several results of BRAD’s ability to dynamically learn new topics and use cases of this code, such as biomarker identification and enrichment.

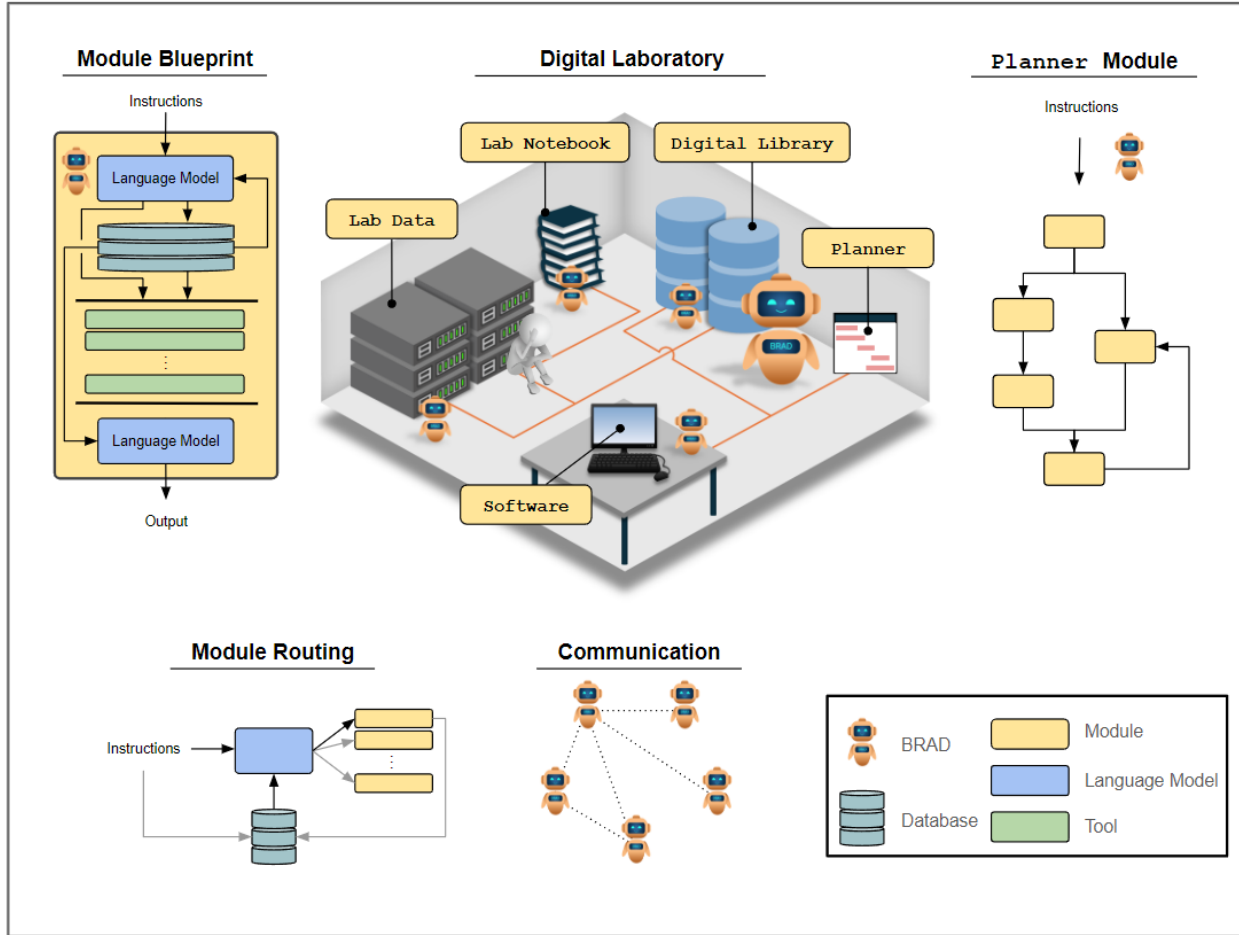


Fig 1. (Module Blueprint) Architecture of BRAD. To process a single instruction or user query, a series of language models and databases are used. First, language models are employed to appropriately select and use the tool, software, or database. Then, an LLM responds to the instruction with the additional information. **(Digital Laboratory)** Overview of the digital lab including multiple BRADs working as a Multi-Agent System to coalesce different lab modules, software, and data. **(Planner Module)** Coordination of modules and agents. Multiple levels of planning and coordination allow individual bots to schedule tasks and for multiple bots to divide and distribute tasks. **(Module Selection)** Semantic routing is used to determine which module will best process a user input. The routing language model maintains a databases that is used to select modules, and the databases can be updated over time based on user feedback. **(Communication)** BRADs are able to communicate by sending both instruction sets and responses to one another. When BRADs communicate with one another, the input from one BRAD to another is treated as a user input, similar to if a human sent the message. **(Key)** The Module block describes the different main features of BRAD, including the Digital Library, Lab Notebook, Software, and so forth. The Language Model block shows where LLMs and the routing model are used to make decisions. The Tool and Database blocks describe generic features, such as interfacing with software, Enrichr, or other capabilities that are specific to each module.

Anatomy of BRAD

At the heart of this architecture is an LLM(s) that orchestrates the usage of BRAD’s different modules. These modules enable BRAD to interface directly with documentation, published research, the internet, software, and user data. This modular architecture grants BRAD the flexibility to integrate diverse tools while offering users a framework to customize and expand BRAD’s capabilities.

When responding to a user query, BRAD’s modules adheres to a standard template (see fig. 1, left): (1) a semantic router [25] selects the most appropriate module to handle the query; (2) within the selected module, a series of LLM prompts are used to retrieve literature, search online, execute code, or other module specific tasks; (3) an LLM summarizes BRAD’s findings and continues the conversation.

For queries demanding multiple modules, BRAD’s **PLANNER** acts similar to a ReAct agent [40] by breaking down the query into a series of tasks to be handled by individual modules. These tasks are executed sequentially, with BRAD compiling the results from each module to form a cohesive response. The **PLANNER** can create linear workflows or more advanced pipelines that include feedback loops, branching, and other control flow mechanisms.

The backbone of BRAD’s design is code that supports its integration with other software and maintain its ease of use. Each BRAD chatbot or chat session outputs all figures, downloaded data, and other relevant information to a centralized output directory. This directory also maintains a comprehensive log that tracks BRAD’s interactions with the user, databases, and other tools. Built on the LangChain framework, BRAD can utilize LLMs and APIs either locally or from providers like NVIDIA and OpenAI. It can also be integrated into any LangChain or LangGraph architecture, ensuring broad compatibility with standard frameworks [2]. In the remainder of this section, we outline the capabilities of BRAD’s main modules, with further implementation details available in §SI.1.

Lab Notebook Module

The **Lab Notebook** Module connects BRAD to internally stored documents, including literature, protocols, textbooks, and any other pdf files. It employs RAG to tailor the LLM’s responses based on curated user-provided information. BRAD can choose between a previously constructed document database or create one dynamically. For further detail on database creation, see §SI 1.2.

RAG functions in two phases: retrieval and generation [20]. During retrieval, the aforementioned database is searched to find chunks of text relevant to the user’s query. In the generation phase, the retrieved documents are combined with the user query to create a prompt for the LLM. This approach enhances response quality and reduces hallucinations [32].

The **Lab Notebook** module allows users to control database construction on their own literature and customize the retrieval and generation processes (see fig. 2). BRAD supports various techniques, including similarity-based search, maximum marginal relevance (MMR) [8], and multi-query retrieval during the retrieval phase, as well as compression [1] and reranking [31] in the generation phase.

Example Use Case. As an example of the output, see 2.2 (Q1): How can we view Pore-C as a hypergraph? This question requires the synthesis of information from different fields. For instance, Pore-C is a recently developed experimental genomics technique that characterizes three-dimensional chromatin structure by identifying multi-way contacts in the chromatin polymer. On the other hand, hypergraphs are a higher-order generalization of graphs that allows edges to contain multiple nodes. Since Pore-C data captures multi-way contacts, hypergraphs are a natural way to view Pore-C data.

Using OpenAI’s GPT-4.0 mini as the underlying LLM for response generation, we equipped BRAD with three different configurations to test BRAD’s question and answering 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 chunks, generated by recursively chunking the papers our database¹. In our view, the *enhanced* RAG, where BRAD makes full use of its **Lab Notebook**, best answers this question.

¹See <https://rajapakse.lab.medicine.umich.edu/papers/digital-library>

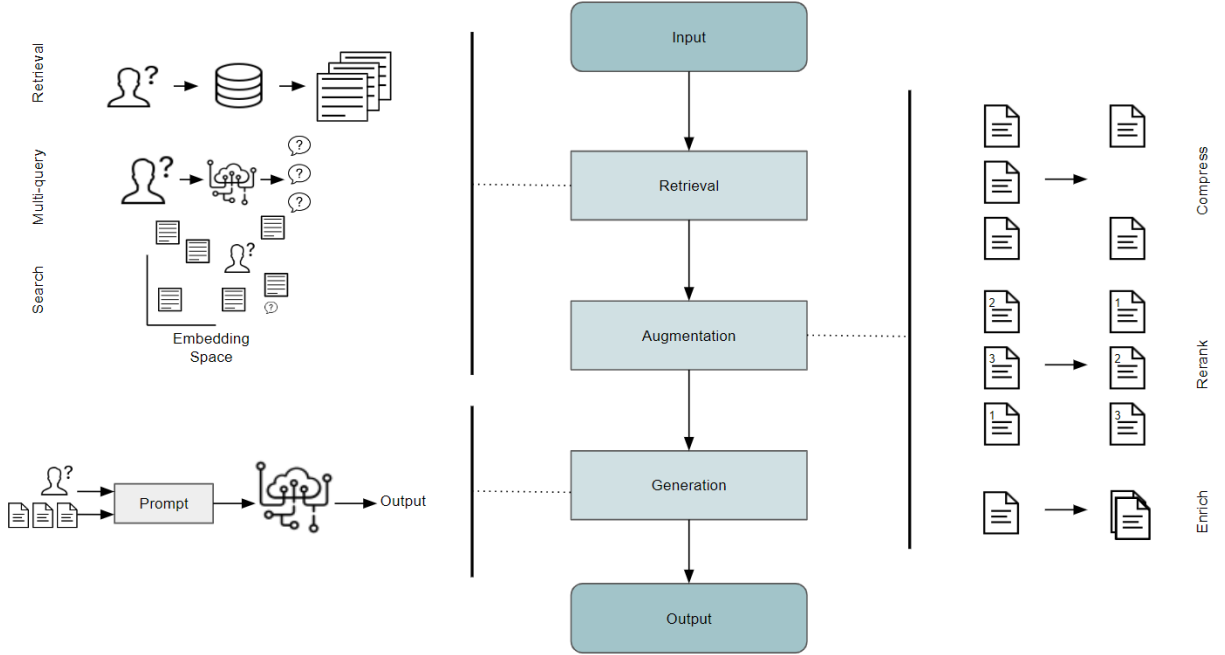


Fig 2. Overview of the Lab Notebook RAG pipeline. **(Retrieval)** Different methods of retrieval, such as similarity search, maximum marginal relevance search, and multi-query search are available. **(Augmentation)** Several augmentation techniques including contextual compression, reranking, document enrichment and more have been implemented. **(Generation)** The generation phase takes the relevant pieces of augmented information from the database and the user prompt to generate a response to the user query.

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. Consequently, we deem *vanilla* LLM's response to be the worst. In particular, its discussion of Higher-Order Interactions parrots its own previously generated text. Additionally, although it claims to give perspectives into how to analysis and visualize the data, it completely forgets to discuss visualization techniques.

Comparing this to *standard* RAG, *standard* RAG is also overly 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 omit the discussion altogether.

As a result, we believe the *enhanced* RAG best answers the question. It fixes many of the issues that the *standard* RAG fixes without the overly repetitive nature of the *standard* RAG. Specifically, its discussion on Higher-Order Interactions and Analysis and Algorithms adds new insights compared to both *vanilla* LLM and the *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:

⋮

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.

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.

⋮

In summary, viewing Pore-C data as a hypergraph allows for a richer representation of the complex interactions within the genome, facilitating deeper analysis and understanding of chromatin architecture and its functional implications.

Vanilla LLM

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:

⋮

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.

See §SI2.1.1 for the full *vanilla* LLM output.

Quantitative Assessment. Beyond the qualitative assesment of the *Lab Notebook* module, we employed the RAG Assessment (RAGAs) framework to quantify the advantage of BRAD’s RAG system [15]. We generated a set of research questions and evaluated BRAD’s ability to answer them using various features of the *Lab Notebook*. These questions were manually classified into three different topics: Mathematics, Biology, and Computer Science and Data. This framework employs an evolution-based question generation approach to produce a diverse range of questions [38]. This process begins by invoking an LLM to generate a basic set of questions from the document database. Then, it takes those same questions as an input and asks the LLM to “evolve” the questions by making them more complex or simple. This prompt engineering is preformed iteratively until the final set of questions are generated. BRAD then answers the generated questions, and its performance is scored based on the expected answer for each question. A detailed explanation of how each metric is calculated is included in SI §2.1.2.

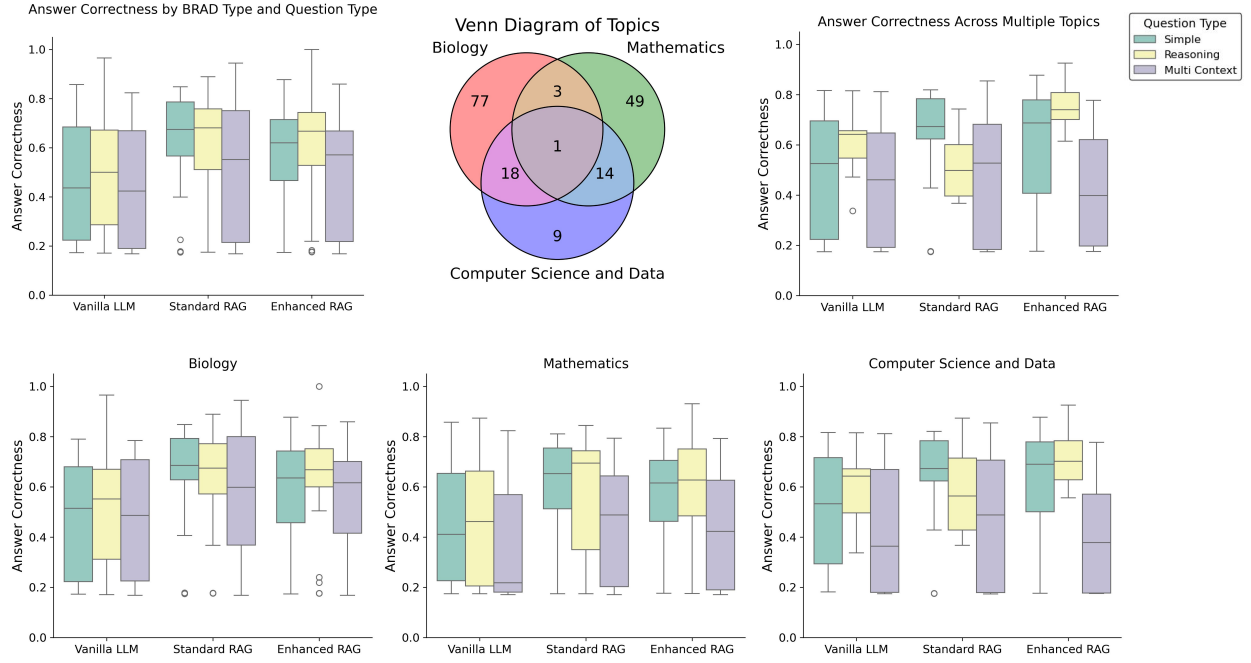


Fig 3. (Top Left) Each of the BRAD chatbots answering every question. **(Top Middle)** Distribution of different question topics. **(Top Right)** Each of the BRAD chatbots only answering questions that are of multiple topics. **(Bottom)** Each of the BRAD chatbots answering all the questions, split by topic.

Using the same document database, we generated three types of questions: (1) *simple questions* where answers are found directly in the database, (2) *reasoning questions* that require deeper understanding of a single piece of text, and (3) *multicontext questions* that require understanding multiple documents to answer successfully. We asked each of the same three BRAD chatbots as before the generated set of questions and compared their responses using RAGAs’s evaluation metrics.

Compared to its counterparts, the *standard* RAG overall has the greatest mean answer correctness, indicating it is the best performer. However, the *enhanced* RAG has a higher mean answer correctness best performance on questions related to computer science and data. We believe that during contextual compression, these mathematical symbols’ meaning is lost in its summary, leading to *enhanced* RAG having a lower mean answer correctness than *standard* RAG.

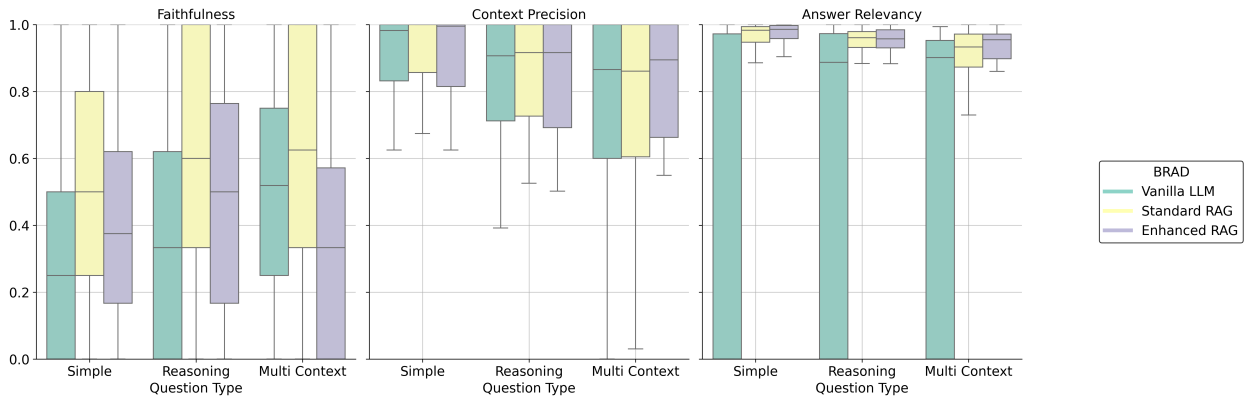


Fig 4. Comparison of faithfulness, context precision across three different BRAD chatbots answering three different question types: simple, reasoning, and multi context.

The faithfulness, a proxy for reliability, of all three models is relatively low. This demonstrates that the

BRAD chatbots do not regurgitate what was presented in the papers. Interestingly, RAG consistently has substantially higher faithfulness. Although this may be perceived as a positive, there can be adverse effects as shown in Q1, where the *standard* RAG parrots text found in the retrieved documents but does not provide synthesis or further analysis.

Context precision is fairly high for all three models, indicating that chunks relevant to the ground truth are being retrieved for the RAG systems and that the *vanilla* LLM, which does have the lowest context precision, provides on-topic answers. We see significantly less variability in the *enhanced* RAG’s context precision with multi-context questions, validating reranking’s effectiveness. This is further validated with higher answer correctness scores for questions that span multiple topics as BRAD needs to identify the most important chunks from at least two distinct subtopics. Finally, both the *standard* and *enhanced* RAG significantly outperform the *vanilla* LLM with respect to answer relevancy, demonstrating considerable mitigation of hallucinations. Note that context recall, which is not shown, was consistently above 95% for all three versions.

Software Module

BRAD’s **Software** Module features a multistage pipeline that utilizes LLMs to write and execute code in response to user queries. It supports running Python, MATLAB, and Bash scripts. Although BRAD is primarily designed for workflows that involve saving data to files at each stage, as is common in bioinformatics, this module also supports LLM generated code that avoids writing files between stages.

BRAD performs a series of steps to safely execute a piece of software (see fig. 5 and SI.fig. 7). The process starts with BRAD reviewing the documentation of all available scripts and leveraging the LLM to select the code that best addresses the task. Based on user input and selected code, BRAD uses the software documentation, few shot learning examples, and the LLM to generate the appropriate code. Refer to §SI.1.3 for details on the implementation.

To ensure safe execution of LLM generated code, BRAD is restricted to running whitelisted software. This allows BRAD to verify that the correct paths, inputs, outputs, and other arguments are provided for each script. Additional checks ensure the closure of all delimiters and other issues identified while testing LLM-generated code.

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 may continue until the code executes without error. Once all issues are resolved, the code is run.

We designed scripts to execute LLM generated code without running whitelisted scripts. While this approach allows BRAD to fully utilize LLM generated code, this process introduces several challenges: (1) ensuring the safety of LLM-generated code, and (2) generating bug-free, free-form code with LLMs. To demonstrate this feature, we designed a script that enables the safe execution of the full **Scrapy** library with BRAD. However, ensuring LLM generated code is error free remains a challenge.

Digital Library Module

The **Digital Library** module provides BRAD access to an array of literature, archives, and online databases (see fig. 5). Building on a standard RAG, BRAD can retrieve and share information from these sources, either directly with the user or through other modules. Below, we outline the integration of several databases, and we’d like to note the relative ease of integrating new databases.

Enrichr. Enrichr is a comprehensive gene set enrichment tool commonly used in many bioinformatics pipelines [9]. We integrated Enrichr into BRAD to leverage gene set annotations, curated pathways, and ontology knowledge to enhance BRAD’s responses to user queries.

Gene Ontology. The Gene Ontology (GO) database is an initiative to provide a uniform approach and vocabulary to studying genes and gene products across different species [4, 3]. When gene names, GO terms, or other keywords are identified, BRAD has the ability to define these GO terms and retrieve charts that diagram their relationships with other genes. Moreover, the GO terms are associated with papers that may be integrated via other databases and modules of BRAD.

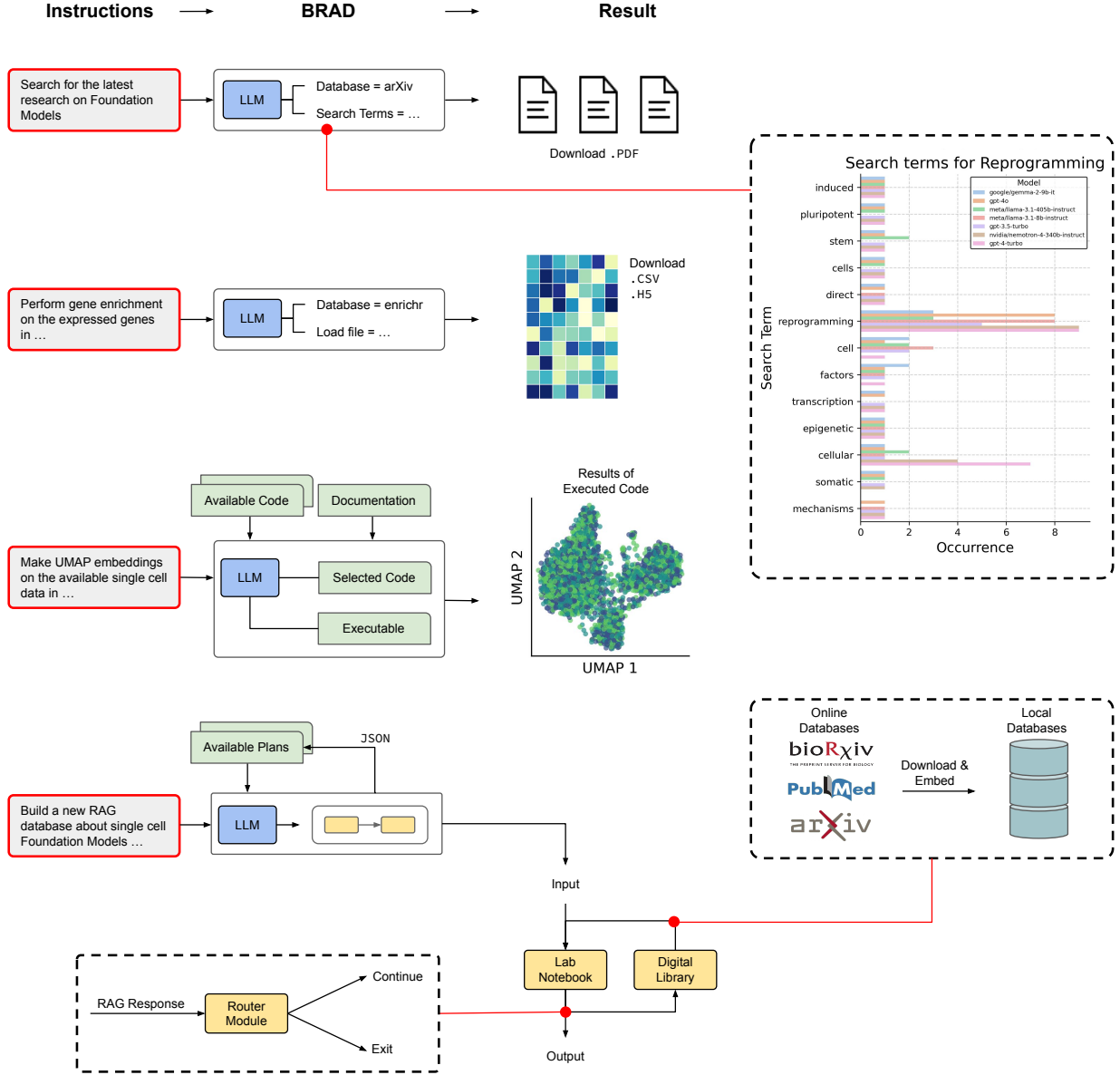


Fig 5. Four example instructions are shown. **(Digital Library: Literature)** When a user's request requires searching new literature, BRAD's LLM will determine the appropriate database and search terms that can be used to query and download new literature. On the right, six LLMs are profiled to show common search terms. **(Digital Library: Databases)** A similar process occurs to search bioinformatics databases to download raw data, gene enrichment, or other data based on a user's query. **(Software Module)** The software module uses both the user's instructions, available software, and software documentation to respond to a user's query. The output of this process is both (1) the results of the executed code and (2) the new LLM generated code that can be revised and edited by the human. **(Planner Module)** The planner module outputs a pipeline of other modules that responds to complex queries. These are selected from previously built pipelines and can be constructed for new requests. In this pipeline, a loop between the Lab Notebook (local database) and Digital Library (online database) are used to build a new RAG database. The Router Module manages the control flow of this pipeline to ensure the loop is run a sufficient and appropriate number of times.

Literature Repositories. BRAD integrates the arXiv, bioRxiv, and PubMed literature 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.

Integration with Other Modules. Online retrieval works symbiotically with the other modules. For instance, the Lab Notebook RAG and the Literature database search complement each other. These modules can be used in tandem application for performing literature reviews and constructing RAG databases (see fig. 5 and §SI2.2). When the Lab Notebook does not contain literature relevant to a user query, new literature can be obtained to improve the RAG database. Conversely, when new literature is obtained, the RAG is used to summarize the documents and respond to the user. Similar iterations can occur between the software engine and the database searches.

Additional BRAD Modules

BRAD also contains modules responsible for documentation generation, planning and executing multistage queries, documentation and code generation, and more. Refer to §SI.1.1-1.6 for additional implementation details.

Digital Laboratory

This section presents various examples and results achieved by BRADs. We demonstrate the ability to (1) perform a comprehensive search about a gene, using network analysis, multiple databases, and RAG, (2) automatically build a RAG database targeting a specific question from scratch, (3) run biomarker identification pipelines, and (4) perform automatic exploratory data analysis. The first three examples are performed using single BRAD chatbot, and performing exploratory search of data is done with multiple BRAD agents that pass information between one another. Each use case demonstrates how a single BRAD or a team of BRADs tackles or automates complex research tasks under human supervision.

Enhanced Search

Integrating the **SOFTWARE** and **Digital Library** modules improves quality of the RAG responses generated by the **Lab Notebook**. Consider, for instance, Q2: What role does PCNA play in the cell cycle and imaging? BRAD orchestrates the following process: (1) identifies which genes interact with PCNA; (2) determines the pathways related to PCNA; and (3) uses the related genes and pathways to search the literature and formulate a comprehensive answer. In addition to providing a detailed text response, this process generates supplementary materials such as images, spreadsheets, and other documentation to offer further context to the user (see fig. 6 and §SI.2.3).

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

System 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.

...

- 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.

Biomarker Identification and Enrichment

Biomarker identification – finding genes that indicate a biological state – remains a challenge [18, 28]. A typical framework for sensor selection involves two steps: (1) modeling and (2) observability analysis. Equipped with modeling and observability software from [28], BRAD was provided with given several datasets [10, 22] and tasked with uncovering potential biomarkers.

After normalizing data with **Scanpy**, BRAD built several models of dynamics for each dataset using Dynamic Mode Decomposition and Data Guided Control frameworks. BRAD then passed the models into observability software to identify and rank the contribution of each gene as a sensor. Finally, BRAD queried the top-ranked genes in Enrichr to determine the relevant biological pathways and processes associated with the data.

BRAD built models of the dynamics in both the data space and based upon Geneformer embeddings [33], which were also generated by the **SOFTWARE** module. While both processes model the dynamics, BRAD correctly identified, based on the **Laboratory Notebook** and documentation found in the observability code, that the observability codes should not be applied in Geneformer embedding space. This constraint results from the inability to interpret observability results in embedding space. A challenge for scFMs will be to make these results interpretable in the original space of the data, perhaps using a decoder or alternative dynamics perspective. Despite reporting that the observability codes did not make sense to execute, BRAD still ran these codes in the embedded space.

Automated Cellular Reprogramming Analysis

A team of BRADs was employed to study *in silico* perturbations of Fibroblasts toward hematopoietic stem cells (HSCs). Several hundred reprogramming regimes were identified with Data Guided Control [30], each consisting of using a subset of ten transcription factors, and after embedding each recipe with Geneformer [33], the BRADs were tasked with performing exploratory analysis to identify the best recipe.

Coordination. A central BRAD generated high-level instructions for a team of five BRAD workers based on detailed documentation of the experiment. Each worker used the **PLANNER** module to translate these objectives into tasks for the **SOFTWARE** module. Expert review led to adjustments, including switching to cosine similarity, removing the literature review task, and adapting file paths for our lab’s system. This human-in-the-loop (HITL) approach ensures the BRAD team’s analysis aligns with our research goals and lab environment.

Execution. After revising the plans, the BRAD team executed its analyses, generating figures, code, and new data files. In minutes, the BRADs produced outputs that would typically take humans hours or even days. However, some duplication of files suggested better data structuring was needed initially. While the figures were unpolished, they enabled rapid exploratory analysis of cell clusters from different transcription factor reprogramming, ensuring reproducibility and allowing for human feedback. All outputs from the BRADs require thorough human review, shifting researchers’ roles from writing and debugging code to validating software, documenting analysis pipelines, and reviewing results. This shift enhances research efficiency and aligns with the software development life cycle, where validation and testing are essential in ensuring accuracy.

Discussion

This work demonstrates the potential of leveraging LLMs to power semi-autonomous bioinformatic assistants, such as BRAD. Our prototype BRADs were deployed on an array of tasks, ranging from standard research projects such as searching literature or gene enrichment, to more complex procedures such as executing software pipelines and performing exploratory data analysis. Through automation, we achieved improved speed, reproducibility, and documentation of our workflows, albeit on well-known, standard tasks in a prototyped environment.

The major strengths of the BRAD system lie in its ability to reliably perform common research tasks. The breakdown and distribution of problems, such as the multiple tools used to answer Q2, the automated execution of the biomarker pipeline, and the distributed workflow on Geneformer, split seemingly complex tasks into small, manageable problems that can be understood and executed by an LLM. This modular approach offers both a blueprint for scaling up the number of agents that collaborate while still allowing humans to have fine control over the different aspects of the pipeline.

There are two central challenges in our current system. First, providing instructions that allow an LLM to split and distribute a complex task in a thoughtful way is a nontrivial problem for the user. Inconsistencies between the organization and capabilities of the user’s instructions, the **PLANNER**, and the available **SOFTWARE** cause issues. Furthermore, the detection of an error or corruption in a multi-stage or multi-BRAD workflow is often unclear to the system, particularly without user intervention.

Maintaining a HITL is both necessary to effectively use BRAD and an important stopgap for preventing errors from propagating in the system. For instance, in the Geneformer example, the human was used to modify parameters or instructions to the individual agents prior to their execution. Allowing users to interact with the **PLANNER** ensures that the initial human instructions are appropriately interpreted prior to distributing the instructions to the team of agents. To augment this, each human-approved plan is saved for potential future use in order to maximize the HITL contributions.

The second challenge of this work is effectively passing information between different modules and agents. For instance, the structure, purpose, and organization of new data files created by one agent in the Geneformer experiment must be communicated between agents that touch the same file. Passing information between modules of a single BRAD is managed with LLM memory, and we plan to develop new solutions for passing information between multiple agents.

In summary, this work highlights the potential use cases of BRADs working as a Multi Agent system to act as 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, successfully 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 the members of the Rajapakse Lab, Adam Lord, and Santosh Srivastava for helpful and inspiring discussions. We also thank Vivan Nyati, a student of Greenhills, Ann Arbor, for his valuable contributions over the summer. 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).

References

1. 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.
2. Ben Affarh. *Generative AI with LangChain: Build Large Language Model (LLM) Apps with Python, ChatGPT, and Other LLMs*. Packt Publishing Ltd., Birmingham, UK, 2023. ISBN 978-1-83508-346-8.
3. 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.
4. 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.
5. Jeffrey A Bennett and Milad Abolhasani. Autonomous chemical science and engineering enabled by self-driving laboratories. *Current Opinion in Chemical Engineering*, 36:100831, 2022. ISSN 2211-3398. doi: <https://doi.org/10.1016/j.coche.2022.100831>. URL <https://www.sciencedirect.com/science/article/pii/S2211339822000417>.
6. Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.
7. Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.
8. Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries, 1998. URL https://www.cs.cmu.edu/~jgc/publication/The_Use_MMR_Diversity_Based_LTMIR_1998.pdf.
9. 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.
10. 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.
11. 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.
12. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
13. Zhuoyun Du, Chen Qian, Wei Liu, Zihao Xie, Yifei Wang, Yufan Dang, Weize Chen, and Cheng Yang. Multi-agent software development through cross-team collaboration, 2024. URL <https://arxiv.org/abs/2406.08979>.
14. Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*, 2023.
15. Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation. 2023. URL <https://arxiv.org/abs/2309.15217>.

16. Shanghua Gao, Ada Fang, Yepeng Huang, Valentina Giunchiglia, Ayush Noori, Jonathan Richard Schwarz, Yasha Ektefaie, Jovana Kondic, and Marinka Zitnik. Empowering biomedical discovery with ai agents, 2024. URL <https://arxiv.org/abs/2404.02831>.
17. Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges, 2024. URL <https://arxiv.org/abs/2402.01680>.
18. 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.
19. 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.
20. 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.
21. Jiajia Liu, Mengyuan Yang, Yankai Yu, Haixia Xu, Kang Li, and Xiaobo Zhou. Large language models in bioinformatics: applications and perspectives. *ArXiv*, 2024.
22. 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.
23. Zhihan Liu, Yubo Chai, and Jianfeng Li. Towards fully autonomous research powered by llms: Case study on simulations, 2024. URL <https://arxiv.org/abs/2408.15512>.
24. Katja Luck, Dae-Kyum Kim, Luke Lambourne, Kerstin Spirohn, Bridget E Begg, Wenting Bian, Ruth Brignall, Tiziana Cafarelli, Francisco J Campos-Laborie, Benoit Charlotiaux, et al. A reference map of the human binary protein interactome. *Nature*, 580(7803):402–408, 2020.
25. Dimitrios Michael Manias, Ali Chouman, and Abdallah Shami. Semantic routing for enhanced performance of llm-assisted intent-based 5g core network management and orchestration. *arXiv preprint arXiv:2404.15869*, 2024.
26. 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. ISSN 0958-1669. doi: <https://doi.org/10.1016/j.copbio.2022.102881>. URL <https://www.sciencedirect.com/science/article/pii/S0958166922002154>.
27. 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.
28. Joshua Pickard, Cooper Stansbury, Amit Surana, Anthony Bloch, and Indika Rajapakse. Biomarker selection for adaptive systems. *arXiv preprint arXiv:2405.09809*, 2024.
29. Simon D Rihm, Yong Ren Tan, Wilson Ang, Markus Hofmeister, Xinhong Deng, Michael Teguh Laksana, Hou Yee Quek, Jiaru Bai, Laura Pascazio, Sim Chun Siong, et al. The digital lab manager: Automating research support. *SLAS technology*, 29(3):100135, 2024.

30. 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.
31. Vansh Sharma and Venkat Raman. A reliable knowledge processing framework for combustion science using foundation models. *Energy and AI*, 16:100365, 2024.
32. Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*, 2021.
33. 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.
34. 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.
35. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
36. Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*, 2024.
37. 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.
38. Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. 2023. URL <https://arxiv.org/abs/2304.12244>.
39. 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.
40. Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL <https://arxiv.org/abs/2210.03629>.

Supporting Information

Contents

1	Anatomy of BRAD Extended	17
1.1	Base Architecture	17
1.2	Lab Notebook Module	18
1.3	Software Module	19
1.4	Digital Library	21
1.5	Planner Module	21
1.6	Module Selection and Control Flow	24
1.7	Prompt Templates	24
2	Results and Examples	25
2.1	Lab Notebook Evaluation	25
2.1.1	Qualitative Example	25
2.1.2	RAG Assessment (RAGAs) evaluation metrics	28
2.2	Building a Database	34
2.3	Advanced Search	37

1 Anatomy of BRAD Extended

This section provides an in-depth overview of BRAD’s software architecture, intended for readers who are either interested in implementing a similar system or seeking to understand the technical specifications of BRAD. For a more comprehensive guide, please refer to the software manual [here](#).

1.1 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.

Language Model Integration. BRAD’s integration is built atop the LangChain interface for querying LLMs. 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 chatbot. 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. After exiting a chatbot, the session state will be saved within the output directory and can be used to turn on BRAD 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 LLMresponses, reading data from files, managing the output directory, and more.

1.2 Lab Notebook Module

RAG employs a two step process: retrieval and generation. During the retrieval phase, a database is searched to identify documents and information similar to the user query. During the generation phase, the retrieved documents and user query are combined to form a single prompt for the LLM. Based upon the additional context provided in the retrieved documents, this RAG can provide improved responses by (1) reducing hallucinations, (2) improving the accuracy of responses, and (3) providing verifiable sources for each response. BRAD contains several features to augment the traditional RAG system.

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. Although there are various 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

Although semantic chunking creates more interpretable chunks, recursive chunking captures for more complex linguistic structures. By identifying sections with a high density of separators, we can filter out chunks that belong to reference sections of a paper. The distribution of these separators is bimodal, with little data in the middle range, helping to prevent the RAG from retrieving chunks filled with buzzwords but lacking substantive content.

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 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 minutes to hours, depending on its size. For instance, our database consisting of more than 500 papers and books we created a database that consist of roughly 18000 pages. Although creating the vectorized database takes a few minutes, once the database is created, BRAD runs considerably fast, despite the size of the database. Furthermore, based on the LLM, this can also heavily influence computational cost.

While the enhancements mentioned above improve the performance of the RAG, they also introduce additional computational costs.

1.3 Software Module

BRAD’s coding module provides a multistage pipeline that uses LLMs in conjunction with custom documentation to both analyze data as well as improve the response to a user’s query. BRAD contains implementations to execute python, Matlab, and bash scripts or pipelines.

Multistage Execution Pipeline. A series of steps are required for BRAD to run a piece of code. The process begins with a user’s query, where BRAD first reads the documentation of all available code to determine the appropriate script to execute. With the users input and selected code, BRAD will reread, in greater detail, the documentation of the code to execute. Using this information, along with a set of examples based on the principle of few shot learning, an LLM then writes a snippet of code to execute the appropriate script.

Once an initial block of code is generated, a series of validations are performed prior to execution. To address issues found in the testing issues of this pipeline with llamas 2 and 3, among other models, the proposed code is verified to ensure closure of all delimiters and correctly pointing to an output directory for all read or write operations, among other common syntax errors. After these validations, the code is compiled to ensure it is bug-free before execution.

Any errors identified during the validation stages are passed, along with the code and original query, back to the LLM for refinement. This iterative process of code generation and validation may be repeated several times until the code executes without error. Once all issues are resolved, the code is run.

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, adhering to standard python or Matlabconventions, enabling BRAD to understand the software’s purpose and execution process.

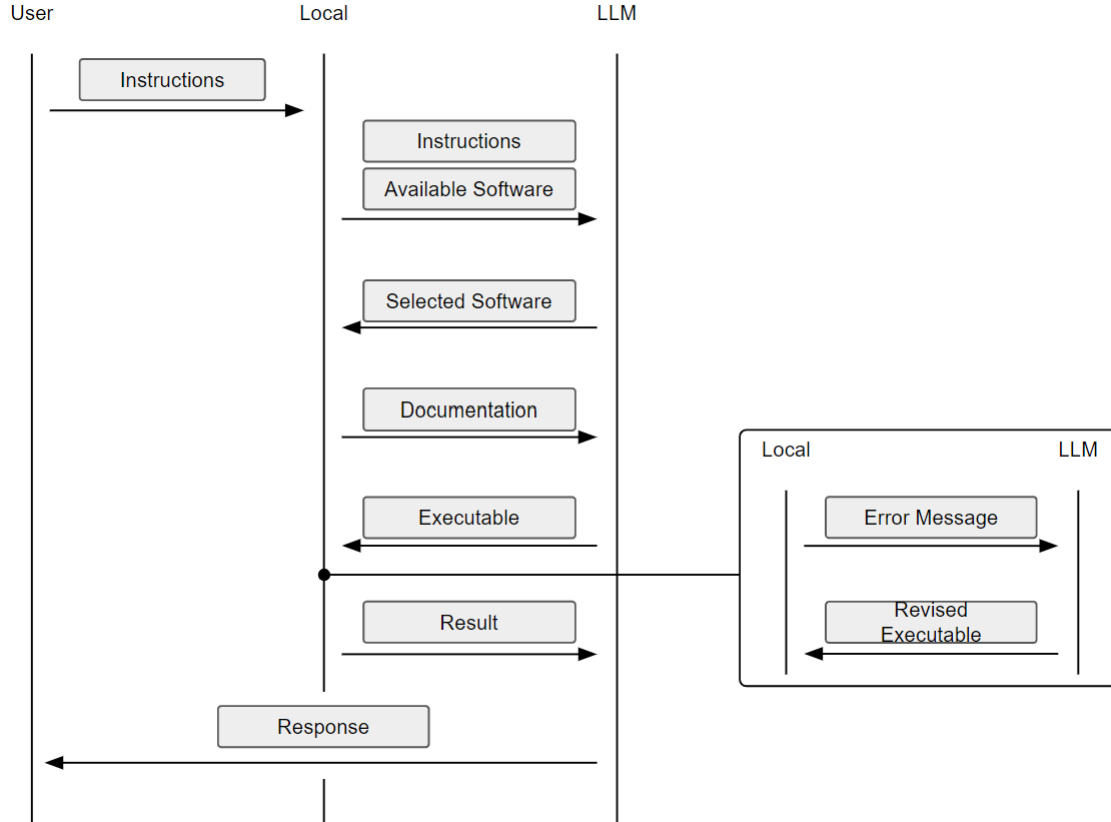


Fig 7. Sequence diagram of SOFTWARE module. (1) Instructions or a query from the user are used to identify that the SOFTWARE module should be run. (2) Summary documentation of the available code are passed along with the user instructions to a LLM. (3) A specific piece of code is selected to run. (4) 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.

- (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 curated.

1.4 Digital Library

Beyond traditional text-based retrieval, there is a growing collection of online databases common to bioinformatics workflows that can be integrated into the RAG pipeline. Here, we outline the integration of several databases and highlight that our architecture allows for the seamless addition of new databases into BRAD; see fig. 8.

Enrichr. Enrichr is a comprehensive gene set enrichment tool commonly used in many bioinformatics pipelines [9, 19, 37]. 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 [4, 3]. 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.

1.5 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. 9 for a high level schematic of a workflow involving the **Planner**.

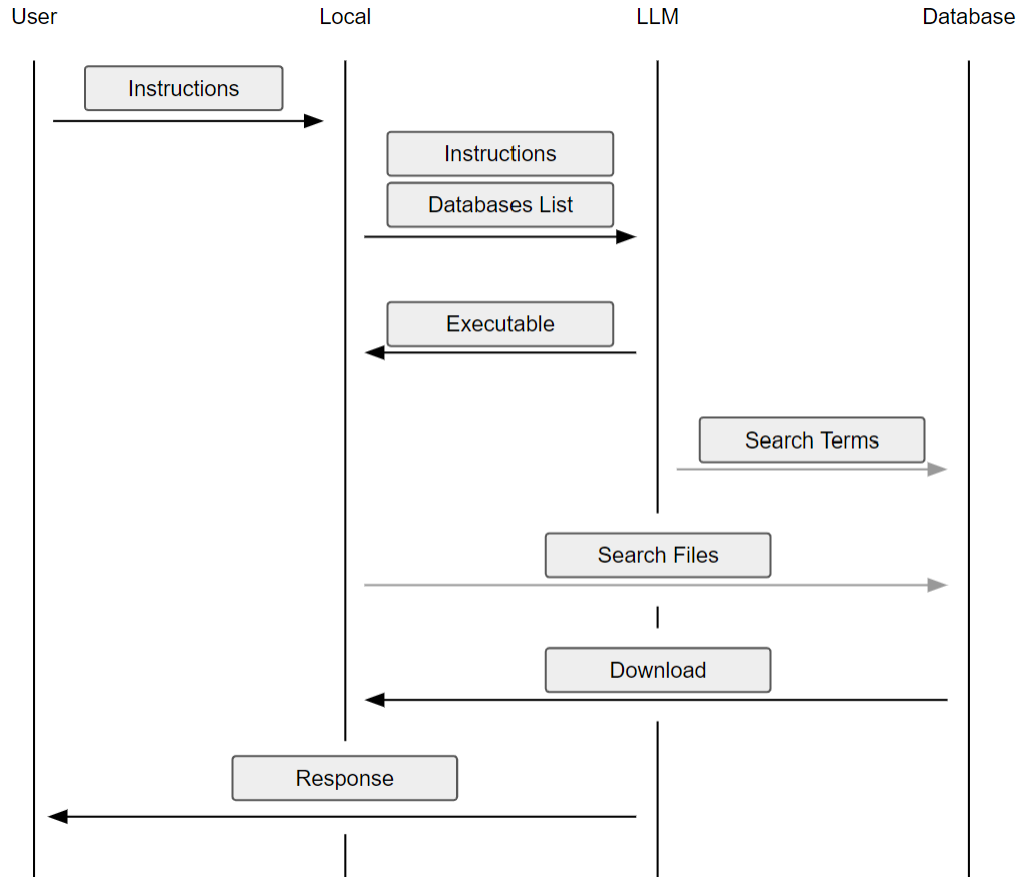


Fig 8. Sequence Diagram of querying the Digital Library. (1) Instructions or a query from the user are used to identify that the **Digital Library** module should be run. (2) The user instructions and a list of available databases are sent to the LLM. (3) The LLM sends executable instructions outlining 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.

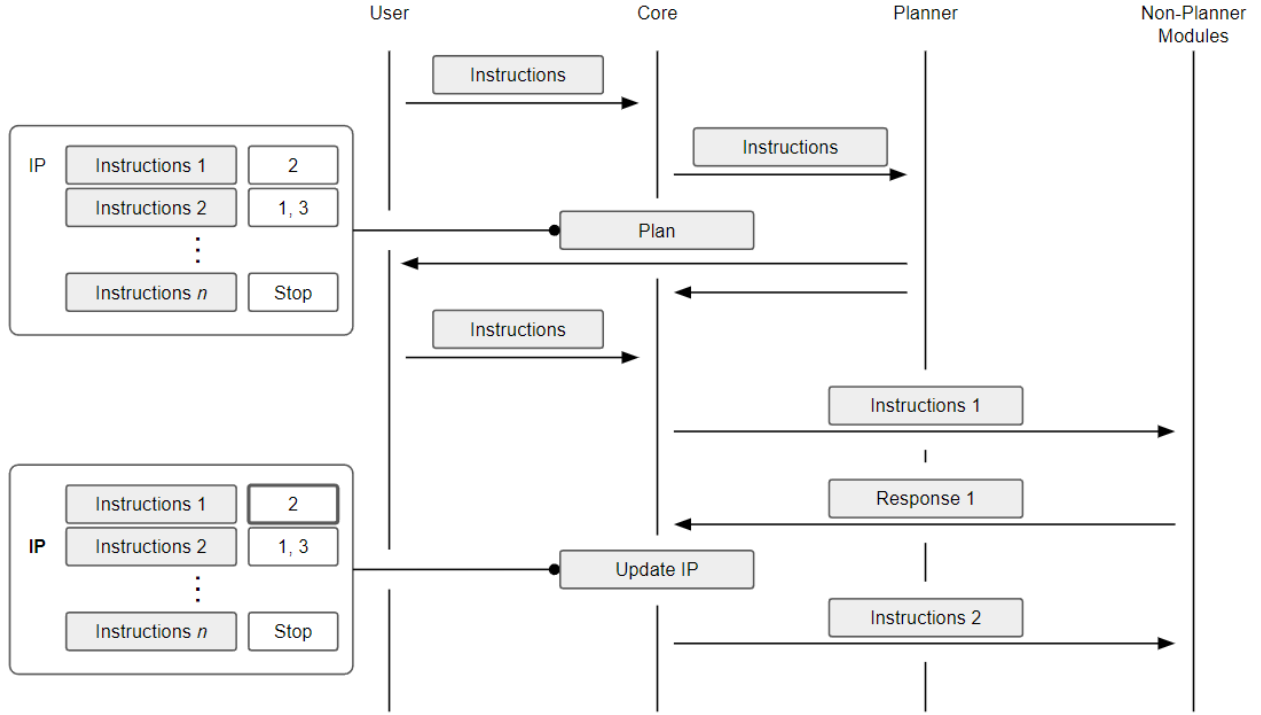


Fig 9. Sequence Diagram for Multi-module Plans with BRAD. Information is shared between four areas: the *user* is a human; the *core* refers to the core functionality of BRAD for managing the state, queued list of prompts, and other information; the **Planner** module; and the non-planner modules, which 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

1.6 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 scheduler’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.

1.7 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 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 Show 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 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 reasoned through examples as input to an LLM to demonstrate to 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 users 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.

2 Results and Examples

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

2.1 Lab Notebook Evaluation

2.1.1 Qualitative Example

Since all the BRAD responses to Q1 were too long, the complete output is provided here.

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.

2.1.2 RAG Assessment (RAGAs) evaluation metrics

Five metrics were used to evaluate the utility of BRAD's RAG system:

- **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.

All metrics are from 0 to 1 with higher scores indicating better performance.

In order to determine these five metrics, a combination of the following four components were used: question, response, context, and ground truth. Since the questions are based on cutting edge research topics, their ground truth is based in research papers in our aforementioned database. The context is the chunks of text RAG uses to generate a response.

To develop quantifiable metrics from LLM responses, ([14]) proposed the following scheme. 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

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

Questions for Quantitative Experiment

Provided is a list of questions used for the Quantitative experiment.

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 Sis interference in V_j - J_j 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?

2.2 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 integrated into BRAD, 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. 10, 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 module, BRAD will correctly identify a lack of information on this topic and can trigger a web search of this topic. Realizing the insufficiency of information, 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 scFMs and a discussion of their use cases.

This process outlined in fig. 10 is designed by the planner module with clear instructions from the user. The LLM engine of BRAD is used at multiple stages throughout this process. Besides its use in the RAG and Web Search modules, the LLM also acts as an evaluator to determine the quality of the RAG’s responses. Once the response meets sufficient quality, the LLM signals to exit the loop. Additionally, we have implemented safeguards to prevent the system from executing an infinite loop.

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.

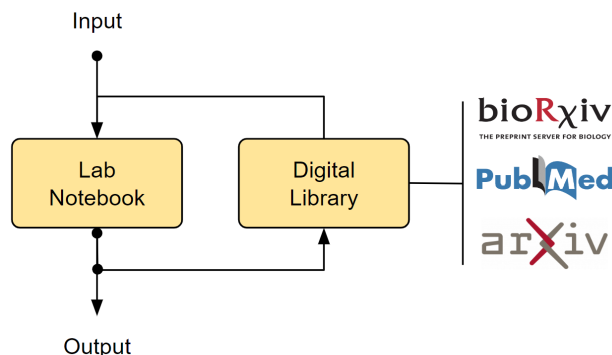
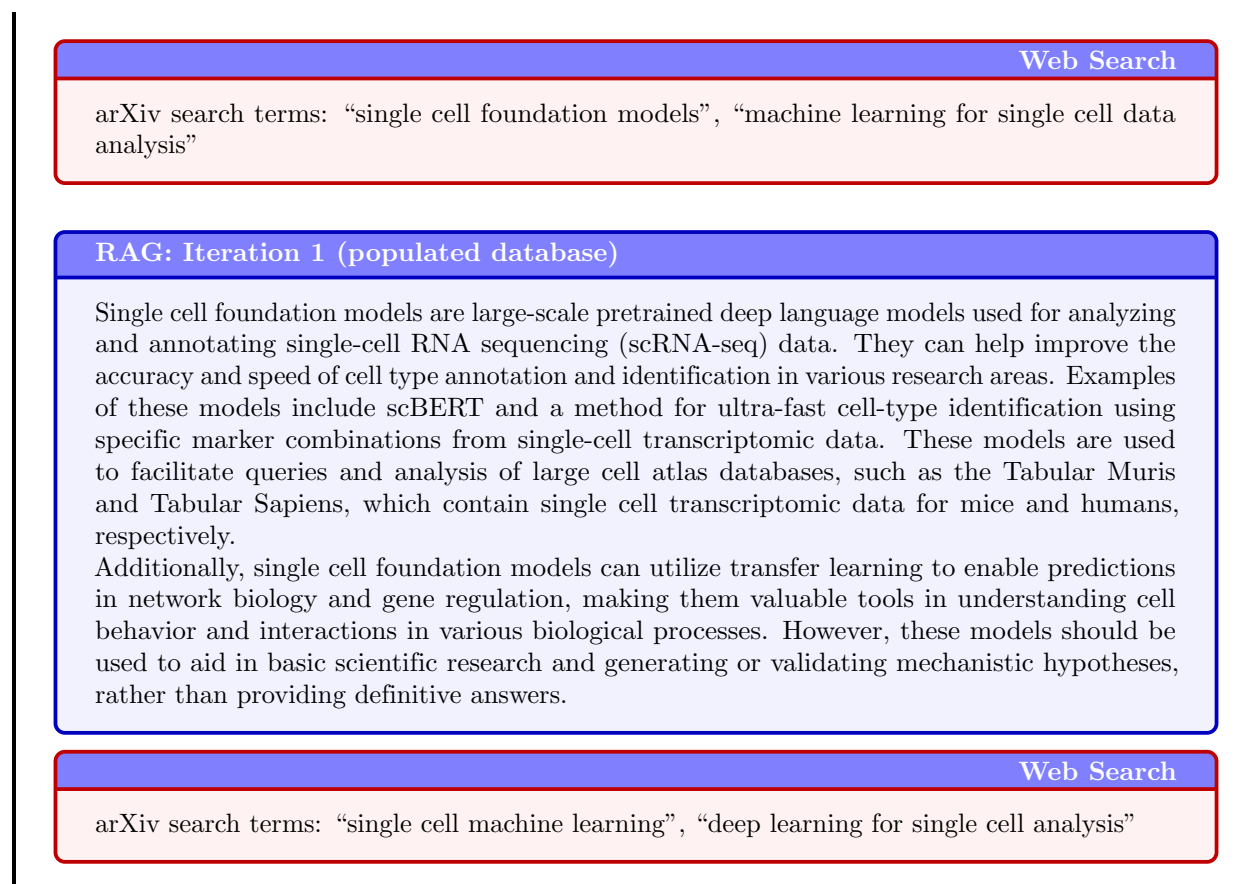


Fig 10. To answer a question with an empty or poorly aligned database, BRAD can iteratively use the Web Search and RAG modules. If the RAG output is assessed as inadequate, the Web Search module retrieves relevant research literature to enhance the answer. This loop may be repeated until the answer to the user's question improves.



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.

2.3 Advanced Search

Below, 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 executed a pipeline consisting of (1) running the **SOFTWARE** module to find first order interactions of PCNA in the **STRING** and **HuRI** databases [27, 24], (2) performing gene enrichment with the **Digital Library** module, and (3) using the **RAG** system in the **Digital Library** 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. 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 **latex** 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 final main section is a summary of the outputs, in which BRAD lists the main information retrieved from the **Lab Notebook** module as well a 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, we find that the reports often reflect or summarize the pipelines well. That said, BRAD can also provide 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

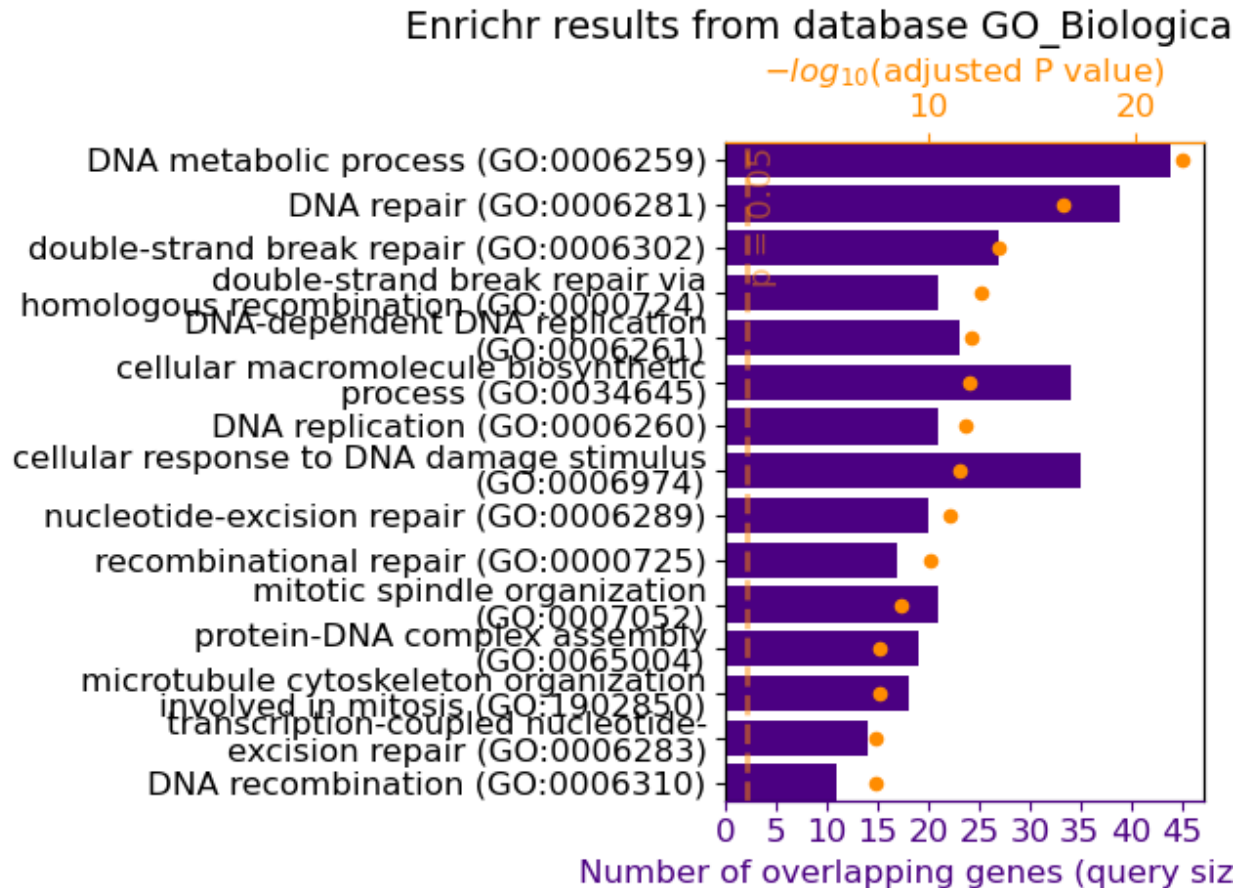
PCNA and Cell Cycle Phases PCNA (Proliferating Cell Nuclear Antigen) plays a crucial role in identifying cell cycle phases. The formation of PCNA foci is tightly correlated with the S phase onset, with an offset of approximately 5 minutes. PCNA loading occurs before PIP-mVenus degradation, and measuring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



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