Paper ML07

Integrating LLMs Using R Shiny for Clinical Data Review by Ensuring Data Privacy and Validity

Peng Zhang, CIMS Global, New Jersey, USA
Ziqian He, CIMS Global, New Jersey, USA
Zhen Wu, CIMS Global, New Jersey, USA
Tai Xie, CIMS Global, New Jersey, USA

ABSTRACT

There is growing interest in open-source analytics and interactive data platforms for healthcare research and clinical applications [1] and R Shiny applications have become the standard for visualizing clinical data. Despite this shift, traditional dashboard interfaces often require significant technical expertise and a deep understanding of data structures, posing challenges for clinical reviewers. To bridge this gap, we introduce DataChat, an innovative R Shiny application that enables conversational interaction with datasets through a natural language interface. By integrating large language models with retrieval-augmented generation (RAG) [2] and leveraging the {ellmer}, {shinychat}, and {ragnar} packages, DataChat allows users to generate dynamic plots and tables while grounding responses in validated statistical outputs. The framework prioritizes data privacy and statistical integrity by performing computations locally and restricting data exposure, ensuring compliance within regulated environments. This approach demonstrates how AI-augmented tools can streamline clinical data review and make complex exploration accessible to non-programmers and cross-functional stakeholders.

INTRODUCTION

Clinical trials produce large volumes of structured data that require repeated review by statisticians, clinicians, and safety experts. In recent years, R Shiny has become a common platform for presenting these data in an interactive form. Compared with static outputs, Shiny applications allow users to explore tables and figures more efficiently during data review.

Despite these advantages, many Shiny applications still rely on predefined filters and fixed user interfaces. Reviewers are often required to understand dataset structure, variable naming conventions, and population definitions in order to locate relevant information. When new questions arise during review meetings, additional programming support is usually required. This limits the flexibility of interactive dashboards and slows the review process.

Recent advances in artificial intelligence, particularly in large language models and retrieval-augmented systems, suggest new ways to interact with data. Rather than navigating menus and filters, users may be able to ask questions directly using natural language. However, the application of these technologies in clinical settings raises concerns related to correctness, reproducibility, and data privacy. These concerns are especially important in regulated environments, where analytical results must be traceable and verifiable.

DataChat is designed to explore how conversational interfaces can support clinical data review while respecting these constraints.

SHINY AND CLINICAL DATA REVIEW

R Shiny is widely used in clinical research to support exploratory analysis and data visualization. Shiny applications allow programmers to encode complex analysis logic behind interactive controls, making data accessible to a broader audience. This approach has been adopted for safety monitoring, efficacy review, and operational reporting across trial phases. However, Shiny applications are typically designed around anticipated use cases. Filters, selectors, and plots must be defined in advance. When reviewers ask questions outside the original design, the dashboard may not provide a direct answer. In practice, this leads to frequent back-and-forth communication between reviewers and programmers.

DataChat approaches this problem by introducing a conversational layer on top of existing Shiny workflows. Rather than replacing dashboards, it provides an alternative way to access the same validated analysis logic.

LARGE LANGUAGE MODELS IN CLINICAL RESEARCH

Large language models have demonstrated strong performance in natural language understanding and generation tasks. Transformer-based architecture enables models to capture contextual relationships across long text sequences. In healthcare research, language models have been applied to document summarization, clinical question answering, and automated reporting [3].

At the same time, multiple studies report that language models may generate fluent but incorrect responses when operating without explicit grounding in domain knowledge. This phenomenon, often described as hallucination, presents a significant challenge in clinical applications [4]. Errors of this type are difficult to detect and undermine trust in automated systems.

These observations suggest that language models should be used as supportive reasoning components rather than autonomous analytical engines. In DataChat, the language model interprets user intent and generates analysis instructions, while all statistical computations remain under the control of validated R code.

RETRIEVAL-AUGMENTED GENERATION (RAG)

Retrieval-augmented generation combines information retrieval with language model generation to improve factual consistency. The approach was formalized by Lewis et al. [2] who showed that retrieving relevant information during generation improves performance on knowledge-intensive tasks.

In healthcare and biomedical research, retrieval-based systems reduce factual errors by grounding model outputs in domain-specific sources [1, 5]. Recent work extends this idea to multimodal clinical data, demonstrating improved reliability in complex reporting tasks.

Most existing retrieval-augmented systems focus on generating narrative outputs. Fewer studies address interactive data analysis, where users expect reproducible numerical results and transparent analytical logic. DataChat adopts retrieval-augmented generation for a different purpose. Retrieved metadata provide context for code generation rather than serving as evidence for narrative text. This design aligns with the requirements of clinical data review, where analytical correctness and traceability are essential.

METHODOLOGY

SYSTEM OVERVIEW

DataChat is implemented as an R Shiny application with modular architecture. The system is designed to support conversational data exploration. Users interact with DataChat through a chat interface embedded within the Shiny environment. All data processing and visualization occur within the same R session.

The system (Figure 1) integrates three main components: a conversational interface, a retrieval layer based on dataset metadata, and a local execution layer that generates tables and figures using validated R code. The large language model acts as an orchestration component rather than a computational engine.

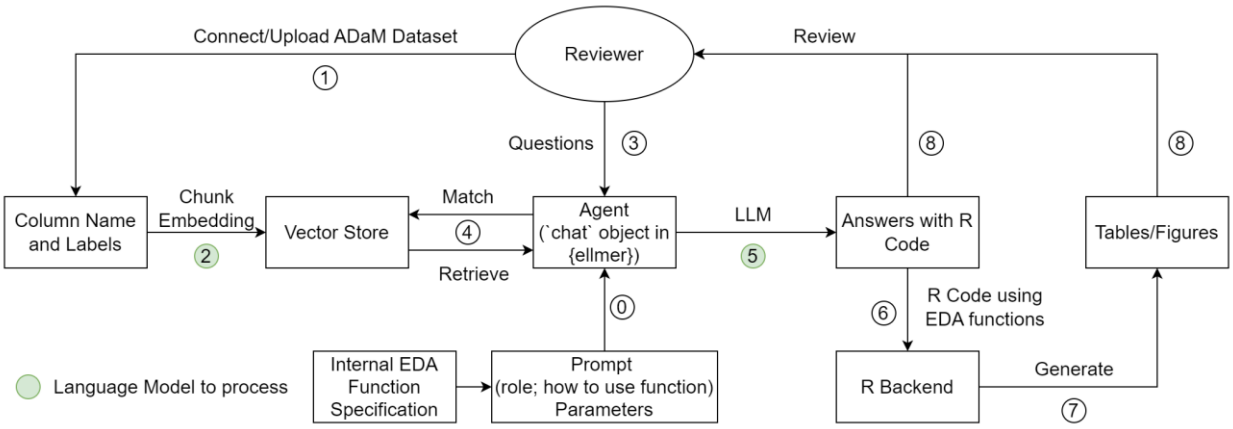

Figure 1: System of DataChat

CONVERSATIONAL INTERFACE AND SESSION MANAGEMENT VIA {SHINYCHAT}

The user-facing layer of DataChat is constructed using the {shinychat} (Figure 2) framework, which facilitates a reactive, stateful communication channel between the clinical reviewer and the R server. Unlike traditional graphical user interface (GUI) inputs that rely on predefined controls and static filters, {shinychat} implements a message-passing architecture that preserves the chronological and logical sequence of the inquiry. This interface is responsible exclusively for user interaction and message presentation, managing the display of queries, system responses, and intermediate execution statuses without performing reasoning or statistical computation. The system maintains a persistent conversation state within the Shiny session, enabling anaphoric reference resolution, where the model understands linguistic pointers to previously generated outputs, and supporting iterative, multi-turn exploration that aligns with the dynamic evolution of questions during clinical review meetings.

To maintain clinical focus and manage cognitive load, the interface utilizes a hierarchical rendering strategy based on the principle of progressive disclosure. High-level clinical interpretations are presented as the primary response, while the underlying technical logs, such as the specific R code used for generation and raw execution traces, are nested within collapsible UI elements. This ensures transparency and auditability without overwhelming the non-programmer stakeholder. Furthermore, {shinychat} serves as a dynamic container for Shiny-compatible objects, allowing validated tables and interactive figures, generated via standard clinical programming packages, to be injected directly into the chat stream for contextual review.

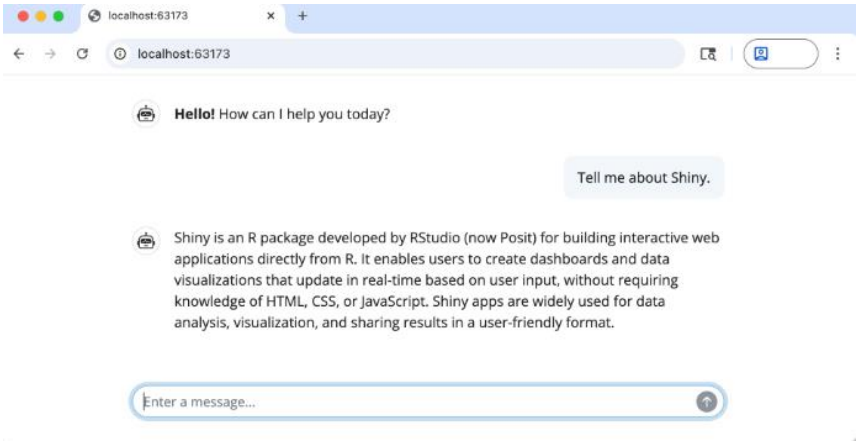

Figure 2: shinychat

LLM ORCHESTRATION AND DETERMINISTIC TOOL INVOCATION VIA {ELLMER}

DataChat utilizes the {ellmer} package as the primary orchestration engine, positioning the Large Language Model (LLM) as a reasoning agent rather than a primary computational engine. This architectural design enforces a strict separation between probabilistic intent interpretation and deterministic statistical execution, a critical requirement in regulated environments where numerical accuracy is paramount. Analytical capabilities are exposed to the LLM via a formalized function registry, where each R function is registered with {ellmer} using precise metadata, including parameter types, default values, and domain-specific descriptions. Upon receiving a query, the LLM performs a semantic mapping to identify the appropriate analytical action and extracts required parameters to emit a structured tool call (Figure 3) rather than unconstrained code.

This tool-constrained design ensures that all statistical computations remain auditable and restricted to validated analytical operators, effectively eliminating the risk of LLM "hallucinations" in numerical outputs. Because the LLM only interacts with the clinical data through these registered tools, the system mitigates failure modes associated with direct model-generated computation. This framework allows DataChat to be integrated into current review processes without requiring extensive re-validation of the language model itself, as the actual data processing relies on established, validated R code.

Function/Component	Technical Role
`chat$stream_async()`	Generates future/promise objects for non-blocking execution; releases R thread.
`chat_append()`	Handles SSE subscription and incremental DOM updates (chunk-based rendering).
`chat$register_tool()`	Performs introspection on R functions to generate JSON Schema for agents.
`R6 Class Object`	Manages state persistence, context windows, and token serialization within the session.
`shinychat UI`	Implements Progressive Disclosure and multimodal rendering (**htmltools**/**gt** support).

Figure 3: Technical specification summary

METADATA-AUGMENTED RETRIEVAL AND PRIVACY GOVERNANCE VIA {RAGNAR}

To ground the conversational reasoning in the specific context of a clinical trial, DataChat employs the {ragnar} package to implement a retrieval-augmented generation (RAG) architecture limited strictly to metadata. This retrieval strategy is based on study-level artifacts, including variable definitions, labels, population rules, and analysis specifications authored within the R application environment. Subject-level data and numerical results are explicitly excluded from the retrieval process to reduce privacy risks and maintain compliance with data governance requirements in regulated clinical settings. {ragnar} transforms these metadata elements into semantic embeddings to construct a vector-based index that supports similarity-based retrieval during each interaction.

Relevant metadata shards are dynamically injected into the conversational context, providing the LLM with the necessary "procedural context" to inform tool selection and parameterization. For example, when a user inquires about patient demographics, {ragnar} retrieves the relevant variable mappings (e.g., AGE, SEX, RACE) from the metadata index, which {ellmer} then uses to construct a valid tool call. By grounding code generation in verified metadata and maintaining all computation within the local R environment, this design ensures that the system provides reproducible numerical results and transparent analytical logic while strictly protecting patient-level data from exposure to external model APIs.

By limiting retrieval to metadata and avoiding direct exposure of subject-level records (Figure 4), this approach can reduce privacy risk and support governance requirements in regulated environments [5]. This retrieval strategy also improves interpretability. When a piece of metadata is used to generate code, its role in the analysis can be traced and reviewed.

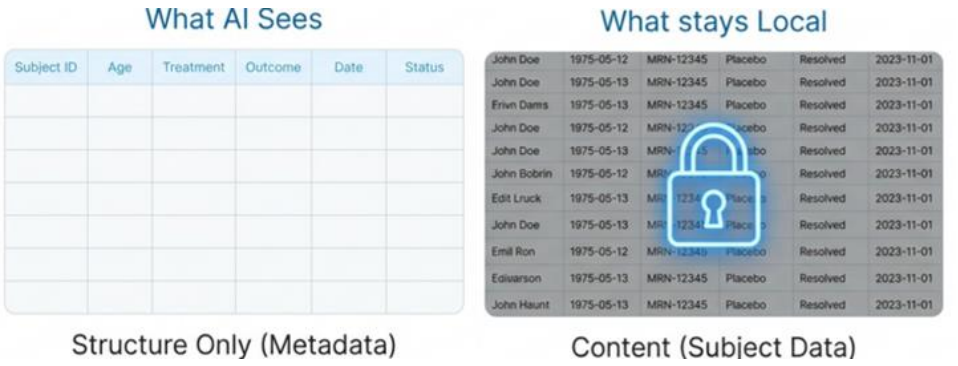

Figure 4: Metadata and privacy risk

LOCAL EXECUTION, VALIDATION AND VISUALIZATION

All statistical calculations are performed locally within the R environment. Tables and figures are generated using standard R packages commonly adopted in clinical programming workflows. Generated outputs are displayed directly in the Shiny interface. Because the execution layer relies on existing validated code, DataChat can be integrated into current review processes without additional validation of the language model itself. The system supports reproducibility by ensuring that the same inputs produce the same outputs, independent of conversational phrasing.

INNOVATION AND RESEARCH DIFFICULTIES

The main contribution of DataChat lies in its use of conversational interaction as a programming interface rather than a reporting tool. The system treats the language model as an assistant that helps construct analysis steps, while statistical logic remains fully controlled by R code. This approach differs from systems that rely on language models to generate final analytical outputs. By restricting retrieval to metadata and analysis specifications, DataChat addresses common concerns related to data privacy and model hallucination. The system also supports iterative exploration, allowing reviewers to refine questions during data review sessions without requiring changes to the underlying dashboard structure.

CONCLUSION

DataChat is intended to complement existing clinical data review tools rather than replace them. It provides an alternative interaction layer that reduces the need for predefined filters and menus. This can be particularly useful in review meetings, where questions evolve dynamically and time for additional programming is limited.The system also highlights several open challenges. Retrieval quality becomes increasingly important as datasets grow more complex. In addition, while statistical results are deterministic, language model–driven code generation introduces variability that must be considered during validation. These challenges are consistent with broader discussions on the use of generative AI in regulated environments.

DataChat demonstrates how large language models can be integrated into R Shiny applications to support conversational clinical data review. By separating reasoning from computation and limiting data exposure, the framework enables natural language interaction while maintaining statistical rigor and data privacy. The system provides a practical example of how generative AI can be used to support, rather than replace, established clinical programming workflows.

REFERENCES

[1] Smolyak, Daniel et al. “Large language models and synthetic health data: progress and prospects.” *JAMIA open* vol. 7,4 ooae114. 26 Oct. 2024, doi:10.1093/jamiaopen/ooae114

[2] Lewis, Patrick et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.” *ArXiv* abs/2005.11401 (2020): n. pag.

[3] Thirunavukarasu, Arun James et al. “Large language models in medicine.” *Nature medicine* vol. 29,8 (2023): 1930-1940. doi:10.1038/s41591-023-02448-8

[4] Agrawal, Atharva Mangeshkumar, et al. "Conversation AI Dialog for Medicare powered by Finetuning and Retrieval Augmented Generation." *arXiv preprint arXiv:2502.02249* (2025).

[5] Kuo, Sheng-Ming, et al. "Automated Clinical Trial Data Analysis and Report Generation by Integrating Retrieval-Augmented Generation (RAG) and Large Language Model (LLM) Technologies." *AI* 6.8 (2025): 188.

ACKNOWLEDGMENTS

We would like to express our deepest gratitude to reviewers for reviewing our work. Their insightful comments and feedback have greatly improved our manuscript.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Peng Zhang
Company: CIMS Global
Address: 285 Davidson Ave, Somerset, NJ 08873
Work Phone: 1-908-790-8888
Email: pzhang@cims-global.com
Website: https://cims-global.com/