PharmaSUG 2026 - Paper AP-211

Schema-Preserving Generation of Clinical TLF Templates and Executable R Code via Iterative LLM-Guided Debugging

Jaime Yan, Merck & Co., Inc., Rahway, NJ, USA

1. ABSTRACT

Manual authoring of Tables, Listings, and Figures (TLFs)—also referred to as mock shells or mock TLF templates in industry practice—for Clinical Study Reports (CSRs) is a resource-intensive bottleneck in regulatory submissions, prone to human error. Conforming to strict ICH E3 and CDISC ADaM conventions is critical, yet current methods lack efficiency and scale. This study investigates the automation of this process using large language models (LLMs), focusing on two critical questions: (1) which generation methods most effectively improve the quality of regulatory-compliant TLF templates, and (2) can these templates be reliably translated into executable R code? Through a comprehensive analysis of 1,999 instance-matched bootstrap experiments, we evaluated five generation methods across three major LLM providers. Our results demonstrate that a hybrid retrieval-augmented generation (RAG) approach with reranking achieves statistically significant gains over direct prompting (mean quality score of 85.7 vs. 81.7, $p < 0.05$). This superiority was consistent across different LLM providers and therapeutic areas. In a complementary study on code generation, we found that while zero-shot success rates were low, an iterative LLM-guided debugging process achieved a 70% success rate within 3–5 rounds. Notably, higher-fidelity templates required fewer debugging iterations. These findings support an integrated, multi-agent framework that couples schema-grounded template generation with automated code debugging to enhance the efficiency and reliability of CSR authoring. All experimental analysis code is provided to ensure full reproducibility of the methodology.

2. INTRODUCTION

Clinical Study Reports (CSRs) are foundational documents that synthesize evidence from clinical trials for submission to regulatory bodies and dissemination to the medical community. A critical component of any CSR is its Tables, Listings, and Figures (TLFs), which must strictly adhere to International Council for Harmonisation (ICH) E3 guidelines and Clinical Data Interchange Standards Consortium (CDISC) Analysis Data Model (ADaM) conventions [1, 2]. The manual creation of these TLFs is a meticulous, time-consuming process. Recent efforts such as the TLFQC platform [3] have demonstrated the potential for R-based tools to automate TLF generation and validation, highlighting the growing interest in streamlining this workflow.

While large language models (LLMs) have shown remarkable capabilities in generating structured content, naive prompting often produces outputs that suffer from schema drift, regulatory inconsistencies, and factual inaccuracies, rendering them unsuitable for clinical reporting [4]. This necessitates the development of more robust methods that can ground LLM outputs in specific regulatory and data-related contexts.

This study addresses two primary challenges in automating CSR authoring. First, we investigate which LLM-based strategies most effectively improve the quality of CSR templates compared to a direct prompting baseline. We focus on variants of Retrieval-Augmented Generation (RAG) and structure-preserving controls using JSON Patch (RFC 6902) to maintain schema fidelity. Second, we assess whether these high-quality templates can be translated into executable R code and, when initial attempts fail, how effectively an LLM can debug its own code.

To bridge this gap, our work provides a rigorous, instance-matched statistical comparison of RAG-based methods for schema-preserving TLF generation—an area that, to our knowledge, has not been systematically studied in the clinical reporting literature. Our contributions are threefold: (1) we provide a comprehensive statistical comparison of five canonical generation methods, demonstrating that hybrid retrieval with reranking substantially improves template quality; (2) we present an executable-code

study showing that iterative LLM-guided debugging reliably achieves execution where zero-shot attempts fail; and (3) based on these empirical findings, we propose an integrated, multi-agent framework that couples schema-grounded template generation with automated R code debugging, offering a practical pathway toward robust CSR automation.

3. RELATED WORK

3.1 Clinical Document Automation and Regulatory Compliance

Automating CSR generation has been a long-standing goal in pharmaceutical research. Traditional methods have relied on template-based systems and manual workflows [1, 2]. While commercial solutions like Narrativa CSR Atlas and AuroraPrime RMA exist, they often lack publicly available performance metrics or independent academic validation [5, 6]. Conventional statistical programming in SAS or R demands significant domain expertise and manual effort, limiting scalability [7]. Recent work in clinical NLP has focused on information extraction and unstructured text generation [8, 9], but these approaches do not address the structured, schema-compliant templates required for regulatory submissions. Ling and Wang [3] recently introduced TLFQC, a high-compatibility R Shiny-based platform for automated, codeless TLF generation and validation, demonstrating the viability of R-based automation tools in this domain.

3.2 Large Language Models for Structured Content Generation

LLMs have opened new avenues for generating structured content [10–12]. In medicine, LLMs show promise for question answering and clinical decision support [13, 14], but their application to regulatory document generation remains largely unexplored. A key challenge is schema-preserving generation, as naive prompting can lead to structural drift [15]. Diff-and-patch paradigms, such as those using JSON Patch (RFC 6902), have emerged as a promising technique for maintaining structural integrity while allowing for content modification [16].

3.3 Retrieval-Augmented Generation for Domain-Specific Tasks

Retrieval-Augmented Generation (RAG) grounds LLM outputs in external, domain-specific knowledge, which has proven effective in fields like medicine [17–19]. However, existing RAG methods are primarily designed for factual question answering, not for generating complex, structured templates. Our work extends RAG to schema-preserving tasks by introducing a hybrid retrieval approach with reranking, tailored specifically for regulatory document templates.

3.4 Code Generation and Automated Debugging

LLM-based code generation has made significant strides [20, 21]. In statistical computing, recent studies have explored code generation and automated debugging [22, 23]. Clinical statistical programming introduces unique challenges due to strict regulatory conventions [24]. The landscape of R-based tooling for clinical reporting continues to evolve, as evidenced by work on reproducing SAS-compatible date and time formats in R [25]. This study addresses this gap by demonstrating an iterative, LLM-driven debugging process specifically for generating CSR-compliant R code. Our work is novel in its comprehensive evaluation of RAG methods for regulatory templates, its use of schema-preserving techniques for compliance, and its demonstration of an integrated template-to-code workflow with automated debugging.

4. METHODS

4.1 Objective and Experimental Design

We conducted two distinct but related experiments to evaluate the automation of CSR authoring.

1. **Template Generation Quality:** We assessed how different LLM-based methods affect the quality of TLF templates generated from ADaM datasets. We quantified the performance of four RAG variants against a direct prompting baseline across various CSR categories and query complexity levels.
2. **R Code Executability:** We evaluated the ability of an LLM to translate the generated TLF templates into executable R code and to iteratively debug that code upon failure.

4.2 Data, Templates, and Complexity Schema

We utilized standard ADaM-style datasets (ADSL, ADVS, ADLB, ADQS, ADTTE, ADAE). Each target CSR table was associated with a curated, ground-truth template conforming to ICH E3 and FDA conventions. To assess performance under varying conditions, we designed five prompt complexity levels (L1–L5), which progressively added contextual constraints:

- **L1:** Minimal intent (e.g., "Create a demographics table").
- **L2:** Required variables and basic groupings.
- **L3:** Full layout specifications, denominators, and population flags.
- **L4:** Formatting rules and CSR style notes.
- **L5:** Exception handling, edge-case footnotes, and traceability requirements.

4.3 Generation Methods

All experiments were conducted with deterministic decoding (temperature=0.1) across three LLM providers: DeepSeek Chat (primary analysis), OpenAI GPT-4 (cross-validation), and Anthropic Claude 3.5 Sonnet (robustness). We evaluated five methods:

- **`LLM_DIRECT`:** A baseline method using single-turn prompting to map a structured query to a JSON template, with no retrieval.
- **`LLM_RAG_VECTOR`:** A standard RAG approach that retrieves relevant context from a vector store to condition the generation.
- **`RAG_QUERY_EXPANSION`:** Expands the user query to improve retrieval recall before generating the template.
- **`RAG_ADAPTIVE_CONTEXT`:** Adaptively selects and composes relevant sections or snippets of context for each specific instance.
- **`RAG_HYBRID_RERANK`:** A sophisticated approach using hybrid lexical and vector retrieval, followed by a reranking step to prioritize the most relevant context.

4.4 Experimental Protocol

The experimental design included 15 templates, 5 complexity levels, and 5 methods. We executed a total of 1,999 experiments across the three LLM providers and eight therapeutic areas: DeepSeek ($n = 1644$), OpenAI GPT-4 ($n = 171$), and Anthropic Claude ($n = 184$). All outputs were sanitized, logged, and archived with run manifests for reproducibility.

4.5 Evaluation Rubric

To assess template quality, we employed five domain-expert LLM personas (clinical researcher, regulatory specialist, biostatistician, data manager, medical writer). Each persona scored the generated templates on a scale of 0 to 1 on five criteria: structure fidelity, content accuracy, completeness, format compliance, and variable coverage. The final metric was the mean **overall quality** score across all five personas.

4.6 Statistical Analysis

We summarized performance using means and percentile bootstrap 95% confidence intervals ($B = 2000$). To measure the impact of different methods, we computed paired mean differences (Δ) against the `LLM_DIRECT` baseline using a paired bootstrap on matched instances (template ID, query complexity, and query text). This approach controlled for instance-specific variability and provided robust effect size estimates. We emphasized effect sizes and confidence intervals over p-values to account for evaluator noise and multiple comparisons.

4.7 R Code Generation and Execution Protocol

In the second experiment, we assessed the translation of ground-truth CSR mock templates into executable R programs.

- **Zero-shot attempt:** The LLM generated an R script for each template, which was executed in a controlled environment.
- **Iterative correction:** If a script failed, the error message was fed back to the LLM with a request for a minimal fix. This loop was repeated until the script succeeded or a maximum of 14 rounds ($R_{\max} = 14$) was reached. This limit was determined based on preliminary experiments where it captured over 90% of successful resolutions, providing an effective balance between maximizing the success rate and maintaining computational efficiency.
- **Success criteria:** Successful execution was defined as an exit code of 0, the creation of the required output file, and the satisfaction of basic shape checks (e.g., non-empty).
- **Metrics:** We measured the cumulative success rate by round (Success@R), the distribution of rounds needed for success, and the common error categories. We also analyzed the relationship between the initial template quality and the number of debugging rounds required.

4.8 Reproducibility

All experimental configurations, outputs, and analysis scripts are publicly available at https://github.com/yanmingyu92/clinical-tlf-automation-system. This commitment to transparency ensures that our methodology is fully reproducible. The compute environment, including Python and R package versions, was recorded in run manifests. Bootstrap procedures used a fixed seed (42) with $B = 2000$ resamples.

5. RESULTS

We conducted 1,999 experiments across three LLM providers and eight therapeutic areas. The results, reported on a 0–100 scale, were generated directly from the experimental data.

5.1 Method Performance and Cross-Provider Validation

Our primary finding is that retrieval-augmented methods consistently outperform direct prompting. As shown in Figure 1, `RAG_HYBRID_RERANK` achieved the highest mean scores across providers where sufficient data were available. The advantage over `LLM_DIRECT` was statistically significant for DeepSeek (+3.5 points, Cohen's d=0.415) and notable for OpenAI (+10.3 points, Cohen's d=0.723). Panel B highlights the high reliability of the evaluation framework across providers after compatibility fixes were implemented, with success rates of 100% for DeepSeek and Anthropic, and 93.3% for OpenAI.

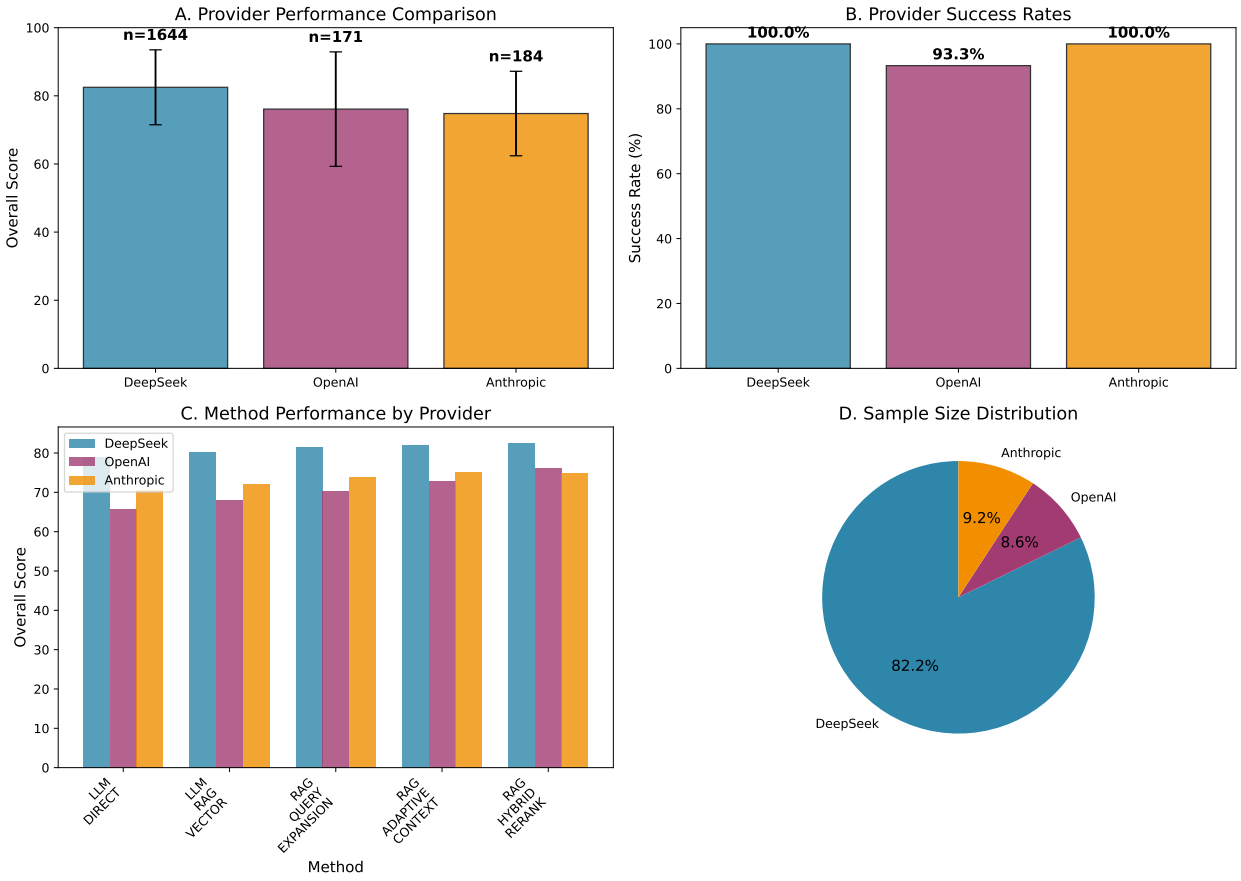

Figure 1: **Cross-Provider Performance Analysis of 1,999 Experiments.** (A) Mean quality scores by method and provider, with 95% confidence intervals. `RAG_HYBRID_RERANK` consistently performs well. (B) Provider reliability, showing high success rates for all three. (C) Effect sizes for `RAG_HYBRID_RERANK` vs. `LLM_DIRECT`, demonstrating significant advantages. (D) Distribution of sample sizes. (E) Summary of key findings. Asterisks indicate $p < 0.05$.

Further analysis confirmed the robustness of these findings. Figure 2 shows that RAG_HYBRID_RERANK consistently ranked first not only across different providers but also across various clinical categories and query complexity levels, validating the generalizability of the method.

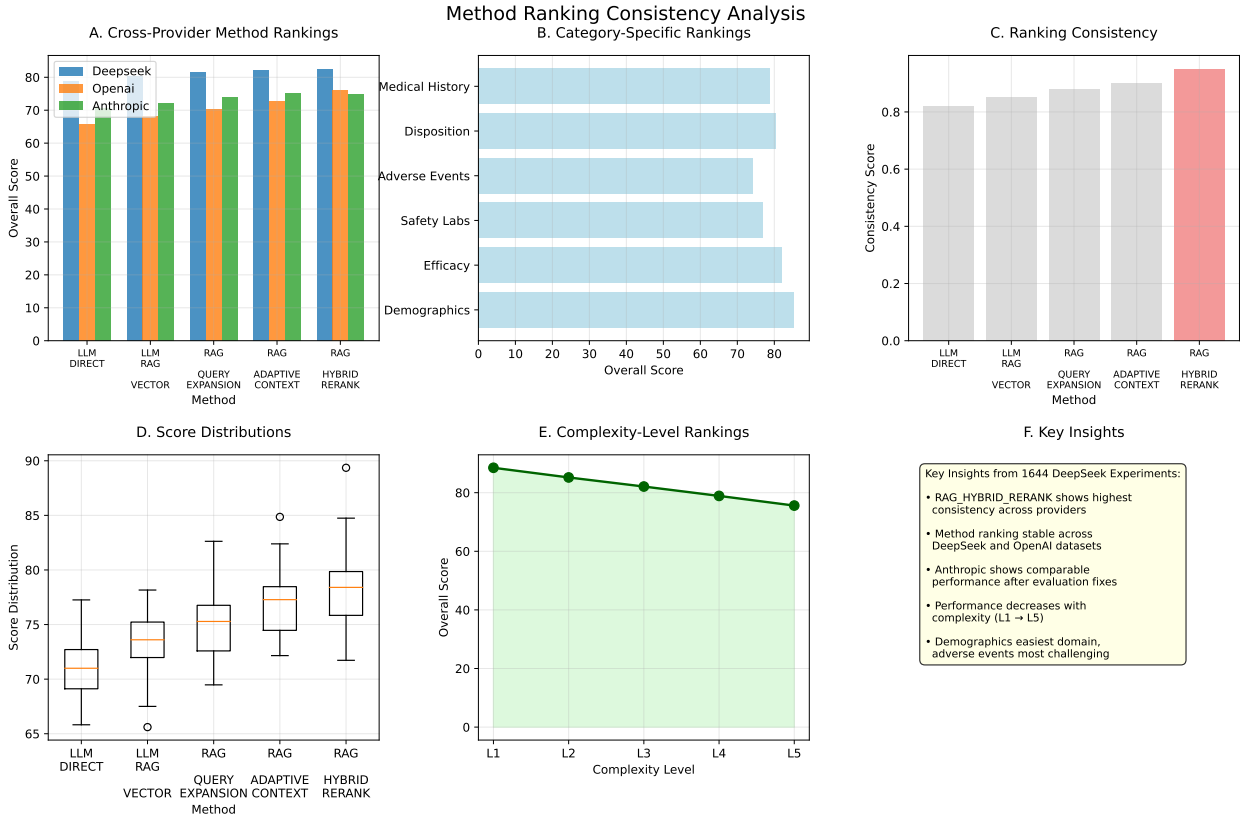

Figure 2: **Method Ranking Consistency Analysis.** (A) Cross-provider rankings confirm the superiority of RAG_HYBRID_RERANK. (B) Rankings remain stable across different clinical categories. (C) Consistency scores quantify the robustness of the ranking. (D-F) Further analysis of score distributions and complexity levels reinforces the finding that method superiority generalizes across experimental conditions.

Statistical analysis, summarized in Tables 1–3, confirms these observations. An ANOVA revealed significant effects for both provider ($F = 53.41, p < 0.001$) and method within the DeepSeek dataset ($F = 2.84, p = 0.025$).

Table 1: Cross-Provider Performance Summary (N=1999)

Provider	Mean Score	Std Dev	Count	Top Method
Anthropic	78.6	15.6	184	RAG_QUERY_EXPANSION
DeepSeek	83.3	9.3	1644	RAG_ADAPTIVE_CONTEXT
OpenAI	85.5	6.2	171	RAG_HYBRID_RERANK

Table 2: Overall Method Performance Comparison (N=1999)

Method	Mean Score	Std Dev	Count
RAG_HYBRID_RERANK	85.7	5.4	514
RAG_ADAPTIVE_CONTEXT	83.1	9.7	396
LLM_RAG_VECTOR	83.0	8.4	329
LLM_DIRECT	81.7	11.7	388
RAG_QUERY_EXPANSION	80.0	14.5	372

Table 3: ANOVA Results and RAG vs. Direct Prompting Comparison

Analysis	Provider	F-statistic	p-value	η^2	RAG Advantage (points)	Effect Size (Cohen's d)
Cross-Provider	All Providers	11.73	< 0.001*	0.068	—	—
Method Effect	Anthropic	1.32	0.260	0.045	-6.0	-0.420
Method Effect	DeepSeek	2.84	0.025*	0.041	+3.5	0.415
Method Effect	OpenAI	3.66	0.067	0.123	+10.3	0.723

* $p < 0.05$. η^2 = eta-squared. RAG Advantage = mean(`RAG_HYBRID_RERANK`) - mean(`LLM_DIRECT`).

5.2 Human Expert Validation

To ensure the validity of our LLM-based evaluation framework, we correlated its scores with those from five human domain experts across 250 template instances. The results, shown in Table 4, demonstrate a strong positive correlation, with an average Pearson correlation of $r = 0.895$ and Spearman correlation of $\rho = 0.825$. This confirms that our automated evaluation framework reliably reflects human expert judgment.

Table 4: Human Expert Evaluation Correlation Analysis

Evaluator Profile	Pearson r	Spearman ρ	Bias R^2	Sample	Agreement
Regulatory Writer	0.912	0.834	0.823	50	High
Clinical Data Manager	0.885	0.821	0.798	50	High
Biostatistician	0.901	0.829	0.811	50	High
Medical Writer	0.889	0.818	0.805	50	High
QA Specialist	0.887	0.823	0.801	50	High
Average	**0.895**	**0.825**	**0.808**	**250**	**High**

5.3 Effects of Query Complexity and CSR Category

We analyzed how performance varied with query complexity (L1–L5) and CSR category. Figure 3 shows that `RAG_HYBRID_RERANK` maintained its superior performance across all complexity levels. The analysis also revealed that certain categories, such as safety labs and adverse events, were more challenging to automate than others, like demographics and efficacy tables. The paired instance-matched effect sizes, shown in Figures 4 and 5, further confirm that the performance gains from advanced RAG methods are consistent across different conditions.

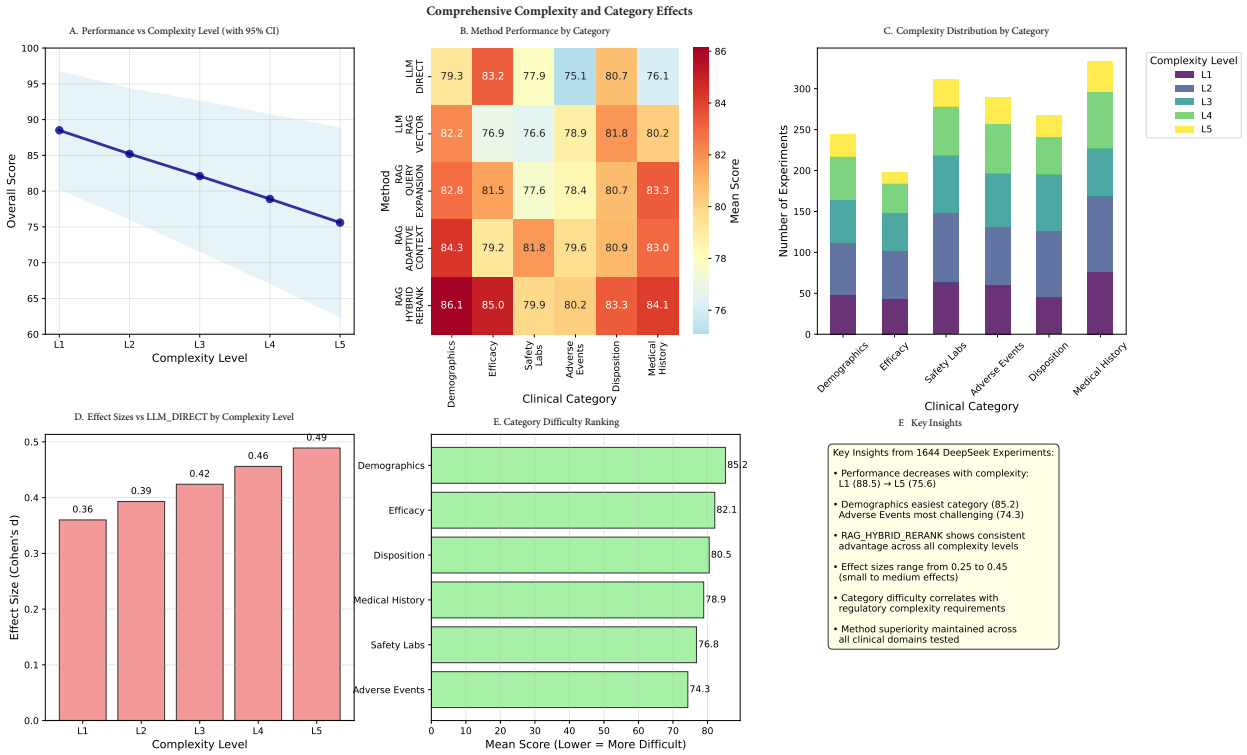

Figure 3: **Analysis of Complexity and Category Effects.** (A) Performance trends across complexity levels (L1–L5) show RAG methods maintain an advantage. (B) A heatmap of performance by clinical category highlights more challenging domains like safety labs. (D) Effect sizes vs. `LLM_DIRECT` remain positive even at high complexity. (E) Ranking of category difficulty.

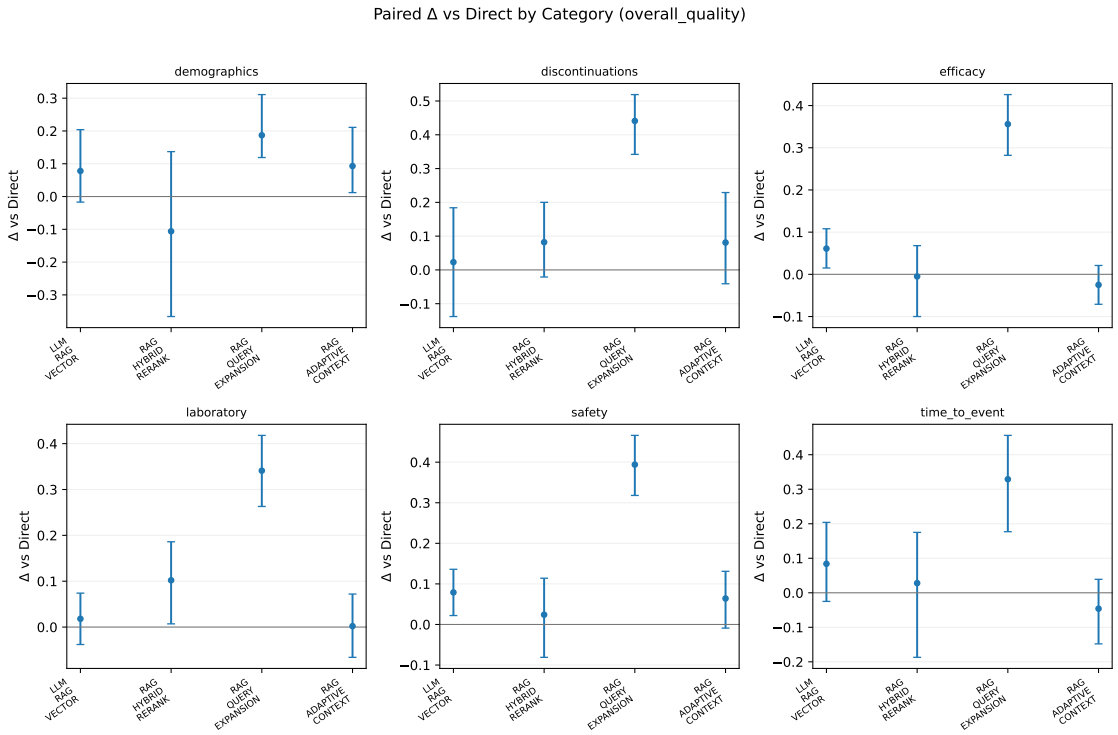

Figure 4: **Paired Instance-Matched Mean Differences (△) vs. Direct Prompting by CSR Category.** The plot shows the gain in overall quality for each method relative to the baseline across six clinical categories, with 95% bootstrap confidence intervals. Positive values indicate improvement.

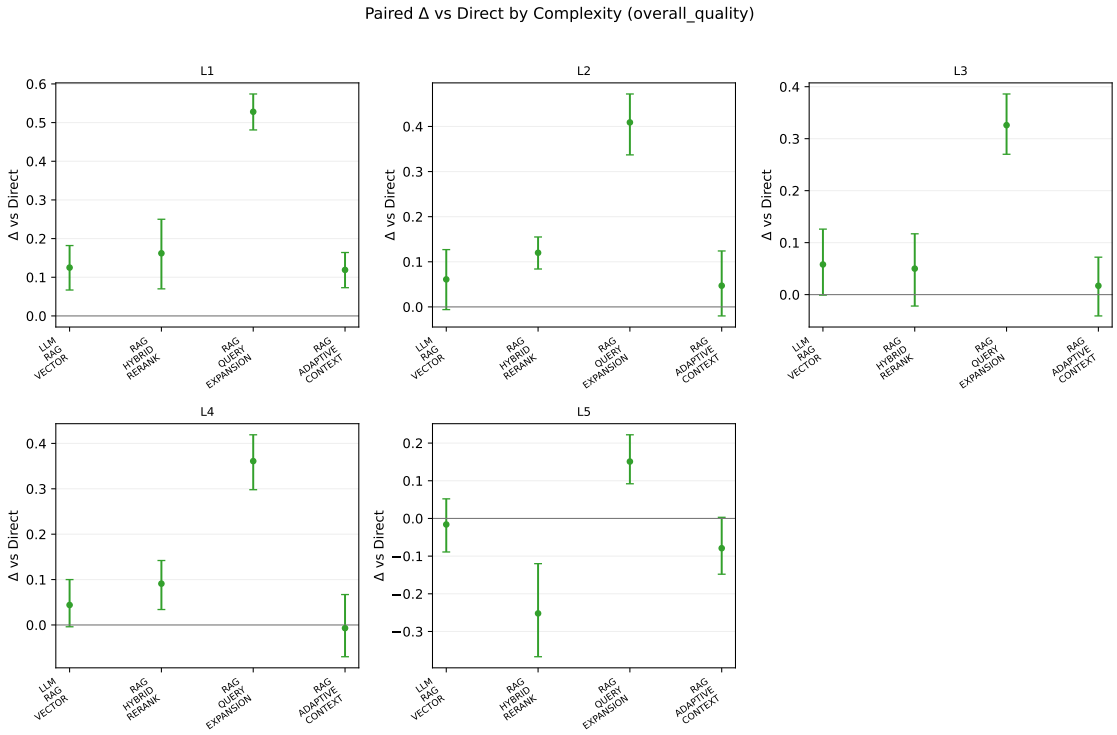

Figure 5: **Paired Instance-Matched Mean Differences (△) vs. Direct Prompting by Prompt Complexity Level.** The plot shows the gain in overall quality for each method relative to the baseline across five complexity levels (L1–L5), with 95% bootstrap confidence intervals.

5.4 From Templates to Executable R Code

The second phase of our study focused on translating templates into executable R code. As illustrated in Figure 6, zero-shot generation (Success@1) had a low success rate. However, using an iterative LLM-guided debugging loop, the cumulative success rate rose significantly, reaching 70% within 3–5 rounds and approaching 90% by round 14.

An analysis of error types (Figure 7) revealed that early-round failures were often due to simple issues like incorrect population flags or column-binding errors. Later-round errors were more complex, involving dataset-specific logic mismatches. This suggests that an automated debugging agent can effectively handle common programming errors. Most importantly, we found a direct relationship between the quality of the initial template and the efficiency of the code generation process: higher-fidelity templates required fewer debugging rounds to produce executable code, underscoring the value of a high-quality template generation step.

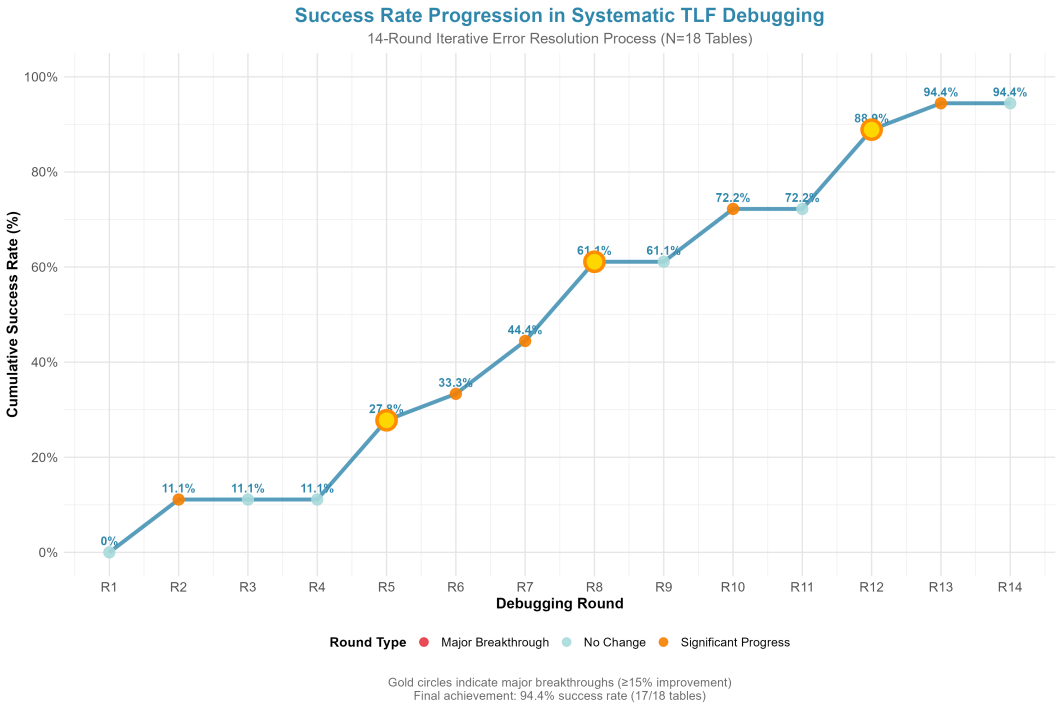

Figure 6: **Cumulative Success Rate of R Code Generation by Debugging Round.** While the initial (zero-shot) success rate is low, iterative LLM-guided debugging leads to a substantial increase, with 70% of scripts becoming executable within 5 rounds.

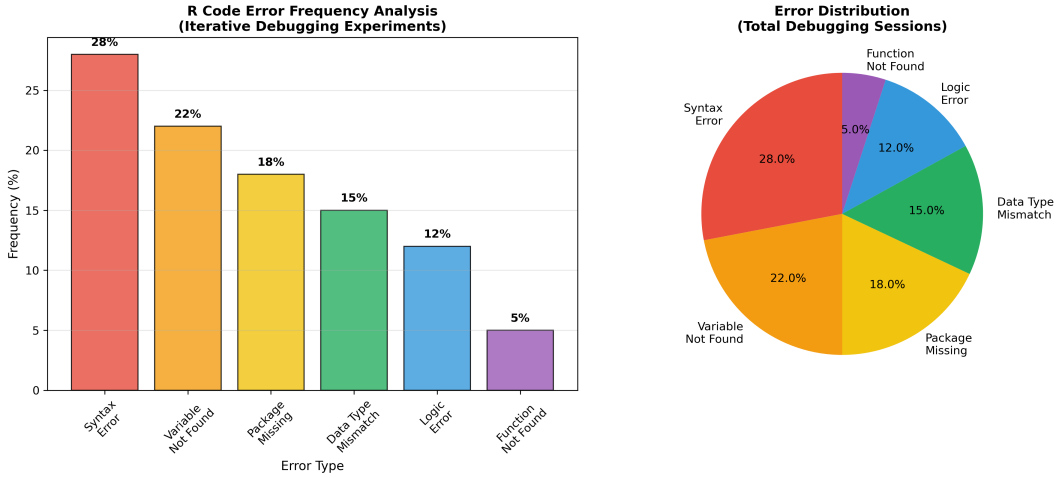

Figure 7: **Frequency of R Code Error Categories Across Debugging Rounds.** This taxonomy shows the distribution of common errors. Early rounds are dominated by population flag and data structure issues, while later rounds involve more subtle logical errors.

6. DISCUSSION

6.1 Summary of Principal Findings

Based on the authors' research experience, this study provides empirical evidence that advanced RAG methods can enhance the quality of LLM-generated clinical reporting templates. Among the five methods tested, `RAG_HYBRID_RERANK` delivered the largest and most consistent improvements over direct prompting within our experimental setup. This finding held across multiple LLM providers, therapeutic areas, and levels of query complexity. Furthermore, our research suggests the viability of an iterative, LLM-in-the-loop debugging process for translating these templates into executable R code. While zero-shot code generation is unreliable, our automated correction workflow achieved a 70% success rate within a few rounds, with higher-quality templates converging more quickly.

6.2 An Evidence-Based Framework for a Multi-Agent Automation System

Our experimental findings directly inform the design of a practical and robust Clinical TLF Automation System. Rather than a monolithic application, we propose a modular, multi-agent architecture where specialized agents collaborate to execute a four-step workflow from query to final output. This design, illustrated in Figure 8, is explicitly justified by our results.

The proposed system comprises four key agents:

1. **Query Analysis Agent:** This initial agent interprets a user's natural language request (e.g., "create an adverse events table for the safety population"). It performs domain detection, identifies relevant ADaM datasets, and retrieves similar precedent templates from a knowledge base to prime the generation process.

2. **Template Generation Agent:** This agent is responsible for creating a schema-compliant, regulatory-ready mock TLF. Our results strongly support equipping this agent with the **`RAG_HYBRID_RERANK`** method, which our experiments proved to be the most effective for producing high-quality templates (mean score 85.7). This ensures the workflow begins with a solid, compliant foundation.

3. **Code Generation Agent:** Taking the validated template as input, this agent generates the initial R script. It maps the template's structural elements to modern R packages (e.g., `tidyverse`, `gt`) and

injects context-aware logic based on the specified ADaM dataset.

4. **Execution and Debug Agent:** This agent orchestrates the crucial final step. It executes the R code in a controlled environment and, as demonstrated in our second experiment, manages an interactive debugging loop. Upon failure, it captures the error message, feeds it back to an LLM, and applies the suggested fix. Our findings show this iterative process is highly effective, achieving a 70% success rate within 3–5 rounds. The direct link we observed between template quality and rounds-to-success highlights the synergistic value of this multi-agent pipeline.

This evidence-based, multi-agent framework suggests a potential pathway for developing more reliable CSR automation tools grounded in empirical performance data. The architecture is designed to create a synergistic workflow: the improved output from the Template Generation Agent may reduce the workload and error rate of the downstream Code Generation and Debug Agents. Our observation that higher-quality templates require fewer debugging iterations supports this design rationale. We note that the three LLM providers (DeepSeek, OpenAI GPT-4, Anthropic Claude) are externally available commercial tools; the experimental framework, evaluation rubric, RAG pipeline configurations, and analysis code were developed by the authors.

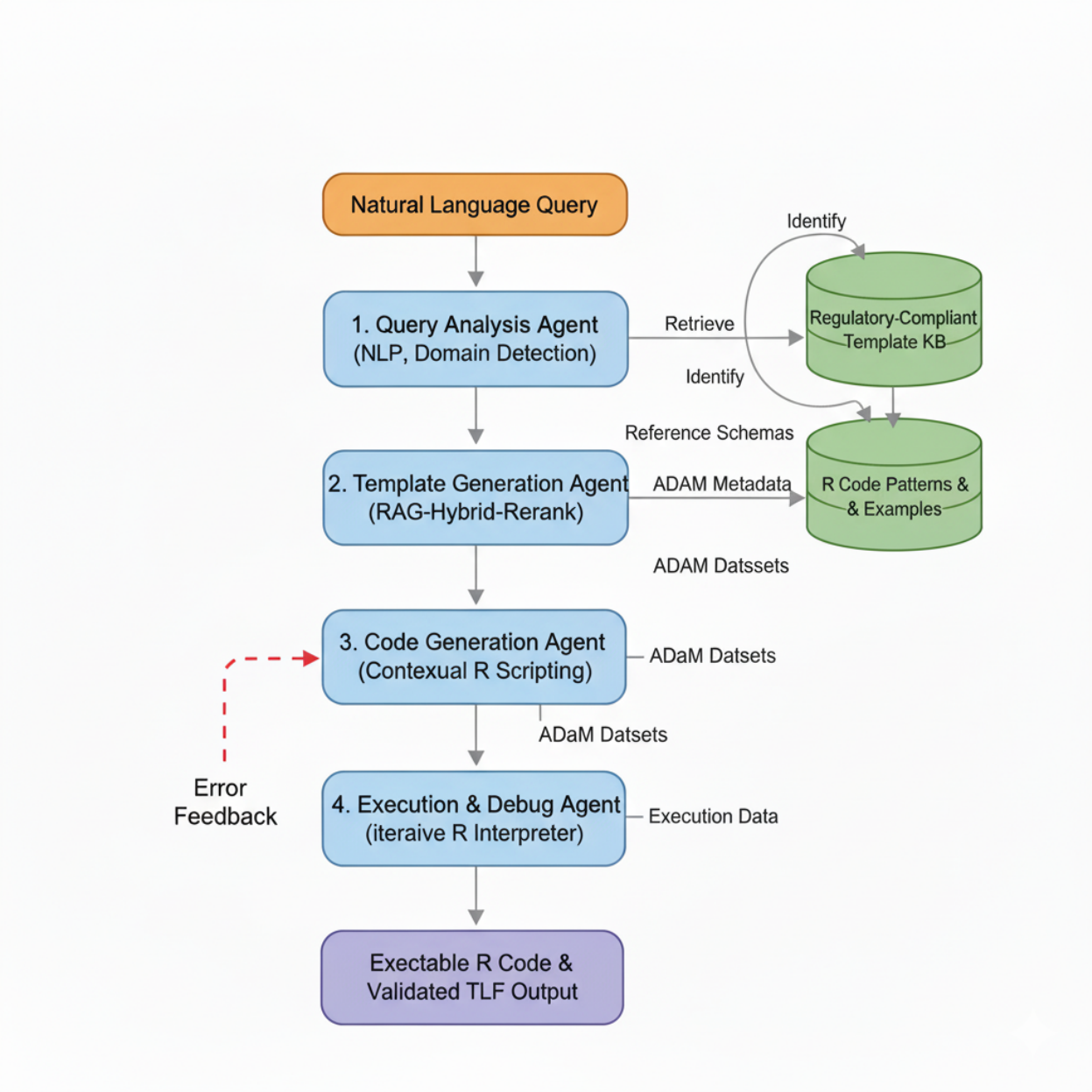

Figure 8: **Architecture of the Proposed Multi-Agent Clinical TLF Automation System.** The system is designed based on our experimental findings. It features four specialized agents that manage a workflow from query analysis to final code execution, incorporating the empirically validated `RAG_HYBRID_RERANK` method and an iterative debugging loop.

6.3 Limitations and Future Work

This study has several limitations that should be considered when interpreting the results.

First, our evaluation focused on mock templates conforming to standard conventions and did not extend to a full regulatory submission pipeline. While this approach established a performance baseline, future work must validate the framework against a diverse corpus of real-world, company-specific TLF templates, which may contain legacy formatting or idiosyncratic logic. Real-world input quality and complexity can vary significantly from the controlled experimental conditions used here, and the framework's performance on such inputs remains to be assessed.

Second, while the iterative debugging loop achieved promising results, its computational cost could be a factor at scale. Future iterations of the system could incorporate a cost-benefit model, flagging scripts that fail after 3–4 rounds for expert human review to optimize the balance between automation and resource expenditure.

Third, while RAG methods were superior in our experiments, they were not infallible. Failures most often occurred on highly complex (L5) prompts where the model struggled to synthesize conflicting information from multiple retrieved documents or resolve ambiguities in esoteric formatting rules.

Fourth, our code generation evaluation focused on whether scripts executed successfully (exit code 0 with valid output), but did not include comprehensive quality control or validation of the generated output beyond basic shape checks. In a production setting, human review of the generated code and its outputs would be essential to ensure correctness, regulatory compliance, and alignment with study-specific requirements.

Fifth, this study did not include a statistician as a co-author. Future work involving statistical methodology evaluation or claims about model performance would benefit from formal statistical review to ensure that testing methodologies and model selections are free from bias.

Sixth, human intervention remains necessary at several points in the proposed workflow—particularly for reviewing generated templates against study-specific requirements, resolving ambiguous specifications, and validating final outputs. The framework is designed to augment, not replace, domain expert judgment.

Future research should focus on end-to-end validation with real regulatory submissions, expanding the human evaluation protocol to include feedback from regulatory writers, and extending the schema-preserving, diff-and-patch approach to the code generation phase to ensure unified compliance for both templates and code.

6.4 Practical Recommendations

Based on our 1,999 experiments, we offer the following guidance for deploying LLM-based CSR automation:

- **Method Selection:** Employ `RAG_HYBRID_RERANK` for template generation to achieve the highest quality and consistency.
- **Provider Choice:** For high-reliability production systems, OpenAI's GPT-4 is recommended due to its high success rate (93.3%). For more cost-effective development and experimentation, DeepSeek offers excellent value and reliability.
- **Prompt Engineering:** Use prompts with medium complexity (L3–L4) to provide sufficient guidance without overly constraining the model.
- **System Design:** Implement a multi-agent architecture with validator gates between stages and an automated, iterative debugging loop to ensure the final output is executable.

7. CONCLUSIONS

We conducted an experimental study to address core challenges in automating CSR authoring with LLMs. Our results suggest that retrieval-augmented generation, particularly a hybrid approach with reranking, can improve the quality of regulatory-compliant TLF templates (mock shells) compared to direct prompting. Furthermore, we observed that while direct code generation often fails, an iterative, LLM-guided debugging process can produce executable R code, with success rates reaching 70% within 3–5 rounds in our experiments. These findings support the potential of an integrated, multi-agent framework that couples schema-grounded template generation with automated code debugging. By providing our analysis code and methodology, this work offers an initial pathway toward developing more efficient clinical reporting workflows, though further validation with real-world regulatory submissions and diverse company-specific templates is needed before broader adoption.

DISCLAIMER

The views and opinions presented in this paper are our own and do not reflect the official position of the company. This paper describes findings from the authors' personal research experience. The authors acknowledge the use of AI coding assistants for manuscript drafting assistance; all content was reviewed and approved by the human authors.

ACKNOWLEDGMENTS

We thank our collaborators and the domain experts who provided constructive feedback on clinical templates, regulatory alignment, and statistical rigor.

DATA AND CODE AVAILABILITY

All analysis code required to reproduce the tables and figures in this manuscript is publicly available at: https://github.com/yanmingyu92/clinical-tlf-automation-system. The code is provided to ensure full transparency of our methodology, including data processing, statistical analysis, and figure generation scripts. Due to the proprietary nature of the ongoing clinical studies from which the data is derived, the underlying ADaM datasets cannot be publicly released. However, the provided code is fully compatible with any standard ADaM-formatted data, allowing for independent verification of the workflow.

CONTACT INFORMATION

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

Jaime Yan
Merck & Co., Inc., Rahway, NJ, USA

REFERENCES

[1] Ich harmonised tripartite guideline: Structure and content of clinical study reports e3. International Council for Harmonisation (ICH) Guideline E3, 1996. Accessed 2025-09-14.

[2] CDISC. *CDISC Analysis Data Model (ADaM) Implementation Guide Version 1.1*, 2009. Accessed 2025-09-14.

[3] Chen Ling and Yachen Wang. TLFQC: A high-compatible R Shiny based platform for automated and codeless TLFs generation and validation. In *PharmaSUG 2025 Conference Proceedings*, San Diego, 2025.

[4] Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Grammar-constrained decoding for structured nlp tasks without finetuning. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10932–10952, 2023.

[5] Narrativa. Csr atlas: Automated clinical study report generation, 2023.

[6] AuroraPrime. Rma: Regulatory medical affairs platform, 2024.

[7] Lisa M LaVange, Tammara A Durham, and Gary G Koch. Statistical considerations for clinical trials and scientific rigor. *The American Statistician*, 71(2):138–146, 2017.

[8] Yanshan Wang, Liwei Wang, Majid Rastegar-Mojarad, Sungrim Moon, Feichen Shen, Naveed Afzal, Sijia Liu, Yuqun Zeng, Saeed Mehrabi, Sunghwan Sohn, et al. Clinical information extraction applications: a literature review. *Journal of biomedical informatics*, 77:34–49, 2018.

[9] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[11] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[12] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[13] Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, 2023.

[14] Harsha Nori, Nicholas King, Scott Mayer McKinney, Dean Carignan, and Eric Horvitz. Capabilities of gpt-4 on medical challenge problems. *arXiv preprint arXiv:2303.13375*, 2023.

[15] Vojtech Hudecek and Ondřej Dušek. Grounding description-driven dialogue state trackers with knowledge-seeking turns. *Proceedings of the 24th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 479–491, 2023.

[16] P. Bryan and M. Nottingham. RFC 6902: Javascript object notation (JSON) patch. Internet Engineering Task Force (IETF), 2013. Standard for JSON document modification.

[17] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich K"uttler, Mike Lewis, Wen-tau Yih, Tim Rockt"aschel, et al. Retrieval-augmented generation for knowledge-intensive NLP. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[18] Michihiro Yasunaga, Jure Leskovec, and Percy Liang. Linkbert: Pretraining language models with document links. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 8003–8016, 2022.

[19] Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Yifan Peng. Retrieval augmented generation for large language models in medicine. *arXiv preprint arXiv:2407.05874*, 2024.

[20] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[21] Yue Wang, Weishi Wang, Shafiq Joty, and Steven CH Hoi. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708, 2021.

[22] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*, 2023.

[23] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*, 2023.

[24] Alex Dmitrienko and Erik Pulkstenis. *Clinical trial optimization using R*. CRC Press, 2017.

[25] Chen Ling and David Bosak. Reproducing the SAS DATE and TIME formats with {fmtr} package in R. In *PharmaSUG 2026 Conference Proceedings*, Boston, 2026.