chatAI4R: Interactive Artificial Intelligence toolkit for Data Science in R

Satoshi Kume [1¶]

1 Bio"Pack"athon, Osaka, Japan ¶ Corresponding author

DOI: 10.21105/joss.09283

Software

- Review
- Repository
- Archive

Editor: Chris Vernon

Reviewers:

- @tzuV
- @khoa-yelo

Submitted: 02 July 2025
Published: 27 March 2026

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

Large Language Models (LLMs) have revolutionized natural language processing (NLP), data mining, and program coding. The chatAI4R package provides a comprehensive toolkit for seamlessly integrating LLMs within R environments. Beyond basic text generation and conversation capabilities, it supports text embeddings and delivers sophisticated LLM assistance through simple function calls, significantly extending R-based data analysis and knowledge discovery processes. Unlike existing R packages, the chatAI4R package offers unique R package development support features. Rather than functioning as a multi-functional API wrapper, it provides comprehensive development automation and AI-assisted data mining capabilities. The package combines command-line and graphical operations, offering flexibility for users across all skill levels. Available on both GitHub and the Comprehensive R Archive Network (CRAN), chatAI4R ensures stability, reliability, and broad community accessibility.

Originally a development aid for R packages since 2023, chatAI4R has evolved into a data-analysis companion by adding interpretation, knowledge extraction, and multi-LLM capabilities. It serves R package developers (automating Roxygen2 docs, function generation, code quality) and data analysts (statistical interpretation, literature processing, insight extraction) with a four-layer architecture that goes beyond multi-functional API wrappers.

State of the Field

Since GPT-4's release (OpenAI, 2024), LLMs have rapidly evolved, transforming NLP, data analysis, and programming approaches. While chat-based interfaces offer intuitive experiences, they are insufficient for complex analytical tasks requiring multi-step processing and statistical integration. Current AI agents still suffer from response time limitations in their speculative processing, which makes them unsuitable for iterative workflows. Therefore, direct programmatic access through R becomes essential, leveraging its rich statistical ecosystem and creating a need for specialized R packages that provide efficient LLM integration for data science applications.

The R ecosystem now includes several LLM-focused packages with distinct approaches. For comprehensive LLM integration, ellmer (Wickham et al., 2025) provides wide provider support with advanced features including streaming outputs, tool calling, and structured data extraction. Basic API access is offered by packages like openai (comprehensive but now archived) (Rudnytskyi, 2024) and gptr (Gu, 2024), which provides a simple interface through its get_response() function for straightforward ChatGPT interactions.

For local LLM deployment, both ollamar (Lin & Safi, 2025) and rollama (Gruber & Weber, 2024) facilitate integration with Ollama, an open-source framework for running local LLMs, enabling private and reproducible model execution focused on text annotation and document embedding capabilities.

Development-focused packages include chatgpt (Rodriguez, 2023) and gptstudio (Nivard et al., 2024), both providing RStudio addins for coding assistance. While chatgpt focuses specifically on OpenAI integration with features like code commenting, auto-completion, and Roxygen2 documentation generation, gptstudio (Nivard et al., 2024) offers broader provider support through a unified interface.

While these existing tools provide valuable functionality, they primarily serve as API wrappers or development assistants, leaving significant gaps in comprehensive R-specific package development support and integrated data analysis workflows. These limitations create an opportunity for a more comprehensive solution that chatAI4R aims to address.

Statement of need

While existing R packages provide basic LLM functionality, critical gaps remain in comprehensive R-specific package development support and integrated data analysis workflows. Current tools serve as general-purpose API wrappers without addressing complex analytical needs.

chatAI4R addresses these limitations through its unique multi-layered conceptual architecture, providing a comprehensive ecosystem for LLM integration specifically designed for R users. The package supports nine LLM API platforms through unified interfaces, including OpenAI GPT models and Google Gemini, and provides innovative access to 18 models simultaneously via io.net's Intelligence API (https://io.net/, as of 14-JAN-2026), a cloud platform that provides distributed GPU computing resources, enabling access to state-of-the-art open-source models such as gpt-oss-120b, DeepSeek-R1, Qwen3, and Llama-4.

The package's core innovation lies in R-specific package development automation. Beyond basic text generation, chatAI4R offers automated R code generation through `createRfunction()`, intelligent comment addition via `addCommentCode()`, automatic Roxygen2 documentation (R's standard documentation format) with `addRoxygenDescription()`, and comprehensive package architecture planning through `designPackage()`, which assists in proposing the overall design and architecture of an R package. These capabilities transform LLMs into powerful development assistants tailored for R programming workflows.

A distinctive feature is the multi-agent discussion system (`discussion_flow_v1()`), where three specialized AI agents—the Beginner Bot, the Expert Bot, and the Peer Reviewer Bot—collaborate through Socratic dialogue (an iterative question-and-answer methodology). This approach enables iterative solution refinement with human intervention at critical decision points, addressing the single-shot interaction limitations present in existing tools.

The chatAI4R package excels in data analysis interpretation through the `interpretResult()` function, providing specialized interpretation for multiple analysis types including PCA and regression. This feature bridges the gap between statistical output and scientific interpretation, a capability that is absent in current R-LLM packages.

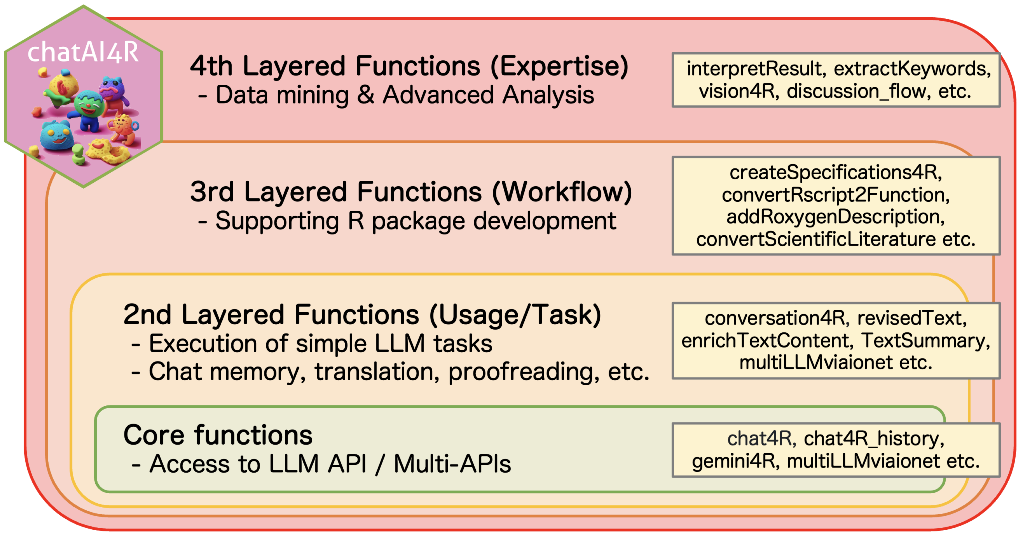
chatAI4R
4th Layered Functions (Expertise)
- Data mining & Advanced Analysis
interpretResult, extractKeywords, vision4R, discussion_flow, etc.
3rd Layered Functions (Workflow)
- Supporting R package development
createSpecifications4R, convertRscript2Function, addRoxygenDescription, convertScientificLiterature etc.
2nd Layered Functions (Usage/Task)
- Execution of simple LLM tasks
- Chat memory, translation, proofreading, etc.
conversation4R, revisedText, enrichTextContent, TextSummary, multiLLMviaionet etc.
Core functions
- Access to LLM API / Multi-APIs
chat4R, chat4R_history, gemini4R, multiLLMviaionet etc.

Figure 1: Four-layered conceptual framework of the chatAI4R package, showing the hierarchical structure from core functions to specialized applications

The package maintains production-level reliability through CRAN distribution, ensuring multi-platform compatibility and rigorous testing. Its open-source nature under the Artistic License 2.0 promotes community contribution while maintaining professional development practices.

Design

The chatAI4R package implements a four-layered application architectural design that provides progressive functionality. This modular approach ensures accessibility for all users.

The Core Functions (Layer 1) establish unified API interfaces for multiple LLM platforms. The `chat4R()`, `chat4R_history()`, `gemini4R()`, `multiLLMviaionet()`, and related functions provide standardized access patterns while handling authentication, rate limiting, and error management.

The Advanced Functions (2nd Layered Functions, Usage/Task) enable execution of simple LLM tasks through intelligent prompt engineering. Functions such as `conversation4R()`, `revisedText()`, `enrichTextContent()`, `TextSummary()`, and `multiLLMviaionet()` provide chat memory management, translation, proofreading, and text processing capabilities.

The Workflow Functions (3rd Layered Functions, Workflow) focus on supporting R package development. The `createSpecifications4R()`, `convertRscript2Function()`, `addRoxygenDescription()`, and `convertScientificLiterature()` functions provide comprehensive development automation, transforming LLMs into powerful assistants tailored for R programming workflows.

The Integration Functions (4th Layered Functions, Expertise) provide data mining and advanced analysis capabilities, while also providing connectivity with R ecosystem tools and development workflows. The `interpretResult()` function employs specialized templates for 13 analysis types, automatically generating domain-appropriate interpretations. The `extractKeywords()` extracts keywords from text, `vision4R()` enables image analysis via Vision API, and the multi-agent discussion system (`discussion_flow_v1()`, `discussion_flow_v2()`) employs role-based prompt engineering where each agent maintains distinct personas, enabling iterative solution refinement.

The package supports integration with six API platforms: OpenAI, Google Gemini, DeepL translation, Replicate, Dify, and IO.net Intelligence API. This comprehensive API connectivity ensures flexibility in model selection and deployment scenarios. The package also facilitates the deployment of R-based web APIs by combining `chatAI4R` with the `plumber` package, allowing users to implement LLM-powered backend processing embedded in external platforms. Additionally, it supports GUI-based interactions, ensuring accessibility for users who prefer graphical interfaces over command-line operations.

The package employs defensive programming with comprehensive error handling, input validation, and graceful degradation when API services are unavailable. Package reliability is ensured through automated testing. Furthermore, to address the challenge of frequent API changes across these platforms, chatAI4R employs a sustainable maintenance strategy combining user contributions with high-frequency, LLM-assisted autonomous code updates, ensuring long-term resilience.

The chatAI4R package provides multiple interaction modes to accommodate different user preferences and workflows. Users can access LLM capabilities through simple function calls or interactive interfaces.

References

Gruber, J. B., & Weber, M. (2024). *Rollama: An R package for using generative large language models through ollama*. https://doi.org/10.48550/arXiv.2404.07654

Gu, W. (2024). *Gptr: A convenient R interface with the OpenAI 'ChatGPT' API*. https://doi.org/10.32614/cran.package.gptr

Lin, H., & Safi, T. (2025). Ollamar: An R package for running large language models. *Journal of Open Source Software*, *10*(105), 7211. https://doi.org/10.21105/joss.07211

Nivard, M., Wade, J., & Calderon, S. (2024). *Gptstudio: Use large language models directly in your development environment*. https://doi.org/10.32614/CRAN.package.gptstudio

OpenAI, O. (2024). *GPT-4 technical report*. https://doi.org/10.48550/arXiv.2303.08774

Rodriguez, J. C. (2023). *Chatgpt: Interface to 'ChatGPT' from R*. https://doi.org/10.32614/CRAN.package.chatgpt

Rudnytskyi, I. (2024). *Openai: R wrapper for OpenAI API*. https://doi.org/10.32614/cran.package.openai

Wickham, H., Cheng, J., Jacobs, A., & Aden-Buie, G. (2025). *Ellmer: Chat with large language models*. https://doi.org/10.32614/CRAN.package.ellmer