The Journal of Open Source Software

LLMR: A unified interface for research with Large Language Models in R

Ali Sanaei [1]

1 Computational Social Science, University of Chicago

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 04 November 2025
Published: unpublished

Summary

LLMR is an R package that provides researchers with a single provider-agnostic interface for generation and embeddings independently and also in inside tidy data workflows. The design enables cross model studies, parameter sweeps, and structured extraction within familiar R idioms. LLMR supports Ollama, OpenAI, Anthropic, Gemini, Groq, DeepSeek, xAI, Together AI, and Voyage AI, and extending the support to new providers is straightforward through writing an S3 call method. LLMR integrates with tidyverse, runs jobs in parallel, caches repeated calls, and offers structured output via JSON Schema. The package is available on CRAN. The source code and the development version are hosted on GitHub.

Statement of need

Researchers now employ large language models to embed text, label text, extract fields, conduct vignette scale experiments, and compare models across prompts and parameters. In R, many available tools bind to a single provider or require people to rewrite code when changing models, which slows comparisons and makes replication difficult (Irudaya Raj, 2023). Local deployments via Ollama are common in privacy sensitive work and teaching, yet local and hosted workflows have previously been referred to separate packages (Gruber & Weber, 2024; Lin & Safi, 2025). LLMR addresses these barriers by treating model calls as data inside tidy pipelines (Wickham et al., 2019). Users define configurations for providers, run model calls as vectorized or row-wise operations, and analyze the resulting columns. This approach would make systematic comparisons and factorial experiments straightforward to set up, and thus researchers retain full control of prompts and parameters while keeping the same idiom for analysis. Similarly, performing combinations of text embedding and text generation is smooth within LLMR.

Features

LLMR provides a consistent configuration object for models and providers, a standard response with finish reasons and token counts, and a tidy pipeline that makes calls over vectors or data frames. It supports structured output using provider switches for JSON Schema, with local parsing and optional validation. Parallel execution uses the future package to scale out jobs, caching can reduce repeated cost during iterative analysis, and error handling applies retries with backoff and records basic diagnostics (Bengtsson, 2021; Wickham et al., 2021). For structured output, OpenAI compatible endpoints accept response_format with json_schema (OpenAI, 2024), Anthropic uses tool definitions with an input_schema (Anthropic, 2025), and Gemini supports response_mime_type and response_schema for JSON output (Google AI for Developers, 2025). In fact the configuration, execution, and parsing steps are kept separate, and thus each portion of the workflow can be replaced or extended without modifying the rest.

The Journal of Open Source Software

Design Logic

All workflows need one or more model configurations via `llm_config`, after that, the model can be called via `call_llm` for either embedding or generative calls. `llm_config` tries to resolve the API key from enviornment variables through a sensible lookup table based on provider ('OPENAI_API_KEY', 'GEMINI_API_KEY', and so on), but can also be directly provided, either as the actual key string (which should be discouraged) or as the name of an environmental variable.

Provider implementations are S3 methods of `call_llm` but given that most providers follow Open AI's specifications, unsupported providers can typically be called using `provider='openai'` but with an overwriting of the `api_url` argument. The rest of the package is essentially wrappers around the `call_llm` function. The first layer is a `call_llm_robust`.

- A stateful chat object can be created by `chat_session` which provides an object with methods like `$send` and `$print`.
- A typical experimental workflow builds an experiment data frame where each row becomes an API call (this can be done via the helper function `build_factorial_experiments`) and then call `call_llm_par` (or `call_llm_par_structured` for structured calls).
- A tidy implementation can use the `llm_mutate` (or `llm_mutate_structured` for structured calls).
- Finally, a typical embedding call can be done via `get_batched_embeddings`.

Using the future package (Bengtsson, 2021) all functions that needs to make more than one api call can be parallized and given the low local computational cost, except for locally run models, there is a considerable speed boost unless a rate-limit is hit from the provider side.

Structured output follows is implemented differently by different providers, and is presently better supported for Open AI and Anthropic models Anthropic (2025). Gemini accepts response_mime_type for JSON and can take response_schema to constrain the shape (Google AI for Developers, 2025). The package also parses and validates JSON locally to keep behavior consistent across providers. Parallel execution uses , and caching during development uses memoise (Wickham et al., 2021). Error handling applies retries with exponential backoff and records concise diagnostics.

Related work

In R, single-provider packages such as openai give direct access to one API (Irudaya Raj, 2023). Several packages target local models through Ollama (Gruber & Weber, 2024; Lin & Safi, 2025). Others provide a tidy interface for chats across vendors (Brüll, 2024; Rapp, 2024). The Ellmer package provides (younger than LLMR) provides cross-platform support and now supports parallization and structured outputs, but lacks support for embeddings, and its design objective is not scientific experimentation.

These tools serve important use cases. LLMR's distinct focus is a unified research workflow inside R that treats model calls as data, supports parallel factorial designs and parameter sweeps, and offers provider-aware structured output with local validation. Libraries in other languages, such as LangChain, give multi-provider abstractions in Python and JavaScript, but they do not integrate with R's data analysis idiom (Chase, 2024).

Quality control

The package includes some unit tests for message normalization, structured parsing and validation, and parallel utilities. Continuous integration runs on Linux, macOS, and Windows. Examples that call external APIs are guarded by environment variables to avoid unintended network use during checks. LLMR is available on CRAN and passes routine CRAN checks for the current release.

Conclusion

LLMR simplifies research with and research about large language models. It makes cross-provider model studies simple inside R by turning model calls into data that can be analyze with standard tools. The package covers hosted and local models, supports structured output, and offers parallel execution with caching and retries, which helps researchers run careful comparisons at scale.

Acknowledgements

The author thanks the R community for feedback and contributions to the package development.

References

Anthropic. (2025). *Tool use with claude*. https://docs.anthropic.com/en/docs/agents-and-tools/tool-use/overview

Bengtsson, H. (2021). A unifying framework for parallel and distributed processing in R using futures. *The R Journal*, *13*(2), 208–227.

Brüll, E. (2024). *Tidyllm: Tidy integration of large language models*. https://CRAN.R-project.org/package=tidyllm

Chase, H. (2024). *LangChain: Building applications with LLMs through composability*. https://python.langchain.com/

Google AI for Developers. (2025). *Structured output, gemini API*. https://ai.google.dev/gemini-api/docs/structured-output

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

Irudaya Raj, V. (2023). *Openai: R interface to OpenAI API*. https://CRAN.R-project.org/package=openai

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

OpenAI. (2024). *Introducing structured outputs in the API*. https://openai.com/index/introducing-structured-outputs-in-the-api/

Rapp, A. (2024). *Tidychatmodels: Chat with all kinds of AI models through a common interface*. https://github.com/AlbertRapp/tidychatmodels

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., & others. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686.

Wickham, H., Hester, J., Chang, W., Müller, K., & Cook, D. (2021). *Memoise: Memoisation of functions*. https://CRAN.R-project.org/package=memoise