2025-09-30

ChatR: A Context-Aware R Programming Assistant

Hanxia Li
Georgia Institute of Technology

Michael Lawrence
R Core
Genentech

Gabriel Becker
R contributor

Executive Summary

Overview

ChatR is an open-source, context-aware assistant for the R ecosystem. In **12 months**, we will deliver an **R package** and **IDE add-in** (with CLI and offline mode) for secure environments. Unlike generic copilots, ChatR runs **inside R**, accessing session objects, packages, and errors to provide accurate, reproducible support for workflows in statistics, bioinformatics, social sciences, etc.

Goals & Methods

- **Reliable help:** Reduce hallucinations; ground answers in official docs with citations.
- **Offline-ready, community-led:** Works in restricted settings; developed openly.
- **Core methods:** R session integration, RAG, safe execution, and agent compatibility

Outcomes

- **Adoption & Efficiency:** R package + IDE plugins with offline support; faster onboarding and expert time savings.
- **Reproducibility:** Grounded, cited, and logged assistance.

Deliverables

- A fully functional, community-adopted R package with IDE plugins and offline support.

Budget

- **~$10,000 total** — ~90% developer effort; ~10% tutorials and community engagement.

Project team

Hanxia Li (Lead Developer, Georgia Institute of Technology) – Responsible for design, implementation, and delivery of ChatR.

Michael Lawrence (Mentor, R Core, Genentech) – Provides guidance on R internals and package design.

Gabriel Becker (Mentor, R contributor) – Advises on architecture, evaluation, and community engagement.

The Problem

- **What the problem is**
  - R workflows (import → transform → model → visualize → report) are complex and fragmented.
  - Generic copilots lack session awareness, hallucinate functions, and cannot run offline, limiting reproducibility and use in secure settings.
- **Who it affects**
  - *Novices*: Steep learning curve, opaque errors.
  - *Experts*: Time lost context-switching and debugging.
  - *High-stakes users*: (healthcare, government) blocked from cloud copilots due to privacy/security.
- **Why it is a problem**
  - Slows research, reduces reproducibility, and raises barriers to adoption.
  - Offline/secure users are excluded from AI support.
- **What solving enables**
  - Productivity gains through faster debugging and search.
  - Reproducibility with version- and citation-aware answers.
  - Accessibility for new users and restricted environments.
- **Existing work**
  - {ragnar} (RAG), {tidyprompt} (prompt structuring), {ellmer} (LLM integration), {btw} (environment capture).
  - Useful but fragmented; none provide a unified, session-aware, offline-ready assistant.

The proposal

Overview

We propose **ChatR**, a context-aware AI assistant built inside the R ecosystem that combines retrieval-augmented generation (RAG), session awareness, and tool invocation. Through a chat interface (R console, RStudio add-in, or VS Code), ChatR classifies queries, gathers context, retrieves documentation, executes code when needed, and returns cited answers. By integrating with R and supporting offline use, ChatR will reduce time spent on debugging and documentation search, improve reproducibility with authoritative sources, and make R more accessible for newcomers in statistics, bioinformatics, social sciences, etc.

Timeline

- **Months 1–2**: Design, infrastructure setup, initial knowledge base.
- **Months 3–5**: MVP delivery and early user testing.
- **Months 6–9**: Expanded tools, autonomous agent loop, evaluation.
- **Months 10–12**: Packaging, documentation, CRAN release, community outreach.

Failure modes & recovery

- *LLM unreliability*: Mitigated by constraining outputs to retrieved documentation.
- *Performance bottlenecks*: Use efficient vector stores, quantized models, caching.
- *Scope creep*: Prioritize core workflows; defer advanced features to future work.

Detail

Minimum Viable Product

By Month 3, ChatR will demonstrate end-to-end functionality with:

- R console chat interface.
- Query classification (docs/help, code, error fix).
- Knowledge base of base R + tidyverse docs (embeddings).
- Basic tools: CRAN search, safe code execution sandbox.
- Responses grounded in retrieved text with citations.

Architecture

ChatR will use a modular design with:

- **Interface:** R console first, then IDE add-ins (RStudio, Shiny).

- **Core & Retrieval:** Orchestrator with embedding search (FAISS or R alternatives).
- **Tools:** Docs/CRAN search, code execution, environment inspection.
- **LLM & Dialogue:** Online APIs or offline OSS models with multi-turn memory.
- **Integration:** MCP/LangChain for extensibility; plumber for local API.

Assumptions

- **User environment:** ChatR installable; offline mode with small models or fallback to API.
- **Tool safety:** Code restricted to R session with safeguards.

If any assumptions fail (e.g., offline models prove too resource-heavy), fallback options such as smaller KBs or API-based models will be provided.

External dependencies

FAISS (with Rcpp fallbacks) for retrieval, OpenAI or local OSS models for LLMs/embeddings, and MCP/LangChain with plumber for integration.

Project plan

Start-up phase

In the start-up phase we will set up GitHub infrastructure (repo, CI, license, reporting), define the architecture, test retrieval on R docs, and announce the project to the community—delivering a console prototype, initial knowledge base, and public roadmap.

Technical delivery

- **M1 – MVP (Month 3):** Minimal assistant with query classification, base R retrieval, basic answers, citations, and context retention.
- **M2 – Enhanced Tools (Months 4–5):** Expand KB to ~100 packages; add CRAN search, safe code execution, and environment introspection.
- **M3 – Autonomy & Error Handling (Months 6–7):** ReAct-style orchestration; sequential tool use; error-diagnosis module.
- **M4 – Evaluation (Month 8):** LLM-judge scoring, user feedback, latency optimization, improved citation display.
- **M5 – Deployment (Months 9–10):** CRAN packaging, MCP/LangChain integration, IDE add-in (e.g., RStudio gadget).
- **M6 – Release (Months 11–12):** Documentation, tutorials, blog posts; v1.0 release and public announcement.

Progress tracking: Bi-weekly mentor reviews and quarterly ISC updates.

Other aspects

Open source (MIT/GPL) with GitHub/CRAN distribution and active community engagement (announcements, blogs, updates, conferences, ISC reports) to ensure transparency and reproducibility.

Budget & funding plan

Total request: ~**$10,000**, allocated mainly to labor with milestone-based disbursement.

- **Developer (~90%)**: 20 hrs/week for 12 months; funds released in two tranches.(Hanxia Li)
- **Community (~10%)**: Tutorials, testing support, incidental costs; unused funds forfeited.

Milestone-based funding plan:

Milestone	Timeline	Work to be Done	Expected Outcome	Budget
MVP Completion	Month 3	Console prototype with retrieval & citation	Demo to ISC, early feedback	15%
Enhanced RAG & Tools	Months 4–5	Expanded KB + tool integrations	Advanced demo	20%
Agent Autonomy	Months 6–7	Multi-step reasoning, error handling	Usable assistant	20%
Evaluation & Tuning	Month 8	Evaluation harness + optimizations	Quality assurance	15%
Integration & Deployment	Months 9–10	CRAN packaging, IDE add-in	Public release candidate	15%
Documentation & Release	Months 11–12	Tutorials, blog posts, v1.0 release	Final report & CRAN submission	15%

This plan ensures that funds are directly tied to measurable progress and successful outcomes.

Success

Definition of done

- Release **ChatR v1.0** (CRAN/Universe) with context-aware assistance, online/offline modes, error diagnosis, broad KB coverage, documentation, and evidence of community adoption.

Measuring success

- Achieve 80% accuracy vs baseline, positive user feedback (4/5), 50 testers with external contributions, and case studies showing time savings and novice accessibility.

Future work

- Sustain through a community-led team, improved AI models, cross-language expansion, deeper IDE integration, community-driven KB, new features (multi-modal, learning mode), and educational impact studies.