Package: llmcoder (via r-universe)

November 4, 2026

Title LLM-Powered Code Generation, Error Fixing, and Chat for 'RStudio'

Version 1.2.0

Maintainer Shiyang Zheng <shiyang.zheng@nottingham.ac.uk>

Author Shiyang Zheng [aut, cre] (<https://orcid.org/0000-0003-0511-4683>)

Description An 'RStudio' addin that integrates large language model (LLM) assistance directly into the code-editing workflow. Features include: (1) generate R code from inline comments; (2) obtain LLM-assisted fixes for console errors; (3) insert plain-English explanations of selected code blocks; (4) a multi-turn Chat Panel with session-context awareness (loaded packages, global objects, source editor contents, console history). Supports 'OpenAI', 'Anthropic' (Claude), 'DeepSeek', 'Groq', 'Together AI', 'OpenRouter', 'Ollama' (fully local, no API key required), and any 'OpenAI'-compatible custom endpoint (e.g. 'LM Studio', 'vLLM', 'llama.cpp').

License MIT + file LICENSE

Depends R (>= 4.1.0)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

URL https://github.com/ShiyangZheng/llmcoder

BugReports https://github.com/ShiyangZheng/llmcoder/issues

Imports rstudioapi (>= 0.13), httr2 (>= 1.0.0), miniUI (>= 0.1.1), shiny (>= 1.7.0), stringi (>= 1.7.0), stringr (>= 1.5.0), rlang (>= 1.0.0), htmltools (>= 0.5.0), jsonlite (>= 1.8.0)

Suggests testthat (>= 3.0.0), withr (>= 2.5.0)

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev zlib1g-dev

Repository https://shiyangzheng.r-universe.dev

Date/Publication 2026-05-08 07:49:34 UTC

RemoteUrl https://github.com/shiyangzheng/llmcoder

RemoteRef HEAD

RemoteSha b0866db88050add87463723dae31f146f3f40b7e

Contents

addin_explain_code 2
addin_fix_console_error 3
addin_fix_selected_error 4
addin_generate_from_comment 5
addin_generate_with_preview 5
addin_settings 6
llmcoder_config 6
llmcoder_setup 7
ollama_list_models 9
session_context_prompt 10
session_context_report 11

Index **12**

`addin_explain_code` *Explain selected R code as inline comments*

Description

Select a block of R code in the editor, then trigger this addin (recommended shortcut: `Ctrl+Shift+E` / `Cmd+Shift+E`). An explanation is inserted as `#` comment lines immediately **above** the selected code block.

Usage

```
addin_explain_code()
```

Details

The LLM receives the selected code and is instructed to produce a concise, human-readable explanation — focusing on *what* the code does and *why*, not on basic R syntax. Every output line is prefixed with `#` so the explanation is valid R that can be left in the source file.

Value

Invisible `NULL` (called for side-effects).

See Also

`addin_generate_from_comment()`, `llmcoder_setup()`

addin_fix_console_error *Fix the last console error automatically*

Description

After running code that produces an error in the R console, trigger this addin (recommended shortcut: `Ctrl+Shift+F` / `Cmd+Shift+F`).

Usage

```
addin_fix_console_error()
```

Details

The addin attempts to recover the most recent error message using several strategies, in order of priority:

1. The `rlang` last-error store (`rlang::last_error()`), which captures errors thrown by **rlang**-aware packages and the tidyverse.
2. Base R's `.Last.error` binding (set whenever an unhandled condition reaches the top level).
3. The `.Last.error.trace` character vector written by some versions of **rlang**.

The complete source file currently open in the editor is also sent to the LLM as context. The LLM returns the entire corrected file, with changed lines annotated as `# FIX: <reason>`. A diff-style preview dialog lets you review and edit the fix before applying it.

Workflow

1. Run code — error appears in console.
2. Trigger this addin.
3. Review the fix in the preview dialog → click **Apply Fix**.

If no recent error is detected, a dialog explains the possible reasons and suggests using `addin_fix_selected_error()` instead.

Value

Invisible `NULL` (called for side-effects).

See Also

`addin_fix_selected_error()`, `llmcoder_setup()`

`addin_fix_selected_error` *Fix an error by selecting its text*

Description

Select the error message text in the editor (or paste it into a temporary comment), then trigger this addin. The addin pairs the selected text with the complete source file currently open in the editor and asks the LLM for a fix, displaying the result in a review dialog.

Usage

```
addin_fix_selected_error()
```

Details

This addin is the recommended fallback when `addin_fix_console_error()` does not detect an error automatically (e.g., because the error occurred inside a `tryCatch()` block or in a separate R process).

Workflow

1. Copy the error message from the console.
2. Paste it anywhere in the source file, or simply select it in the console output if your terminal supports that.
3. Select the error text in the editor.
4. Trigger this addin.
5. Review and apply the suggested fix.

Value

Invisible `NULL` (called for side-effects).

See Also

`addin_fix_console_error()`, `llmcoder_setup()`

addin_generate_from_comment

Generate R code from a comment (silent insert)

Description

Places the cursor on a line beginning with #, then triggers this addin (default shortcut: `Ctrl+Shift+G` on Windows/Linux, `Cmd+Shift+G` on macOS). The LLM reads the comment text and the surrounding code context, then inserts the generated R code on the line immediately below the comment.

Usage

```
addin_generate_from_comment()
```

Details

The addin extracts the text of the comment at the cursor position and up to `getOption("llmcoder.context_lines", 40L)` lines of preceding code as context. The provider, model, and API key are taken from options set by `llmcoder_setup()` or the **LLMcoder Settings** addin.

No dialog is shown; code is inserted immediately. Use `addin_generate_with_preview()` if you prefer to review the output first.

Value

Invisible `NULL` (called for side-effects).

See Also

`addin_generate_with_preview()`, `llmcoder_setup()`

addin_generate_with_preview

Generate R code with an editable preview dialog

Description

Same as `addin_generate_from_comment()` but opens a Shiny gadget so you can review and optionally edit the generated code before it is inserted into the editor. Recommended shortcut: `Ctrl+Shift+P` / `Cmd+Shift+P`.

Usage

```
addin_generate_with_preview()
```

Details

The preview dialog shows the generated code in an editable text area. Click **Insert** to place it in the editor, or close the dialog to discard the result.

Value

Invisible `NULL` (called for side-effects).

See Also

`addin_generate_from_comment()`, `llmcoder_setup()`

---

addin_settings *Open the LLMcoder settings dialog*

Description

Launches an interactive Shiny gadget that lets you configure the LLM provider, model, API key, Ollama URL (for local models), custom base URL (for LM Studio / vLLM / llama.cpp), and context-window size. Settings can optionally be persisted to `~/.Rprofile` so they survive R restarts.

Usage

```
addin_settings()
```

Value

Invisible `NULL` (called for side-effects).

See Also

`llmcoder_setup()`, `llmcoder_config()`

---

llmcoder_config *Show the current LLMcoder configuration*

Description

Returns (and prints) the active provider, model, API key (masked), context-lines setting, and any provider-specific URLs.

Usage

```
llmcoder_config()
```

Value

An object of class "llmcoder_config": a named list with elements provider, model, api_key, context_lines, ollama_url, and custom_url. The API key is masked for security. When printed, it displays in a human-readable table.

See Also

llmcoder_setup()

Examples

```
# Show current configuration (reads from option values)
llmcoder_config()

# Capture the config as a list for programmatic use
cfg <- llmcoder_config()
cfg$provider
cfg$model
```

---

llmcoder_setup *Configure LLMcoder for the current session*

Description

Sets the LLM provider, API key, model, and related options for the current R session. For **permanent** configuration that survives restarts, use **Addins > LLMcoder Settings**, which writes to ~/.Rprofile.

Usage

```
llmcoder_setup(
  provider = c("openai", "anthropic", "deepseek", "ollama", "groq", "together",
    "openrouter", "custom"),
  api_key = NULL,
  model = NULL,
  context_lines = 40L,
  ollama_url = "http://localhost:11434",
  custom_url = ""
)
```

Arguments

provider	Character. One of "openai", "anthropic", "deepseek", "ollama", "groq", "together", "openrouter", or "custom".
api_key	Character. Your API key. Not required when provider = "ollama".

model	Character. Model identifier. If NULL or "", a sensible default is chosen for the provider (see **Details**).
context_lines	Integer. Number of lines of code above the cursor that are sent as context to the LLM (default 40). Higher values improve suggestion quality but increase latency and token cost.
ollama_url	Character. Base URL of the Ollama server (default "http://localhost:11434"). Only used when provider = "ollama".
custom_url	Character. Base URL of a custom OpenAI-compatible server (e.g. "http://localhost:1234/v1" for LM Studio). Only used when provider = "custom".

Details

Provider defaults:

Provider	Default model	Notes
openai	gpt-4o-mini	Fast, cost-effective
anthropic	claude-sonnet-4-20250514	Strongest reasoning
deepseek	deepseek-chat	Very cheap, great code quality
ollama	llama3	No API key, fully local
groq	llama-3.3-70b-versatile	Extremely fast inference
together	meta-llama/Llama-3-70b-chat-hf	Large open-source model choice
openrouter	openai/gpt-4o-mini	Unified gateway for 100+ models
custom	"" (must specify)	Any OpenAI-compat endpoint

Value

Invisible NULL.

See Also

llmcoder_config(), addin_settings()

Examples

```
## Not run:
# OpenAI
llmcoder_setup("openai", api_key = Sys.getenv("OPENAI_API_KEY"))
llmcoder_setup("openai", api_key = Sys.getenv("OPENAI_API_KEY"), model = "gpt-4o")

# Anthropic Claude
llmcoder_setup("anthropic", api_key = Sys.getenv("ANTHROPIC_API_KEY"))

# DeepSeek (cheapest, excellent code quality)
llmcoder_setup("deepseek", api_key = Sys.getenv("DEEPSEEK_API_KEY"))

# Ollama — fully local, no API key needed
llmcoder_setup("ollama", model = "qwen2.5-coder:7b")
llmcoder_setup("ollama", model = "codellama:13b",
               ollama_url = "http://192.168.1.10:11434")  # remote server
```

```
# Groq — extremely fast inference on open models
llmcoder_setup("groq",
  api_key = Sys.getenv("GROQ_API_KEY"),
  model   = "llama-3.3-70b-versatile")

# Together AI — wide open-source model selection
llmcoder_setup("together",
  api_key = Sys.getenv("TOGETHER_API_KEY"),
  model   = "mistralai/Mixtral-8x7B-Instruct-v0.1")

# OpenRouter — unified gateway, supports 100+ models
llmcoder_setup("openrouter",
  api_key = Sys.getenv("OPENROUTER_API_KEY"),
  model   = "anthropic/claude-3.5-sonnet")

# LM Studio or any OpenAI-compatible local server
llmcoder_setup("custom",
  api_key    = "lm-studio",
  model      = "local-model",
  custom_url = "http://localhost:1234/v1")

# Reduce context window to save tokens
llmcoder_setup("openai",
  api_key       = Sys.getenv("OPENAI_API_KEY"),
  context_lines = 20L)

## End(Not run)
```

`ollama_list_models` *List models available on a running Ollama server*

Description

Queries `GET /api/tags` on the local Ollama REST API and returns the names of all installed models. Useful for populating the model selector in the Settings gadget.

Usage

```
ollama_list_models(
  base_url = getOption("llmcoder.ollama_url", "http://localhost:11434")
)
```

Arguments

`base_url`	Character. Ollama base URL. Defaults to the value of `getOption("llmcoder.ollama_url", "http://localhost:11434")`.

Details

Ollama must be running (`ollama serve`) before calling this function. Models are installed with `ollama pull <model>` from the terminal.

Value

Character vector of model tag names, or `NULL` if Ollama is not reachable.

Examples

```
## Not run:
ollama_list_models()
# [1] "llama3:latest"  "qwen2.5-coder:7b"  "mistral:latest"

## End(Not run)
```

---

session_context_prompt

Build a session-context system-prompt block

Description

Convenience wrapper around `session_context_report()` that wraps the report in a descriptive header so the LLM can distinguish it from user content.

Usage

```
session_context_prompt(...)
```

Arguments

`...`	Passed to `session_context_report()`.

Value

Character string suitable for prepending to a system prompt.

Examples

```
## Not run:
ctx_prompt <- session_context_prompt()

## End(Not run)
```

`session_context_report` *Capture a human-readable report of the current R session state*

Description

`session_context_report()` collects and formats the following information from the current R session:

- R version and operating system.
- Loaded add-on packages (non-base).
- Global environment objects grouped by class.
- Contents of the active source editor (via **rstudioapi**).
- Console command history (via **rstudioapi**; falls back to `~/.Rhistory`).

This report is primarily used internally to populate the system prompt sent to the LLM in the `addin_chat_panel()` gadget, so the model has full awareness of the analyst's working environment.

Usage

```
session_context_report(max_objs = 20L, max_hist = 30L, quiet = FALSE)
```

Arguments

`max_objs`	Maximum number of global objects to list per class group (default 20).
`max_hist`	Maximum number of console history lines to include (default 30).
`quiet`	If `TRUE`, suppress the `message()` emitted when RStudio API is unavailable (default `FALSE`).

Value

Character string. A multi-section report ready to embed in a system prompt.

Examples

```
## Not run:
report <- session_context_report()
cat(report)

## End(Not run)
```

Index

addin_chat_panel(), *11*
addin_explain_code, 2
addin_fix_console_error, 3
addin_fix_console_error(), *4*
addin_fix_selected_error, 4
addin_fix_selected_error(), *3*
addin_generate_from_comment, 5
addin_generate_from_comment(), *2*, *5*, *6*
addin_generate_with_preview, 5
addin_generate_with_preview(), *5*
addin_settings, 6
addin_settings(), *8*

llmcoder_config, 6
llmcoder_config(), *6*, *8*
llmcoder_setup, 7
llmcoder_setup(), *2–7*

ollama_list_models, 9

session_context_prompt, 10
session_context_report, 11
session_context_report(), *10*