Package: PacketLLM (via r-universe)

June 29, 2026

Title AI Assistant Gadget for 'RStudio'

Version 0.1.3

Description Provides an interactive 'RStudio' gadget for working with an AI assistant during package and script development. The gadget can use selected editor text, the active source file, package metadata, and uploaded files as context for code explanation, code generation, documentation, and review workflows. It offers model presets, assistant behavior settings, responsive code-focused output, and explicit copy, insert, and replace actions for the active source editor. API interactions via the 'httr' package are performed asynchronously using 'promises' and 'future' to avoid blocking the R console. The backend is configured via the OPENAI_API_KEY environment variable.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/AntoniCzolgowski/PacketLLM

BugReports https://github.com/AntoniCzolgowski/PacketLLM/issues

Imports future, httr, pdftools, promises, readtext, rstudioapi, shiny, shinyjs, stats, tools, utils

Depends R (>= 4.1.0)

Suggests htmltools, knitr, remotes, rmarkdown, testthat (>= 3.0.0), usethis

Config/testthat/edition 3

VignetteBuilder knitr

Language en-US

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Antoni Czolgowski [aut, cre]

Maintainer Antoni Czolgowski <antoni.czolgowski@gmail.com>

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

Date/Publication 2026-06-29 15:50:02 UTC

RemoteUrl https://github.com/cran/PacketLLM

RemoteRef HEAD

RemoteSha 2ba71ac034fe70732e491cc86eb19ae898b2be26

Contents

add_attachment_to_active_conversation

Add attachment to active conversation

Description

Add attachment to active conversation

Usage

```
add_attachment_to_active_conversation(name, content)
```

Arguments

name	File name.
content	File content as string.

Value

Logical.

add_message_to_active_history

Add a message to the active conversation

Description

Locks the model on the first assistant message. Sets title on the first user message.

Usage

```
add_message_to_active_history(role, content)
```

Arguments

role	'user'|'assistant'|'system'
content	Message content

Value

Result list (type + extra fields) or error list.

`add_user_message` *Add user message to the active conversation*

Description

Add user message to the active conversation

Usage

```
add_user_message(text)
```

Arguments

`text`	Single character string.

Value

Invisible NULL.

`available_model_presets` *Available model presets*

Description

Returns the model presets shown in the PacketLLM UI.

Usage

```
available_model_presets()
```

Value

A data frame with preset labels, model IDs, and descriptions.

`available_openai_models` *List of available AI models for selection in the UI*

Description

List of available AI models for selection in the UI

Usage

```
available_openai_models
```

`call_openai_chat` *Call the configured AI model*

Description

Sends messages to the Responses API and returns assistant text.

Usage

```
call_openai_chat(
  messages,
  model,
  reasoning_effort = "medium",
  verbosity = "low",
  max_output_tokens = NA_integer_
)
```

Arguments

`messages`	List of messages, each with `role` and `content`.
`model`	Model ID to use.
`reasoning_effort`	Reasoning effort: `low`, `medium`, `high`, or `xhigh`.
`verbosity`	Output verbosity: `low`, `medium`, or `high`.
`max_output_tokens`	Optional maximum output tokens.

Value

Character string with assistant reply, or NULL on unexpected response.

Examples

```
## Not run:
  msgs <- list(list(role = "user", content = "What does httr do?"))
  call_openai_chat(messages = msgs, model = "gpt-5.4-mini")

## End(Not run)
```

`capture_rstudio_context` *Capture RStudio editor and project context*

Description

Captures lightweight context from the active RStudio editor and project. The function returns a graceful empty context outside RStudio.

Usage

```
capture_rstudio_context(mode = "auto", max_chars = 6000)
```

Arguments

`mode`	One of `auto`, `focused`, `project`, or `none`.
`max_chars`	Maximum characters to keep from source content.

Value

A structured list describing the captured context.

`check_api_key` *Check API Key*

Description

Verifies that `OPENAI_API_KEY` is set. Stops if missing.

Usage

```
check_api_key()
```

Value

Invisible TRUE if set.

Examples

```
## Not run:
  check_api_key()

## End(Not run)
```

create_new_conversation *Create a new conversation*

Description

Create a new conversation

Usage

```
create_new_conversation(
  activate = FALSE,
  add_initial_settings = TRUE,
  title = NULL
)
```

Arguments

activate	Logical. Activate immediately?
add_initial_settings	Logical. Add default model and system message?
title	Optional title; if NULL, a time-based title is used.

Value

Character: conversation ID.

delete_conversation *Delete a conversation*

Description

Delete a conversation

Usage

```
delete_conversation(id)
```

Arguments

id	Conversation ID.

Value

TRUE if deleted, FALSE otherwise.

`get_active_chat_history` *Get active chat history*

Description

Get active chat history

Usage

```
get_active_chat_history()
```

Value

List of messages (possibly empty).

`get_active_conversation` *Get active conversation object*

Description

Get active conversation object

Usage

```
get_active_conversation()
```

Value

List or NULL.

`get_active_conversation_attachments` *Get attachments for active conversation*

Description

Get attachments for active conversation

Usage

```
get_active_conversation_attachments()
```

Value

List (possibly empty).

get_active_conversation_id

Get active conversation ID

Description

Get active conversation ID

Usage

```
get_active_conversation_id()
```

Value

Character or NULL.

get_all_conversation_ids

Get all conversation IDs

Description

Get all conversation IDs

Usage

```
get_all_conversation_ids()
```

Value

Character vector.

get_assistant_response

Get assistant response for the active conversation

Description

Prepares messages and calls the API. Adds the reply to history.

Usage

```
get_assistant_response()
```

Value

Character with assistant reply or error message.

get_conversation_attachments *Get attachments by ID*

Description

Get attachments by ID

Usage

```
get_conversation_attachments(id)
```

Arguments

id	Conversation ID.

Value

List or NULL.

get_conversation_data *Get conversation data by ID*

Description

Get conversation data by ID

Usage

```
get_conversation_data(id)
```

Arguments

id	Conversation ID.

Value

List or NULL.

get_conversation_history *Get conversation history by ID*

Description

Get conversation history by ID

Usage

```
get_conversation_history(id)
```

Arguments

id	Conversation ID.

Value

List or NULL.

get_conversation_model *Get model for conversation*

Description

Get model for conversation

Usage

```
get_conversation_model(id)
```

Arguments

id	Conversation ID.

Value

Character or NULL.

get_conversation_title *Get conversation title by ID*

Description

Get conversation title by ID

Usage

```
get_conversation_title(id)
```

Arguments

id	Conversation ID.

Value

Character or NULL.

initialize_history_manager *Initialize the history manager*

Description

Initializes state and creates or restores a conversation, then activates it.

Usage

```
initialize_history_manager(persist = FALSE)
```

Arguments

persist	Logical. Restore and save local gadget history?

Value

Character: ID of the created conversation.

is_conversation_started *Has the conversation started (model locked)?*

Description

Has the conversation started (model locked)?

Usage

```
is_conversation_started(id)
```

Arguments

id	Conversation ID.

Value

Logical.

packetllm_addin *Launch PacketLLM from the RStudio Addins menu*

Description

Launch PacketLLM from the RStudio Addins menu

Usage

```
packetllm_addin()
```

Value

Value passed to `run_llm_chat_app()`.

parse_pages *Parse page range*

Description

This function processes a character string specifying a page range (e.g., "1-3,5") and returns a numeric vector containing the individual page numbers, sorted and unique.

Usage

```
parse_pages(pages_str)
```

Arguments

pages_str	Character string specifying pages, e.g., "1-3,5".

Value

A numeric vector containing the unique page numbers specified in the input string, sorted in ascending order. Returns an empty integer vector if the input string is empty or contains only whitespace. Stops with an error if the input `pages_str` is not a single character string or if the format within the string is invalid (e.g., non-numeric parts, invalid ranges).

Examples

```
# Example 1: Simple range and single page
page_string1 <- "1-3, 5"
parsed_pages1 <- parse_pages(page_string1)
print(parsed_pages1) # Output: [1] 1 2 3 5

# Example 2: Multiple ranges and single pages, with spaces and duplicates
page_string2 <- " 2, 4-6, 9 , 11-12, 5 "
parsed_pages2 <- parse_pages(page_string2)
print(parsed_pages2) # Output: [1] 2 4 5 6 9 11 12 (sorted, unique)

# Example 3: Single number
page_string3 <- "10"
parsed_pages3 <- parse_pages(page_string3)
print(parsed_pages3) # Output: [1] 10

# Example 4: Empty string input
page_string_empty <- ""
parsed_pages_empty <- parse_pages(page_string_empty)
print(parsed_pages_empty) # Output: integer(0)

# Example 5: Invalid input (non-numeric) - demonstrates error handling
page_string_invalid <- "1-3, five"
## Not run:
# This will stop with an error message about "five"
```

```
tryCatch(parse_pages(page_string_invalid), error = function(e) print(e$message))

## End(Not run)

# Example 6: Invalid range format (missing end) - demonstrates error handling
page_string_invalid_range <- "1-"
## Not run:
# This will stop with an error message about invalid range format
tryCatch(parse_pages(page_string_invalid_range), error = function(e) print(e$message))

## End(Not run)

# Example 7: Invalid range format (start > end) - demonstrates error handling
page_string_invalid_order <- "5-3"
## Not run:
# This will stop with an error message about invalid range values
tryCatch(parse_pages(page_string_invalid_order), error = function(e) print(e$message))

## End(Not run)
```

`read_file_content` *Read file content*

Description

This function reads the content of a file with the extension .R, .pdf, .docx, .txt, or .csv and returns it as a single character string. For PDF files, if the `pages` parameter is provided, only the selected pages will be read. For CSV files, the data is converted into a compact, model-friendly Markdown representation: a short header with the dimensions, a list of columns with their inferred types, and the rows rendered as a Markdown table. The field delimiter is detected automatically (comma, semicolon, tab, or pipe), and large files are truncated to the first 1000 rows with an explicit note.

Usage

```
read_file_content(file_path, pages = NULL)
```

Arguments

`file_path`	Character string. Path to the file.
`pages`	Optional. A numeric vector specifying which pages (for PDF) should be read.

Value

A character string containing the file content, with pages separated by double newlines for PDF files and CSV data rendered as a Markdown table. Stops with an error if the file does not exist, the format is unsupported, or required packages (`pdftools` for PDF, `readtext` for DOCX) are not installed or if `pages` is not numeric when provided.

Examples

```
# --- Example for reading an R file ---
# Create a temporary R file
temp_r_file <- tempfile(fileext = ".R")
writeLines(c("x <- 1", "print(x + 1)"), temp_r_file)

# Read the content
r_content <- tryCatch(read_file_content(temp_r_file), error = function(e) e$message)
print(r_content)

# Clean up the temporary file
unlink(temp_r_file)

# --- Example for reading a TXT file ---
temp_txt_file <- tempfile(fileext = ".txt")
writeLines(c("Line one.", "Second line."), temp_txt_file)
txt_content <- tryCatch(read_file_content(temp_txt_file), error = function(e) e$message)
print(txt_content)
unlink(temp_txt_file)

# --- Example for reading a CSV file ---
temp_csv_file <- tempfile(fileext = ".csv")
write.csv(head(mtcars, 3), temp_csv_file, row.names = FALSE)
csv_content <- tryCatch(read_file_content(temp_csv_file), error = function(e) e$message)
cat(csv_content)
unlink(temp_csv_file)

# --- Example for PDF (requires pdftools, only run if installed) ---
## Not run:
# This part requires the 'pdftools' package and a valid PDF file.
# Provide a path to an actual PDF file to test this functionality.
# Replace "path/to/your/sample.pdf" with a real path.

pdf_file_path <- "path/to/your/sample.pdf"

# Check if pdftools is installed and the file exists
if (requireNamespace("pdftools", quietly = TRUE) && file.exists(pdf_file_path)) {

  # Example: Read all pages
  pdf_content_all <- tryCatch(
    read_file_content(pdf_file_path),
    error = function(e) paste("Error reading all pages:", e$message)
  )
  # print(substr(pdf_content_all, 1, 100)) # Print first 100 chars

  # Example: Read only page 1
  pdf_content_page1 <- tryCatch(
    read_file_content(pdf_file_path, pages = 1),
    error = function(e) paste("Error reading page 1:", e$message)
  )
  # print(pdf_content_page1)
```

```
} else if (!requireNamespace("pdftools", quietly = TRUE)) {
  message("Skipping PDF example: 'pdftools' package not installed.")
} else {
  message("Skipping PDF example: File not found at '", pdf_file_path, "'")
}

## End(Not run)
# Note: Reading DOCX files is also supported if the 'readtext' package
# is installed, but a simple runnable example is difficult to create
# without including a sample file or complex setup.
```

`reset_history_manager` *Reset the history manager*

Description

Reset the history manager

Usage

```
reset_history_manager(clear_persistent = FALSE)
```

Arguments

`clear_persistent`
: Logical. Delete saved local gadget history too?

Value

Invisible NULL.

`run_llm_chat_app` *Run the PacketLLM gadget*

Description

Launches PacketLLM as an RStudio/Shiny gadget. The gadget can use RStudio editor context when available, while still running outside RStudio with reduced functionality.

Usage

```
run_llm_chat_app()
```

Value

Value passed to shiny::stopApp() (typically NULL).

Examples

```
## Not run:
run_llm_chat_app()

## End(Not run)
```

`set_active_conversation` *Set the active conversation*

Description

Set the active conversation

Usage

```
set_active_conversation(id)
```

Arguments

`id`	Conversation ID or NULL.

Value

Invisible NULL.

`set_conversation_model` *Set model for conversation (if not started)*

Description

Set model for conversation (if not started)

Usage

```
set_conversation_model(id, model_name)
```

Arguments

`id`	Conversation ID.
`model_name`	Model name (must be in available_openai_models).

Value

Logical.

`set_conversation_system_message` *Set system message for conversation*

Description

Set system message for conversation

Usage

```
set_conversation_system_message(id, message)
```

Arguments

`id`	Conversation ID.
`message`	Single string system message.

Value

Logical.

Index