Package 'chatAI4R'

May 8, 2026

Type Package

Title Chat-Based Interactive Artificial Intelligence for R

Version 1.3.1

Date 2026-1-8

Maintainer Satoshi Kume <satoshi.kume.1984@gmail.com>

Description The Large Language Model (LLM) represents a groundbreaking advancement
in data science and programming, and also allows us to extend the world of R.
A seamless interface for integrating the 'OpenAI' Web APIs into R is provided in this package.
This package leverages LLM-
based AI techniques, enabling efficient knowledge discovery and data analysis.
The previous functions such as seamless translation and image generation have been moved
to other packages 'deepRstudio' and 'stableDiffusion4R'.

Depends R (>= 4.2.0)

Imports httr, jsonlite, assertthat, clipr, crayon, rstudioapi, future,
igraph, deepRstudio, curl, base64enc, glue, utils

Suggests testthat, knitr

License Artistic-2.0

URL https://kumes.github.io/chatAI4R/,
https://github.com/kumeS/chatAI4R

BugReports https://github.com/kumeS/chatAI4R/issues

RoxygenNote 7.3.3

Encoding UTF-8

NeedsCompilation no

Author Satoshi Kume [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7481-2843>)

Repository CRAN

Date/Publication 2026-01-10 16:30:02 UTC

Contents

`addCommentCode` *Add Comments to R Code*

Description

This function adds comments to R code without modifying the input R code. It can either take the selected code from RStudio or read from the clipboard.

Usage

```
addCommentCode(
  Model = "gpt-4o-mini",
  language = "English",
  SelectedCode = TRUE,
  provider = "auto"
)
```

Arguments

`Model`	A character string specifying the model to be used. Default is "gpt-4o-mini" for OpenAI or "gemini-2.0-flash" for Gemini.
`language`	A character string specifying the language for the comments. Default is "English".
`SelectedCode`	A logical value indicating whether to use the selected code in RStudio. Default is TRUE.
`provider`	A character string specifying the API provider. Options: "auto" (default), "openai", "gemini". When "auto", automatically detects available API keys with priority: OpenAI → Gemini.

Details

Add Comments to R Code

Value

A message indicating completion if ‘SelectedCode‘ is TRUE, otherwise the commented code is copied to the clipboard.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Option 1: Auto-detect available API (OpenAI or Gemini)
# Select some text in RStudio and then run the rstudio addins
addCommentCode(Model = "gpt-4o-mini", language = "English", SelectedCode = TRUE)

# Option 2: Explicitly use OpenAI
addCommentCode(Model = "gpt-4o-mini", provider = "openai", SelectedCode = TRUE)

# Option 3: Use Gemini API
addCommentCode(Model = "gemini-2.0-flash", provider = "gemini", SelectedCode = TRUE)

# Option 4: Copy text to clipboard and execute
addCommentCode(Model = "gpt-4o-mini", language = "English", SelectedCode = FALSE)

## End(Not run)
```

---

addRoxygenDescription *Add Roxygen Description to R Function*

Description

This function adds a Roxygen description to an R function using the GPT-4 model. It can either take the selected code from RStudio or read from the clipboard.

Usage

```
addRoxygenDescription(
  Model = "gpt-4o-mini",
  SelectedCode = TRUE,
  verbose = TRUE
)
```

Arguments

Model	A character string specifying the GPT model to be used. Default is "gpt-4o-mini".
SelectedCode	A logical value indicating whether to use the selected code in RStudio. Default is TRUE.
verbose	Logical flag to indicate whether to display the generated text. Default is TRUE.

Details

Add Roxygen Description to R Function

Value

A message indicating completion if 'SelectedCode' is TRUE, otherwise the Roxygen-annotated code is copied to the clipboard.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Option 1
# Select some text in RStudio and then run the rstudio addins
# Option 2
# Copy the text into your clipboard then execute
addRoxygenDescription(Model = "gpt-4o-mini", SelectedCode = FALSE)

## End(Not run)
```

---

autocreateFunction4R *autocreateFunction4R (development / experimental)*

Description

This function generates an R function based on a given description, proposes improvements, and then generates an improved version of the function. It uses an AI model (OpenAI GPT) with robust prompts to ensure clean R code output. This is an experimental function.

Usage

```
autocreateFunction4R(
  Func_description,
  packages = "base",
  Model = "gpt-5-nano",
  temperature = 1,
  View = TRUE,
  roxygen = TRUE,
  api_key = Sys.getenv("OPENAI_API_KEY"),
  verbose = TRUE
)
```

Arguments

Func_description	A character string that describes the function to be generated.
packages	A character string that specifies the packages to be used in the function. Default is "base".

Model	A character string that specifies the AI model to use. Default is "gpt-5-nano".
temperature	A numeric value that controls the randomness of the AI model's output. Higher values (e.g., 1.0) make output more random, lower values (e.g., 0.3) make it more focused and deterministic. Default is 1. IMPORTANT: When using "gpt-5-nano", temperature MUST be set to 1.
View	A logical that indicates whether to view the intermediate steps. Default is TRUE.
roxygen	A logical that indicates whether to include roxygen comments in the generated function. Default is TRUE.
api_key	A character string that represents the API key for the AI model being used. Default is the "OPENAI_API_KEY" environment variable.
verbose	A logical flag to print the message Default is TRUE.

Details

Generate and Improve R Functions (experimental)

Value

The function returns a character string that represents the generated and improved R function.

Author(s)

Satoshi Kume

Examples

```
## Not run:
  # Example 1: Basic usage with default settings
  Sys.setenv(OPENAI_API_KEY = "<APIKEY>")
 autocreateFunction4R(Func_description = "identify 2*n+3 sequence",  Model = "gpt-5-mini")

  # Example 2: Generate function using specific packages
  autocreateFunction4R(
    Func_description = "create a data frame with random numbers",
    Model = "gpt-5-mini", packages = "dplyr, tidyr"
  )

## End(Not run)
```

---

chat4R *Chat4R: Interact with gpt-4o-mini (default) using OpenAI API*

Description

This function uses the OpenAI API to interact with the gpt-4o-mini model (default) and generates responses based on user input. In this function, currently, "gpt-4o-mini" (default), "gpt-4o", "gpt-4", and "gpt-4-turbo" can be selected as OpenAI's LLM model.

Usage

```
chat4R(
  content,
  Model = "gpt-4o-mini",
  temperature = 1,
  simple = TRUE,
  fromJSON_parsed = FALSE,
  check = FALSE,
  api_key = Sys.getenv("OPENAI_API_KEY")
)
```

Arguments

content	A string containing the user's input message.
Model	A string specifying the GPT model to use (default: "gpt-4o-mini").
temperature	A numeric value controlling the randomness of the model's output (default: 1).
simple	Logical, if TRUE, only the content of the model's message will be returned.
fromJSON_parsed	Logical, if TRUE, content will be parsed from JSON.
check	Logical, if TRUE, prints detailed error information (message, type, param, code) if the API response includes an error. If there is no error, "No error" is printed.
api_key	A string containing the user's OpenAI API key. Defaults to the value of the environment variable "OPENAI_API_KEY".

Details

Chat4R Function

Value

A data frame or list containing the response from the GPT model, depending on arguments.

Author(s)

Satoshi Kume

Examples

```
## Not run:
Sys.setenv(OPENAI_API_KEY = "Your API key")
chat4R(content = "What is the capital of France?", check = TRUE)


## End(Not run)
```

chat4Rv2 *chat4Rv2: Interact with gpt-4o-mini (default) using OpenAI API*

Description

This function uses the OpenAI API to interact with the gpt-4o-mini model (default) and generates responses based on user input. In this function, currently, "gpt-4o-mini" (default), "gpt-4o", "gpt-4", and "gpt-4-turbo" can be selected as OpenAI's LLM model.

Usage

```
chat4Rv2(
  content,
  Model = "gpt-4o-mini",
  temperature = 1,
  max_tokens = 50,
  simple = TRUE,
  fromJSON_parsed = FALSE,
  system_prompt = "",
  api_key = Sys.getenv("OPENAI_API_KEY")
)
```

Arguments

content	A string containing the user's input message.
Model	A string specifying the GPT model to use (default: "gpt-4o-mini"). The function automatically handles parameter compatibility for newer models (o3, o1, gpt-4o series) that require max_completion_tokens instead of max_tokens.
temperature	A numeric value controlling the randomness of the model's output (default: 1).
max_tokens	A numeric value specifying the maximum number of tokens to generate (default is 50).
simple	Logical, if TRUE, only the content of the model's message will be returned.
fromJSON_parsed	Logical, if TRUE, content will be parsed from JSON.
system_prompt	A string containing the system message to set the context. If provided, it will be added as the first message in the conversation. Default is an empty string.
api_key	A string containing the user's OpenAI API key. Defaults to the value of the environment variable "OPENAI_API_KEY".

Details

chat4Rv2 Function

Value

A data frame containing the response from the GPT model.

Author(s)

Satoshi Kume

Examples

```
## Not run:
Sys.setenv(OPENAI_API_KEY = "Your API key")
# Using chat4Rv2 without system_prompt (default behavior)
response <- chat4Rv2(content = "What is the capital of France?")
response

# Using chat4Rv2 with a system_prompt provided
response <- chat4Rv2(content = "What is the capital of France?",
                     system_prompt = "You are a helpful assistant.")
response

## End(Not run)
```

---

chat4R_history *chat4R_history: Use chat history for interacting with GPT.*

Description

This function use the chat history with the the specified GPT model, and chat the AI model.

Usage

```
chat4R_history(
  history,
  api_key = Sys.getenv("OPENAI_API_KEY"),
  Model = "gpt-4o-mini",
  temperature = 1
)
```

Arguments

history	A list of message objects. Each object should have a 'role' that can be 'system', 'user', or 'assistant', and 'content' which is the content of the message from that role.
api_key	A string. Input your OpenAI API key. Defaults to the value of the environment variable "OPENAI_API_KEY".
Model	A string. The model to use for the chat completion. Default is "gpt-4o-mini".
temperature	The temperature to use for the chat completion. Default is 1.

Details

Chat History for R

Value

A data frame containing the parsed response from the Web API server.

Author(s)

Satoshi Kume

Examples

```
## Not run:
Sys.setenv(OPENAI_API_KEY = "Your API key")

history <- list(list('role' = 'system', 'content' = 'You are a helpful assistant.'),
                list('role' = 'user', 'content' = 'Who won the world series in 2020?'))

chat4R_history(history)

## End(Not run)
```

`chat4R_streaming` *chat4R_streaming: Interact with GPT-4o (default) with streaming using OpenAI API and set system context*

Description

This function uses the OpenAI API to interact with the GPT-4o model (default) and generates responses based on user input with streaming data back to R. In this function, currently, "gpt-4o-mini", "gpt-4o", and "gpt-4-turbo" can be selected as OpenAI's LLM model. Additionally, a system message can be provided to set the context.

Usage

```
chat4R_streaming(
  content,
  Model = "gpt-4o-mini",
  temperature = 1,
  system_set = "",
  api_key = Sys.getenv("OPENAI_API_KEY")
)
```

Arguments

`content`	A string containing the user's input message.
`Model`	A string specifying the GPT model to use (default: "gpt-4o-mini").
`temperature`	A numeric value controlling the randomness of the model's output (default: 1).

system_set	A string containing the system message to set the context. If provided, it will be added as the first message in the conversation. Default is an empty string.
api_key	A string containing the user's OpenAI API key. Defaults to the value of the environment variable "OPENAI_API_KEY".

Details

Chat4R Function with Streaming and System Context

Value

A data frame containing the response from the GPT model (streamed to the console).

Author(s)

Satoshi Kume

Examples

```
## Not run:
  Sys.setenv(OPENAI_API_KEY = "Your API key")

  # Without system_set
  chat4R_streaming(content = "What is the capital of France?")

  # With system_set provided
  chat4R_streaming(
    content = "What is the capital of France?",
    system_set = "You are a helpful assistant."
  )

## End(Not run)
```

---

checkErrorDet *Check Error Details*

Description

A function to analyze and provide guidance on how to fix an error message copied from the R console.

Usage

```
checkErrorDet(
  Summary_nch = 100,
  Model = "gpt-4o-mini",
  language = "English",
  verbose = TRUE,
  SlowTone = FALSE
)
```

Arguments

Summary_nch	An integer specifying the maximum number of characters for the summary.
Model	A string specifying the model to be used, default is "gpt-4o-mini". Currently, "gpt-4", "gpt-4-0314" and "gpt-4o-mini" can be selected as gpt-4 models. Execution with GPT-4 is recommended.
language	A string specifying the output language, default is "English".
verbose	A logical value to control the verbosity of the output, default is TRUE.
SlowTone	A logical value to control the printing speed of the output, default is FALSE.

Details

Check Error Details

This function provides a way to check error details in R. It takes an error message from the R console, executes the function, and shows how to fix the error in the specified language.

Value

The function prints the guidance on how to fix the error message.

Author(s)

Satoshi Kume

Examples

```
## Not run:
  # Copy the error message that you want to fix.
  checkErrorDet()
  checkErrorDet(language = "Japanese")

## End(Not run)
```

---

checkErrorDet_JP *Check Error Details in Japanese*

Description

A function to analyze and provide guidance on how to fix an error message copied from the R console.

Usage

```
checkErrorDet_JP(
  Summary_nch = 100,
  Model = "gpt-4o-mini",
  verbose = TRUE,
  SlowTone = FALSE
)
```

Arguments

Summary_nch	An integer specifying the maximum number of characters for the summary.
Model	A string specifying the model to be used, default is "gpt-4-0314". Currently, "gpt-4", "gpt-4-0314" and "gpt-4o-mini" can be selected as gpt-4 models. Execution with GPT-4 is recommended.
verbose	A logical value to control the verbosity of the output, default is TRUE.
SlowTone	A logical value to control the printing speed of the output, default is FALSE.

Details

Check Error Details in Japanese via RStudio API

This function provides a way to check error details in R. It reads the error message from the clipboard, executes the function, and shows how to fix the error in Japanese.

Value

The function prints the guidance on how to fix the error message in Japanese. If verbose is FALSE, it returns the guidance as a string.

Author(s)

Satoshi Kume

Examples

```
## Not run:
  # Analyzing error message from the clipboard
 checkErrorDet_JP(Summary_nch = 100, Model = "gpt-4o-mini", verbose = TRUE, SlowTone = FALSE)

## End(Not run)
```

---

completions4R *completions4R: Generate text using OpenAI completions API (Legacy - DEPRECATED)*

Description

[WARNING] DEPRECATED AND SCHEDULED FOR REMOVAL: This function uses the legacy OpenAI completions API which is being phased out by OpenAI. **This function will be removed in a future version.**

MIGRATION REQUIRED: Please migrate to chat4R() for all new implementations. The chat4R() function uses the modern OpenAI Chat Completions API and provides better performance, more features, and continued support.

Migration Example: “‘r # Old (deprecated): result <- completions4R("Your prompt here")

New (recommended): result <- chat4R("Your prompt here") text_content <- result$content “‘

This function sends a request to the OpenAI completions API to generate text based on the provided prompt and parameters.

Usage

```
completions4R(
  prompt,
  api_key = Sys.getenv("OPENAI_API_KEY"),
  Model = "gpt-3.5-turbo-instruct",
  max_tokens = 50,
  temperature = 1,
  simple = TRUE
)
```

Arguments

prompt	A string. The initial text that the model responds to.
api_key	A string. The API key for OpenAI. Defaults to the value of the environment variable "OPENAI_API_KEY".
Model	A string. The model ID to use, such as "davinci-002" or "gpt-3.5-turbo-instruct".
max_tokens	Integer. The maximum number of tokens to generate.
temperature	A numeric value to control the randomness of the generated text. A value close to 0 produces more deterministic output, while a higher value (up to 2) produces more random output.
simple	If TRUE, returns the generated text without newline characters. If FALSE, returns the full response from the API.

Value

The generated text or the full response from the API, depending on the value of ‘simple‘.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# DEPRECATED: Use chat4R() instead for new code
Sys.setenv(OPENAI_API_KEY = "Your API key")

prompt <- "Translate the following English text to French: 'Hello, world!'"

completions4R(prompt)

## End(Not run)
```

conversation4R *Conversation Interface for R*

Description

Interface to communicate with OpenAI's models using R, maintaining a conversation history and allowing for initialization of a new conversation.

Usage

```
conversation4R(
  message,
  api_key = Sys.getenv("OPENAI_API_KEY"),
  system_set = "",
  ConversationBufferWindowMemory_k = 2,
  Model = "gpt-4o-mini",
  language = "English",
  initialization = FALSE,
  verbose = TRUE
)
```

Arguments

message	A string containing the message to be sent to the model.
api_key	A string containing the OpenAI API key. Default is retrieved from the system environment variable "OPENAI_API_KEY".
system_set	A string containing the system_set for the conversation. Default is an empty string.
ConversationBufferWindowMemory_k	An integer representing the conversation buffer window memory. Default is 2.
Model	A string representing the model to be used. Default is "gpt-4o-mini".
language	A string representing the language to be used in the conversation. Default is "English".
initialization	A logical flag to initialize a new conversation. Default is FALSE.
verbose	A logical flag to print the conversation. Default is TRUE.

Details

Conversation Interface for R with OpenAI

This function provides an interface to communicate with OpenAI's models using R. It maintains a conversation history and allows for initialization of a new conversation.

Value

Prints the conversation if verbose is TRUE. No return value.

Author(s)

Satoshi Kume

Examples

```
## Not run:
conversation4R(message = "Hello, OpenAI!",
               api_key = "your_api_key_here",
               language = "English",
               initialization = TRUE)

## End(Not run)
```

`convertBullet2Sentence` *convertBullet2Sentence*

Description

Convert bullet points to sentences using OpenAI GPT model.

Usage

```
convertBullet2Sentence(
  Model = "gpt-4o-mini",
  temperature = 1,
  verbose = TRUE,
  SpeakJA = FALSE,
  SelectedCode = TRUE
)
```

Arguments

`Model`	The OpenAI GPT model to use for text generation. Default is "gpt-4o-mini".
`temperature`	The temperature parameter for text generation. Default is 1.
`verbose`	Logical flag to indicate whether to display progress. Default is TRUE.
`SpeakJA`	Logical flag to indicate whether to use Japanese speech output. Default is FALSE.
`SelectedCode`	Logical flag to indicate whether to use selected text in RStudio. Default is TRUE.

Details

Convert Bullet Points to Sentences

This function takes bullet points as input and converts them into sentences. The function uses the OpenAI GPT model for text generation to assist in the conversion. The function can either take the selected text from the RStudio environment or from the clipboard.

Value

Inserts the converted sentences into the RStudio editor if ‘SelectedCode‘ is TRUE, otherwise writes to clipboard.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Option 1
# Select some text in RStudio and then run the rstudio addins
# Option 2
# Copy the text into your clipboard then execute
convertBullet2Sentence(Model = "gpt-4o-mini", SelectedCode = FALSE)

## End(Not run)
```

---

convertRscript2Function *convertRscript2Function*

Description

Convert selected R script to an R function using LLM.

Usage

```
convertRscript2Function(
  Model = "gpt-4o-mini",
  SelectedCode = TRUE,
  verbose = TRUE
)
```

Arguments

Model	The OpenAI GPT model to use for text generation. Default is "gpt-4o-mini".
SelectedCode	Logical flag to indicate whether to use selected code in RStudio. Default is TRUE.
verbose	A logical value indicating whether to print the result to the console, default is TRUE.

Details

Convert Selected R Script to R Function

This function takes a selected portion of an R script and converts it into an R function. The function uses the OpenAI GPT model for text generation to assist in the conversion. The function can either take the selected code from the RStudio environment or from the clipboard.

Value

Inserts the converted function into the RStudio editor if ‘SelectedCode‘ is TRUE, otherwise writes to clipboard.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Option 1
# Select some text in RStudio and then run the rstudio addins
# Option 2
# Copy the text into your clipboard then execute
convertRscript2Function(Model = "gpt-4o-mini", SelectedCode = F)

## End(Not run)
```

`convertScientificLiterature` *convertScientificLiterature*

Description

Convert input text into scientific literature.

Usage

```
convertScientificLiterature(Model = "gpt-4o-mini", SelectedCode = TRUE)
```

Arguments

`Model`	The OpenAI GPT model to use for text generation. Default is "gpt-4o-mini".
`SelectedCode`	Logical flag to indicate whether to read the input from RStudio's active document. Default is TRUE.

Details

Convert to Scientific Literature

This function assists in converting the input text into scientific literature. It uses the OpenAI GPT model for text generation to assist in the conversion process. The function reads the input either from the RStudio active document or the clipboard.

Value

Inserts the converted text into the RStudio active document if SelectedCode is TRUE, otherwise writes to the clipboard.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Option 1
# Select some text in RStudio and then run the rstudio addins
# Option 2
# Copy the text into your clipboard then execute
convertScientificLiterature(SelectedCode = FALSE)

## End(Not run)
```

---

createEBAYdes *Create eBay Product Description*

Description

This function generates a product description for eBay listings in English. It uses the GPT-4 model for text generation and can take input from either RStudio or the clipboard.

Usage

```
createEBAYdes(
  Model = "gpt-5-nano",
  SelectedCode = TRUE,
  verbose = TRUE,
  SlowTone = FALSE
)
```

Arguments

Model	The GPT-4 model to use for text generation. Default is "gpt-5-nano".
SelectedCode	Whether to get the input from the selected code in RStudio. Default is TRUE.
verbose	Whether to display progress information. Default is TRUE.
SlowTone	Whether to print the output slowly. Default is FALSE.

Details

Create eBay Product Description

Value

The generated eBay product description.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Option 1
# Select some text in RStudio and then run the rstudio addins

# Option 2
# Copy the text into your clipboard then execute
createEBAYdes(Model = "gpt-4o-mini", SelectedCode = FALSE, verbose = TRUE)

# Option 3
# clipr the text into your clipboard then execute
clipr::write_clip("A new A100 GPU")
createEBAYdes(Model = "gpt-4o-mini", SelectedCode = FALSE, verbose = TRUE)

## End(Not run)
```

---

createImagePrompt_v1 *Create Image Prompt version 1*

Description

This function creates a prompt for generating an image from text using an AI model. This is an experimental function.

Usage

```
createImagePrompt_v1(content, Model = "gpt-4o-mini", len = 200)
```

Arguments

content	A character string describing the image to be generated. If not provided, the function will throw a warning and stop.
Model	A character string specifying the AI model to be used for text generation.
len	Integer specifying the maximum length of the text input.

Details

Create Image Prompt version 1

Value

A character string that serves as the prompt for generating an image.

Author(s)

Satoshi Kume

Examples

```
## Not run:
createImagePrompt_v1(content = "A Japanese girl animation with blonde hair.")

## End(Not run)
```

---

createImagePrompt_v2 *Generate Image Prompts version 2*

Description

Generates optimal prompts for creating images using the OpenAI API. Given a base prompt, negative elements, and style guidance, it generates three optimized prompts for image creation. This is an experimental function.

Usage

```
createImagePrompt_v2(
  Base_prompt = "",
  removed_from_image = "",
  style_guidance = "N/A",
  Model = "gpt-4o-mini",
  len = 1000
)
```

Arguments

Base_prompt	A string. The main description of the image you want to generate (e.g., "A serene mountain landscape").
removed_from_image	A string. Elements or attributes to avoid or exclude from the image (e.g., "people, buildings, text"). Use empty string if not applicable.
style_guidance	A string. Style or quality guidance for the image generation (e.g., "photorealistic", "artistic", "detailed", "minimalist"). Default is "N/A".
Model	A string. This is the model used for generating the prompts. Default is "gpt-4o-mini".
len	An integer. This is the maximum length of the generated prompts. Must be between 1 and 1000. Default is 1000.

Details

Generate Image Prompts version 2

Value

A vector of strings. Each string in the vector is a generated prompt (typically 3 prompts).

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Example 1: Landscape without specific exclusions
base_prompt <- "A peaceful mountain lake surrounded by pine trees"
removed_from_image <- ""
style_guidance <- "photorealistic, high detail"

res <- createImagePrompt_v2(Base_prompt = base_prompt,
                            removed_from_image = removed_from_image,
                            style_guidance = style_guidance,
                            len = 200)
print(res)

# Example 2: Portrait with specific exclusions
base_prompt <- "A portrait of a wise elderly person"
removed_from_image <- "hat, glasses, jewelry"
style_guidance <- "oil painting style"

res <- createImagePrompt_v2(Base_prompt = base_prompt,
                            removed_from_image = removed_from_image,
                            style_guidance = style_guidance)
print(res)

## End(Not run)
```

---

createRcode *Create R Code from Clipboard Content and Output into the R Console*

Description

Reads text from the clipboard and generates R code based on the given input, printing the result to the R console.

Usage

```
createRcode(
  Summary_nch = 100,
  Model = "gpt-4o-mini",
  SelectedCode = TRUE,
  verbose = TRUE,
  SlowTone = FALSE
)
```

Arguments

Summary_nch	The maximum number of characters for the summary.
Model	The model to be used for code generation, default is "gpt-4o-mini".
SelectedCode	A logical flag to indicate whether to read from RStudio's selected text. Default is TRUE.
verbose	A logical value indicating whether to print the result to the console, default is TRUE.
SlowTone	A logical value indicating whether to print the result slowly, default is FALSE.

Details

Create R Code from Selected Text or Clipboard Content

This function reads text from your selected text or clipboard, interprets it as a prompt, and generates R code based on the given input. The generated R code is then printed to the R console with optional slow printing. This function can be executed from RStudio's Addins menu.

Value

Prints the generated R code to the R console.

RStudio Addins

This function can be added to RStudio's Addins menu for easy access.

Author(s)

Satoshi Kume

Examples

```
## Not run:

# Copy the origin of R code to your clipboard then execute from RStudio's Addins.


## End(Not run)
```

---

createRfunction *Create R Function from Selected Text or Clipboard Content and Output into the R Console*

Description

This function reads text either from your selected text in RStudio or from the clipboard, interprets it as a prompt, and generates an R function based on the given input. The generated R code is then printed into the source file or the R console with optional slow printing.

Usage

```
createRfunction(
  Model = "gpt-4o-mini",
  SelectedCode = TRUE,
  verbose = TRUE,
  SlowTone = FALSE
)
```

Arguments

Model	A character string representing the model to be used. Default is "gpt-4o-mini".
SelectedCode	A logical value indicating if the selected text should be used as input. Default is TRUE.
verbose	A logical value indicating if progress should be printed. Default is TRUE.
SlowTone	A logical value indicating if slow printing should be used. Default is FALSE.

Details

Create R Function from Selected Text or Clipboard Content

Value

This function returns the generated R code as a clipboard content if SelectedCode is FALSE.

Author(s)

Satoshi Kume

Examples

```
## Not run:

#Copy the idea text of the R function to your clipboard and run this function.
createRfunction(SelectedCode = FALSE)

## End(Not run)
```

---

createSpecifications4R

Create Specifications for R Function

Description

This function generates specifications for an R function from your selected text or clipboard. It takes in a text input, model name, verbosity, and tone speed to generate the specifications.

Usage

```
createSpecifications4R(
  Model = "gpt-5-nano",
  SelectedCode = TRUE,
  verbose = TRUE,
  SlowTone = FALSE
)
```

Arguments

Model	A character string specifying the GPT model to be used. Default is "gpt-5-nano".
SelectedCode	A logical flag to indicate whether to read from RStudio's selected text. Default is TRUE.
verbose	A logical value indicating whether to print the output. Default is TRUE.
SlowTone	A logical value indicating whether to print the output slowly. Default is FALSE.

Details

Create Specifications for R Function

Value

The function prints the generated specifications to the console.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Option 1
# Select some text in RStudio and then run the rstudio addins
# Option 2
# Copy the text into your clipboard then execute
clipr::write_clip("A function which can compute the mean of a vector of any size")
createSpecifications4R(Model = "gpt-5-nano", SelectedCode = FALSE, verbose = TRUE, SlowTone = FALSE)

## End(Not run)
```

designPackage *designPackage*

Description

Assist in proposing the overall design and architecture of an R package.

Usage

```
designPackage(Model = "gpt-4o-mini", verbose = TRUE, SlowTone = FALSE)
```

Arguments

Model	The OpenAI GPT model to use for text generation. Default is "gpt-4o-mini".
verbose	Logical flag to indicate whether to display the generated text. Default is TRUE.
SlowTone	Logical flag to indicate whether to print the text slowly. Default is FALSE.

Details

Design Package for R

This function assists in proposing the overall design and architecture of an R package. It uses the OpenAI GPT model for text generation to assist in the design process. The function reads the input from the clipboard.

Value

Prints the proposed design and architecture based on the verbosity and tone speed settings.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Copy the text into your clipboard then execute
designPackage(Model = "gpt-4o-mini", verbose = TRUE, SlowTone = FALSE)

## End(Not run)
```

DifyChat4R *DifyChat4R Function (Completion Messages Only)*

Description

This function sends a query to the Dify API using the completion-messages endpoint in blocking mode. It provides a simple interface for interacting with Dify AI applications. The function uses the stable blocking response mode to avoid streaming parsing issues.

Usage

```
DifyChat4R(
  query,
  user = "abc-123",
  api_key = Sys.getenv("DIFY_API_KEY"),
  conversation_id = "",
  files = NULL
)
```

Arguments

query	A character string representing the user's input query.
user	A character string representing the user identifier. Default is "abc-123".
api_key	A character string for the Dify API secret key. Defaults to the value of the environment variable "DIFY_API_KEY".
conversation_id	A character string representing the conversation ID. Default is an empty string.
files	A list of lists representing file information to be sent with the query. Default is NULL. Each file should be a list containing at least the keys: - type (e.g., "image") - transfer_method (e.g., "remote_url") - url (the file URL)

Value

A list containing the parsed JSON response from the Dify API.

Examples

```
## Not run:
  # Set your Dify API key in the environment or pass it directly
  Sys.setenv(DIFY_API_KEY = "YOUR-DIFY-SECRET-KEY")

  # Basic usage
  response <- DifyChat4R(query = "Hello world!")
  print(response)

  # With conversation context
  response <- DifyChat4R(
```

```
  query = "How are you?",
  conversation_id = "conv-123",
  user = "user-456"
)
print(response)

## End(Not run)
```

`discussion_flow_v1` *discussion_flow_v1: Interactions and Flow Control Between LLM-based Bots (LLBs)*

Description

Simulates interactions and flow control between three different roles of LLM-based bots (LLBs).

Usage

```
discussion_flow_v1(
  issue,
  Domain = "bioinformatics",
  Model = "gpt-5-nano",
  api_key = Sys.getenv("OPENAI_API_KEY"),
  language = "English",
  Summary_nch = 50,
  verbose = TRUE
)
```

Arguments

`issue`	The issue to be discussed. Example: "I want to solve linear programming and create a timetable."
`Domain`	The domain of the discussion, default is "bioinformatics".
`Model`	The model to be used, default is "gpt-5-nano".
`api_key`	The API key for OpenAI, default is retrieved from the system environment variable "OPENAI_API_KEY".
`language`	The language for the discussion, default is "English".
`Summary_nch`	The number of characters for the summary, default is 50.
`verbose`	Logical, whether to print verbose output, default is TRUE.

Details

Interactions and Flow Control Between LLM-based Bots (LLBs)

This function is described to simulate the interactions and flow control between three different roles of LLM-based bots, abbreviated as LLBs, and to reproduce more realistic dialogues and discussions. Here is a brief description of the roles: A (Beginner): This bot generates questions and summaries based on the content of the discussion provided by the user. B (Expert): This bot provides professional answers to questions posed by LLB A. C (Peer Reviewer): This bot reviews the dialog between LLB A and LLB B and suggests improvements or refinements. The three parties independently call the OpenAI API according to their roles. In addition, it keeps track of the conversation history between the bots and performs processes such as questioning, answering, and peer review. The function is designed to work in a "domain," which is essentially a specific area or topic around which conversations revolve. It is recommended to use GPT-4 or a model with higher accuracy than GPT-4. English is recommended as the input language, but the review will also be conducted in Japanese, the native language of the author.

Value

A summary of the conversation between the bots.

Author(s)

Satoshi Kume

Examples

```
## Not run:
issue <- "I want to discuss about a potentiality of multivariate analysis for metabolomics data."

#Run Discussion with the domain of bioinformatics
discussion_flow_v1(issue)

## End(Not run)
```

`discussion_flow_v2` *discussion_flow_v2: Interactions and Flow Control Between LLM-based Bots (LLBs)*

Description

Simulates interactions and flow control between three different roles of LLM-based bots (LLBs).

Usage

```
discussion_flow_v2(
  issue,
  Domain = "bioinformatics",
  Model = "gpt-4o-mini",
```

```
  api_key = Sys.getenv("OPENAI_API_KEY"),
  language = "English",
  Summary_nch = 50,
  Sentence_difficulty = 2,
  R_expert_setting = TRUE,
  verbose = TRUE,
  sayENorJA = TRUE,
  rep_x = 3
)
```

Arguments

issue	The issue to be discussed. Example: "I want to perform differential gene expression analysis from RNA-seq data and interpret enriched pathways."
Domain	The domain of the discussion, default is "bioinformatics".
Model	The LLM model to be used, default is "gpt-4o-mini".
api_key	The API key for OpenAI, default is retrieved from the system environment variable "OPENAI_API_KEY".
language	The language for the discussion, default is "English".
Summary_nch	The number of characters for the summary, default is 50.
Sentence_difficulty	Numeric, the complexity level for sentence construction, default is 2.
R_expert_setting	Logical, whether R expert settings are enabled, default is TRUE.
verbose	Logical, whether to print verbose output, default is TRUE.
sayENorJA	Logical, whether to speak in English or Japanese, default is TRUE. This feature is available on macOS systems only.
rep_x	Numeric, a number of repeats for the conversations, default is 3.

Details

Interactions and Flow Control Between LLM-based Bots (LLBs)

In the v2 model, we added a regulation of the difficulty of the sentence, the human intervention in their conversation between LLM bots, and number of repetitions of conversation. This function is described to simulate the interactions and flow control between three different roles of LLM-based bots, abbreviated as LLBs, namely A (Beginner), B (Expert), and C (Peer Reviewer). These roles have distinct functions and work together to facilitate more complex and meaningful discussions. Here is a brief description of the roles: A (Beginner): This bot generates questions and summaries based on the content of the discussion provided by the user. B (Expert): This bot provides professional answers to questions posed by LLB A. C (Peer Reviewer): This bot reviews the dialog between LLB A and LLB B and suggests improvements or refinements. The three parties independently call the OpenAI API according to their roles. In addition, it keeps track of the conversation history between the bots and performs processes such as questioning, answering, and peer review. The function is designed to work in a "domain," which is essentially a specific area or topic around which conversations revolve. It is recommended to use GPT-4 or a model with higher accuracy than GPT-4. English is recommended as the input language, but the review will also be conducted in Japanese, the native language of the author.

Value

A summary of the conversation between the bots.

Author(s)

Satoshi Kume

Examples

```
## Not run:
issue <-  "I want to perform differential gene expression analysis from
           RNA-seq data and interpret enriched pathways."

#Run Discussion with the domain of bioinformatics
discussion_flow_v2(issue)

## End(Not run)
```

---

enrichTextContent *Enrich Text Content v2*

Description

This function doubles the amount of text without changing its meaning. The GPT-4 model is currently recommended for text generation. It can either read from the RStudio selection or the clipboard.

Usage

```
enrichTextContent(Model = "gpt-4o-mini", SelectedCode = TRUE, verbose = TRUE)
```

Arguments

Model	A character string specifying the AI model to be used for text enrichment. Default is "gpt-4o-mini".
SelectedCode	A logical flag to indicate whether to read from RStudio's selected text. Default is TRUE.
verbose	Logical flag to indicate whether to display the generated text. Default is TRUE.

Details

Enrich Text Content

Value

If SelectedCode is TRUE, the enriched text is inserted into the RStudio editor and a message "Finished!!" is returned. Otherwise, the enriched text is placed into the clipboard.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Option 1
# Select some text in RStudio and then run the rstudio addins
# Option 2
# Copy the text into your clipboard then execute
enrichTextContent(Model = "gpt-4o-mini", SelectedCode = TRUE)

## End(Not run)
```

---

extractKeywords *extractKeywords*

Description

Extract keywords from input text and output them in a comma-separated format.

Usage

```
extractKeywords(Model = "gpt-4o-mini", verbose = TRUE, SlowTone = FALSE)
```

Arguments

Model	The OpenAI GPT model to use for text generation. Default is "gpt-4o-mini".
verbose	Logical flag to indicate whether to print the result. Default is TRUE.
SlowTone	Logical flag to indicate the speed of the output. Default is FALSE.

Details

Extract Keywords from Text

This function extracts keywords from the input text. It uses the OpenAI GPT model for text generation to assist in the extraction process. The function reads the input from the clipboard and outputs the extracted keywords in a comma-separated format.

Value

Prints the extracted keywords based on verbosity and tone speed settings.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Option 1
# Select some text in RStudio and then run the rstudio addins
# Option 2
# Copy the text into your clipboard then execute
extractKeywords(Model = "gpt-4o-mini", verbose = TRUE, SlowTone = FALSE)

## End(Not run)
```

---

gemini4R *Gemini API Request (v1beta, 2025-07)*

Description

A thin, dependency-light R wrapper for Google Gemini API (‘models.generateContent‘ / ‘models.streamGenerateContent‘).

Usage

```
gemini4R(
  mode,
  contents,
  model = "gemini-2.0-flash",
  store_history = FALSE,
  api_key = Sys.getenv("GoogleGemini_API_KEY"),
  max_tokens = 2048,
  ...
)
```

Arguments

mode	One of ‘"text"‘, ‘"stream_text"‘, ‘"chat"‘, ‘"stream_chat"‘.
contents	Character vector (single-turn) or list of message objects (chat modes). See Examples.
model	Gemini model ID. Default ‘"gemini-2.0-flash"‘.
store_history	Logical. If TRUE, chat history is persisted to the ‘chat_history‘ env-var (JSON).
api_key	Your Google Gemini API key (default: ‘Sys.getenv("GEMINI_API_KEY")‘).
max_tokens	Maximum output tokens. NULL for server default.
...	Additional ‘httr::POST‘ options (timeouts etc.).

Value

For non-stream modes, a parsed list. For stream modes, a list with ‘full_text‘ and ‘chunks‘.

Author(s)

Satoshi Kume (revised 2025-07-01)

Examples

```
## Not run:
gemini4R("text",
         contents = "Explain how AI works.",
         max_tokens = 256)

## End(Not run)
```

`geminiGrounding4R` *Gemini API Google Search Grounding Request (v1/v1beta, 2025-07)*

Description

A thin R wrapper for Google Gemini API with Google Search grounding enabled. Grounding improves factuality by incorporating web search results.

Usage

```
geminiGrounding4R(
  mode,
  contents,
  model = "gemini-2.5-flash",
  api_version = "v1beta",
  store_history = FALSE,
  dynamic_threshold = 0.7,
  api_key = Sys.getenv("GoogleGemini_API_KEY"),
  max_tokens = 2048,
  debug = FALSE,
  enable_grounding = TRUE,
  ...
)
```

Arguments

`mode`	One of ‘"text"‘, ‘"stream_text"‘, ‘"chat"‘, ‘"stream_chat"‘.
`contents`	Character vector (single-turn) or list of message objects (chat modes). See Examples.
`model`	Gemini model ID. Default ‘"gemini-2.5-flash"‘.
`api_version`	API version for the Generative Language API (‘"v1"‘ or ‘"v1beta"‘). Default ‘"v1beta"‘.
`store_history`	Logical. If TRUE, chat history is persisted to the ‘chat_history‘ env-var (JSON).

dynamic_threshold
: Numeric [0,1] for dynamic retrieval threshold (default: 0.7). Only used for Gemini 1.5 models with 'googleSearchRetrieval'; ignored for newer models that use 'google_search'.

api_key
: Your Google Gemini API key (default: 'Sys.getenv("GoogleGemini_API_KEY")').

max_tokens
: Maximum output tokens. Default 2048.

debug
: Logical. If TRUE, prints request details for debugging.

enable_grounding
: Logical. If TRUE, enables Google Search grounding (default).

...
: Additional 'httr::POST' options (timeouts etc.).

Value

For non-stream modes, a parsed list. For stream modes, a list with 'full_text' and 'chunks'.

Author(s)

Satoshi Kume (revised 2025-07-01)

Examples

```
## Not run:
  # Synchronous text generation with grounding:
  result <- geminiGrounding4R(
    mode = "text",
    contents = "What is the current Google stock price?",
    store_history = FALSE,
    debug = TRUE,  # Enable debug to see request details
    api_key = Sys.getenv("GoogleGemini_API_KEY")
  )
  print(result)

  # Basic text generation without grounding (for troubleshooting):
  basic_result <- geminiGrounding4R(
    mode = "text",
    contents = "Hello, how are you?",
    enable_grounding = FALSE,
    debug = TRUE,
    api_key = Sys.getenv("GoogleGemini_API_KEY")
  )
  print(basic_result)

  # Chat mode with history storage:
  chat_history <- list(
    list(role = "user", text = "Hello"),
    list(role = "model", text = "Hi there! How can I help you?")
  )
  chat_result <- geminiGrounding4R(
    mode = "chat",
    contents = chat_history,
```

```
  store_history = TRUE,
  dynamic_threshold = 0.7,
  api_key = Sys.getenv("GoogleGemini_API_KEY")
)
print(chat_result)

# Streaming text generation:
stream_result <- geminiGrounding4R(
  mode = "stream_text",
  contents = "Tell me a story about a magic backpack.",
  store_history = FALSE,
  dynamic_threshold = 0.7,
  api_key = Sys.getenv("GoogleGemini_API_KEY")
)
print(stream_result$full_text)

## End(Not run)
```

`interpretResult` *Interpret Analysis Results*

Description

This function constructs an interpretation prompt based on the analysis type and passes it to the 'chat4R' function.

Usage

```
interpretResult(
  analysis_type,
  result_text,
  custom_template = NULL,
  model = "gpt-5-nano",
  temperature = 1,
  api_key = Sys.getenv("OPENAI_API_KEY"),
  use_fallback = TRUE,
  fallback_model = "gpt-5-nano"
)
```

Arguments

`analysis_type`	A character string indicating the type of analysis. Valid values include "summary", "PCA", "regression", "group_comparison", "visualization", "time_series", "clustering", "biological_implication", "statistical_metrics", "test_validity", "report", "preprocessing", and "custom".
`result_text`	An object containing the analysis result to be interpreted. If it is not a character string, it will be converted to one using capture.output.

custom_template
: An optional custom prompt template to be used when analysis_type is "custom". If NULL, a default prompt is used.

model
: The chat model to use (default: "gpt-5-nano"). Set to another supported model string if needed.

temperature
: Sampling temperature passed to the chat model (default: 1).

api_key
: API key passed to ‘chat4R‘ (defaults to OPENAI_API_KEY).

use_fallback
: If TRUE, falls back to ‘fallback_model‘ when the primary model errors.

fallback_model
: Model used when the primary model errors (default: "gpt-5-nano"). Ignored if ‘use_fallback = FALSE‘ or the primary succeeds.

Value

The interpretation produced by AI (data.frame with a ‘content‘ column by default)

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Example: interpret PCA results of the iris dataset
pca_res <- prcomp(iris[, 1:4], scale. = TRUE)
interpretation <- interpretResult("PCA", summary(pca_res))
cat(interpretation$content)

## End(Not run)
```

---

ionet_models *ionet_models: Retrieve the available model list from io.net API*

Description

Convenience wrapper around list_ionet_models() to get the current model list. By default it uses cached data; set force_refresh = TRUE to refetch.

Usage

```
ionet_models(api_key = NULL, verbose = TRUE, force_refresh = FALSE)
```

Arguments

api_key
: Optional API key for fetching current models. Defaults to IONET_API_KEY environment variable.

verbose
: Logical indicating whether to show detailed fetching information. Default is TRUE.

force_refresh
: Logical indicating whether to force refresh the model list from API. Default is FALSE.

Details

List io.net Models (simple alias)

Value

A character vector of model names, or a data frame if detailed=TRUE in the underlying call.

`list_ionet_models` *list_ionet_models: Display available LLM models on io.net API*

Description

This function returns a list of all available LLM models that can be used with the multiLLMviaionet function. Models are categorized by family/provider. This function will attempt to fetch the current model list from the io.net API.

Usage

```
list_ionet_models(
  category = "all",
  detailed = FALSE,
  api_key = NULL,
  force_refresh = FALSE,
  verbose = FALSE
)
```

Arguments

`category`	Optional character string to filter models by category. Options: "llama", "deepseek", "qwen", "phi", "mistral", "specialized", "all". Default is "all".
`detailed`	Logical indicating whether to show detailed model information. Default is FALSE.
`api_key`	Optional API key for fetching current models. Defaults to IONET_API_KEY environment variable.
`force_refresh`	Logical indicating whether to force refresh the model list from API. Default is FALSE.
`verbose`	Logical indicating whether to show detailed fetching information. Default is FALSE.

Details

List Available io.net Models

Value

A character vector of model names, or a data frame if detailed=TRUE

Author(s)

Satoshi Kume

Examples

```
## Not run:
  # List all available models (from API if available)
  all_models <- list_ionet_models()

  # Force refresh from API
  fresh_models <- list_ionet_models(force_refresh = TRUE)

  # List only Llama models
  llama_models <- list_ionet_models("llama")

  # Show detailed information
  model_info <- list_ionet_models(detailed = TRUE)

## End(Not run)
```

`multiLLMviaionet` *multiLLMviaionet: Execute ALL available LLM models simultaneously via io.net API*

Description

This function automatically runs the same prompt across ALL currently available LLM models on the io.net API (by default it now selects one random model to reduce API load). It supports running multiple models (configurable) in parallel with streaming responses and comprehensive error handling.

The function dynamically fetches the current list of available models from the io.net API and executes a configurable subset (default 1 random model; set max_models or random_selection to control). Results are cached for 1 hour to improve performance. If the API is unavailable, it falls back to a static model list.

Typical models observed on io.net (30-Nov-2025) include: - moonshotai/Kimi-K2-Thinking, moonshotai/Kimi-K2-Instruct-0905 - zai-org/GLM-4.6 - meta-llama/Llama-4-Maverick-17B-128E-Instruct-FP8, Llama-3.3-70B-Instruct, Llama-3.2-90B-Vision-Instruct - Qwen/Qwen3-235B-A22B-Thinking-2507, Qwen3-Next-80B-A3B-Instruct, Qwen2.5-VL-32B-Instruct, Intel/Qwen3-Coder-480B-A35B-Instruct-int4-mixed-ar - deepseek-ai/DeepSeek-R1-0528 - mistralai/Devstral-Small-2505, Magistral-Small-2506, Mistral-Large-Instruct-2411, Mistral-Nemo-Instruct-2407 - openai/gpt-oss-120b, openai/gpt-oss-20b

Usage

```
multiLLMviaionet(
  prompt,
  max_models = 1,
```

```
  streaming = FALSE,
  random_selection = TRUE,
  api_key = Sys.getenv("IONET_API_KEY"),
  max_tokens = 1024,
  temperature = 0.7,
  timeout = 300,
  parallel = TRUE,
  verbose = TRUE,
  refresh_models = FALSE,
  retries = 1,
  retry_wait = 2,
  monitor_timeout = 120
)
```

Arguments

prompt	A string containing the input prompt to send to all selected models.
max_models	Integer specifying maximum number of models to run (1-50). Default is 1 (single model).
streaming	Logical indicating whether to use streaming responses. Default is FALSE.
random_selection	Logical indicating whether to randomly select models from all available models. If TRUE (default), randomly selects up to max_models from all available models. If FALSE, uses models in order (up to max_models limit).
api_key	A string containing the io.net API key. Defaults to the environment variable "IONET_API_KEY".
max_tokens	Integer specifying maximum tokens to generate per model. Default is 1024.
temperature	Numeric value controlling randomness (0-2). Default is 0.7.
timeout	Numeric value in seconds for request timeout per model. Default is 300.
parallel	Logical indicating whether to run models in parallel. Default is TRUE.
verbose	Logical indicating whether to show detailed progress. Default is TRUE.
refresh_models	Logical indicating whether to force-refresh the io.net model list before execution. Default is FALSE.
retries	Integer number of retries for transient errors (429/5xx/timeout/connection). Default is 1.
retry_wait	Base wait in seconds before retry (multiplied by attempt count). Default is 2.
monitor_timeout	Maximum seconds to wait for all async futures in the progress loop (prevents hangs). Default is 120.

Details

Multi-LLM via io.net API

Value

A list containing:

results List of responses from each model with metadata

summary Summary statistics including execution times and token usage

errors Any errors encountered during execution

models_used Character vector of models that were actually executed

Author(s)

Satoshi Kume

Examples

```
## Not run:
  # Set your io.net API key
  Sys.setenv(IONET_API_KEY = "your_ionet_api_key_here")

  # Basic usage - single random model (default)
  result_one <- multiLLMviaionet(
    prompt = "Explain quantum computing in simple terms"
  )
  print(result_one$summary)
  print_multiLLM_results(result_one)

  # Run first 5 models in list order (no random selection)
  result_top5 <- multiLLMviaionet(
    prompt = "What is machine learning?",
    max_models = 5,
    random_selection = FALSE
  )
  print_multiLLM_results(result_top5)

  # Random selection of 10 models from all available
  result_random10 <- multiLLMviaionet(
    prompt = "Write a Python function to calculate fibonacci numbers",
    max_models = 10,
    random_selection = TRUE,
    temperature = 0.3,
    streaming = FALSE
  )
  print_multiLLM_results(result_random10)

## End(Not run)
```

ngsub *ngsub*

Description

Remove extra spaces and newline characters from text.

Usage

```
ngsub(text)
```

Arguments

text	The input text from which extra spaces and newline characters need to be removed.

Details

Remove Extra Spaces and Newline Characters

This function removes extra spaces and newline characters from the given text. It replaces sequences of multiple spaces with a single space and removes newline characters followed by a space.

Value

Returns the modified text as a character string.

Author(s)

Satoshi Kume

Examples

```
## Not run:
ngsub("This is  a text \n with  extra   spaces.")

## End(Not run)
```

`OptimizeRcode` *Optimize and Complete R Code*

Description

Optimizes and completes the R code from the selected code or clipboard.

Usage

```
OptimizeRcode(
  Model = "gpt-4o-mini",
  SelectedCode = TRUE,
  verbose = TRUE,
  verbose_Reasons4change = FALSE,
  SlowTone = FALSE
)
```

Arguments

`Model`	A string specifying the machine learning model to use for code optimization. Default is "gpt-4o-mini".
`SelectedCode`	A boolean indicating whether to get the code from RStudio or the clipboard. Default is TRUE.
`verbose`	A logical value indicating whether to print the result to the console, default is TRUE.
`verbose_Reasons4change`	A boolean indicating whether to provide detailed reasons for the changes made. Default is FALSE
`SlowTone`	A boolean indicating whether to print the output slowly. Default is FALSE.

Details

Optimize and Complete R Code

This function takes a snippet of R code and optimizes it for performance and readability. It uses a machine learning model to generate the optimized code.

Value

A message indicating the completion of the optimization process.

Author(s)

Satoshi Kume

Examples

```
## Not run:
#Copy your R code then run the following function.
OptimizeRcode(SelectedCode = FALSE)

## End(Not run)
```

`print.multiLLM_result` *Print method for multiLLM_result*

Description

Print method for multiLLM_result

Usage

```
## S3 method for class 'multiLLM_result'
print(x, ...)
```

Arguments

`x`	A multiLLM_result object
`...`	Additional arguments (unused)

`print_multiLLM_results` *Pretty-print multiLLMviaionet results*

Description

Display model responses with status, handling NULLs and errors cleanly.

Usage

```
print_multiLLM_results(result, show_failed = TRUE, trim = NULL)
```

Arguments

`result`	A result object returned by ‘multiLLMviaionet()‘.
`show_failed`	Logical; if TRUE (default), also prints failed/timeout models and their errors.
`trim`	Optional integer; if not NULL, truncates long responses to the first ‘trim‘ characters for display.

Value

Invisibly returns NULL.

`proofreadEnglishText` *Proofread English Text*

Description

A function to proofread English text or text in different languages during R package development. It translates the input into English if necessary and returns meticulously checked English text.

Usage

```
proofreadEnglishText(
  Model = "gpt-4o-mini",
  SelectedCode = TRUE,
  verbose = TRUE
)
```

Arguments

`Model`	A string specifying the model to be used for proofreading, defaulting to "gpt-4o-mini". Currently, "gpt-4", "gpt-4o-mini" and "gpt-4-0613" can be selected as gpt-4 models. Execution with GPT-4 is recommended.
`SelectedCode`	A logical value indicating whether to read the selected text from the RStudio editor (TRUE) or from the clipboard (FALSE). Defaults to TRUE.
`verbose`	Logical flag to print the progress. Default is TRUE.

Details

Proofread English Text During R Package Development via RStudio API

This function provides a feature to proofread English text during the development of an R package. It can either take the selected text from the RStudio editor or read from the clipboard, executes the proofreading, and returns the result to the user's clipboard or replaces the selected text. The user can then paste and check the result if read from the clipboard. The function adheres to R package policies and carefully proofreads the English text. Execution with GPT-4 is recommended.

Value

The proofread text, which is also written to the clipboard if SelectedCode is FALSE, or replaces the selected text if SelectedCode is TRUE.

Author(s)

Satoshi Kume

Examples

```
## Not run:
  # Proofreading selected text in RStudio
  proofreadEnglishText(Model = "gpt-4o-mini", SelectedCode = TRUE)
  # Proofreading text from the clipboard
  proofreadEnglishText(Model = "gpt-4o-mini", SelectedCode = FALSE)

## End(Not run)
```

`proofreadText` *proofreadText*

Description

Proofreads text during the development of an R package.

Usage

```
proofreadText(Model = "gpt-4o-mini", SelectedCode = TRUE, verbose = TRUE)
```

Arguments

`Model`	The Large Language Model to be used for proofreading. Default is "gpt-4o-mini".
`SelectedCode`	Logical flag to indicate whether to use the selected text in RStudio editor. Default is TRUE.
`verbose`	Logical flag to print the progress. Default is TRUE.

Details

Proofread Text During R Package Development Through the RStudio API

This function offers a feature for proofreading text while developing an R package. It can either use the text selected in the RStudio editor or read from the clipboard, perform the proofreading, and then either replace the selected text or return the result to the user's clipboard. The language of the output will match the language of the input text. Using GPT-4 for execution is recommended.

Value

NULL if 'SelectedCode' is TRUE, otherwise returns the proofread text to the clipboard.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Proofread text from clipboard
proofreadText(SelectedCode = FALSE)

## End(Not run)
```

`RcodeImprovements` *Suggest Improvements to the R Code on Your Clipboard*

Description

This function uses LLM to analyze the R code from the clipboard and suggests improvements. The function can also control the verbosity and speed of the output.

Usage

```
RcodeImprovements(
  Summary_nch = 100,
  Model = "gpt-4o-mini",
  verbose = TRUE,
  SlowTone = FALSE
)
```

Arguments

`Summary_nch`	An integer specifying the maximum number of characters for the summary. Default is 100.
`Model`	A character string specifying the GPT model to be used. Default is "gpt-4o-mini".
`verbose`	A logical value indicating whether to print the result. Default is TRUE.
`SlowTone`	A logical value indicating whether to print the result slowly. Default is FALSE.

Details

Suggest Improvements for R Code

Value

No return value; the function prints the suggestions for code improvement.

Author(s)

Satoshi Kume

Examples

```
## Not run:
#Copy your function to your clipboard
RcodeImprovements(Summary_nch = 100, Model = "gpt-4o-mini")

## End(Not run)
```

`refresh_ionet_models` *refresh_ionet_models: Manually refresh the cached model list from io.net API*

Description

This function forces a refresh of the model list cache by fetching the current models from the io.net API. Use this if you suspect the model list has changed and you want to update immediately rather than waiting for the 1-hour cache to expire.

Usage

```
refresh_ionet_models(api_key = NULL, verbose = TRUE)
```

Arguments

`api_key`	Optional API key for fetching current models. Defaults to IONET_API_KEY environment variable.
`verbose`	Logical indicating whether to show detailed fetching information. Default is TRUE.

Details

Refresh io.net Model Cache

Value

A character vector of current model names from the API, or NULL if the API call failed

Author(s)

Satoshi Kume

Examples

```
## Not run:
  # Refresh model list from API
  current_models <- refresh_ionet_models()

  # Refresh silently
  current_models <- refresh_ionet_models(verbose = FALSE)

## End(Not run)
```

`removeQuotations` *Remove All Types of Quotations from Text*

Description

This function takes a text string as input and removes all occurrences of single, double, and back quotations marks.

Usage

```
removeQuotations(text)
```

Arguments

`text`	A character string from which quotations will be removed.

Value

A character string with all types of quotations removed.

Author(s)

Satoshi Kume

Examples

```
## Not run:
removeQuotations("\"XXX'`\"YYY'`") # Returns "XXXYYY"

## End(Not run)
```

`replicatellmAPI4R` *replicatellmAPI4R: Interact with Replicate API for LLM models in R*

Description

This function interacts with the Replicate API (v1) to utilize language models (LLM) such as Llama. It sends a POST request with the provided input and handles both streaming and non-streaming responses.

Note: This function is primarily designed and optimized for Large Language Models (LLMs). While the Replicate API supports various model types (e.g., image generation, audio models), this function's features (especially 'collapse_tokens') are tailored for LLM text outputs. For non-LLM models, consider setting 'collapse_tokens = FALSE' or use the full response mode with 'simple = FALSE'.

Usage

```
replicatellmAPI4R(
  input,
  model_url,
  simple = TRUE,
  fetch_stream = FALSE,
  collapse_tokens = TRUE,
  api_key = Sys.getenv("Replicate_API_KEY")
)
```

Arguments

input	A list containing the API request body with parameters including prompt, max_tokens, top_k, top_p, min_tokens, temperature, system_prompt, presence_penalty, and frequency_penalty.
model_url	A character string specifying the model endpoint URL (e.g., "/models/meta/meta-llama-3.1-405b-instruct/predictions").
simple	A logical value indicating whether to return a simplified output (only the model output) if TRUE, or the full API response if FALSE. Default is TRUE.
fetch_stream	A logical value indicating whether to fetch a streaming response. Default is FALSE.
collapse_tokens	A logical value indicating whether to collapse token vectors into a single string for LLM text outputs. When TRUE (default), token vectors like c("The", " capital", " of", " France") are automatically combined into "The capital of France". This parameter only affects LLM text outputs and has no effect on other model types (e.g., image URLs, audio data). Default is TRUE.
api_key	A character string representing the Replicate API key. Defaults to the environment variable "Replicate_API_KEY".

Value

If fetch_stream is FALSE, returns either a simplified output (if simple is TRUE) or the full API response. When simple is TRUE and collapse_tokens is TRUE, returns a single character string. In streaming mode, outputs the response stream directly to the console.

Author(s)

Satoshi Kume

Examples

```
## Not run:
  Sys.setenv(Replicate_API_KEY = "Your API key")
  input <- list(
    input = list(
      prompt = "What is the capital of France?",
      max_tokens = 1024,
```

```
    top_k = 50,
    top_p = 0.9,
    min_tokens = 0,
    temperature = 0.6,
    system_prompt = "You are a helpful assistant.",
    presence_penalty = 0,
    frequency_penalty = 0
  )
)
model_url <- "/models/meta/meta-llama-3.1-405b-instruct/predictions"

# Default: collapse_tokens = TRUE (returns single string)
response <- replicatellmAPI4R(input, model_url, fetch_stream = TRUE)
print(response)
# [1] "The capital of France is Paris."

# Keep tokens separated
response_tokens <- replicatellmAPI4R(input, model_url, fetch_stream = TRUE,
                                     collapse_tokens = FALSE)
print(response_tokens)
# [1] "The" " capital" " of" " France" " is" " Paris" "."

## End(Not run)
```

`revisedText` *Revised Scientific Text*

Description

This function prompts the user to input text, revision comments, and additional background information. It then revises the text according to the comments and outputs the revised text.

Usage

```
revisedText(verbose = TRUE)
```

Arguments

`verbose`	Logical, whether to output verbose messages, default is TRUE.

Details

Revise Scientific Text

Value

Revised text or relevant message.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Interactive paper revision workflow
revisedText()

# Example interaction:
# Q: Do you have text ready for input? → Yes
# Q: Do you want to use your clipboard as input text? → No
# → You enter: "The result shows significant increase in accuracy."
#
# Q: Do you have revision comments? → Yes
# Q: Do you want to use your clipboard as revision comments? → No
# → You enter: "Please make it more formal and add statistical details"
#
# Q: Do you have additional background information? → No
#
# Q: Which language model do you prefer?
# → You select: 1 (gpt-5-nano)
#
# Output: AI will revise the text according to your comments

## End(Not run)
```

---

searchFunction *Search R Functions based on Text*

Description

Searches for an R function related to the provided text either through the RStudio editor selection or clipboard.

Usage

```
searchFunction(
  Summary_nch = 100,
  Model = "gpt-4o-mini",
  SelectedCode = TRUE,
  verbose = TRUE,
  SlowTone = FALSE
)
```

Arguments

Summary_nch	Numeric, number of characters to limit the function description (default = 100).
Model	String, the model used for the search, default is "gpt-4o-mini".
SelectedCode	Logical, whether to get text from RStudio selection (default = TRUE).
verbose	Logical, whether to print the results verbosely (default = TRUE).
SlowTone	Logical, whether to slow down the print speed for readability (default = FALSE).

Details

Search the R function based on the provided text

This function searches for an R function that corresponds to the text provided either through the RStudio editor selection or the clipboard. It fetches the related R function and outputs its name, package, and a brief description. The function uses GPT-4 for its underlying search.

Value

Console output of the identified R function, its package, and a brief description.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# To search for an R function related to "linear regression"
searchFunction(Summary_nch = 50, SelectedCode = FALSE)

## End(Not run)
```

---

slow_print_v2 *Slowly Print Text*

Description

Prints the characters of the input text string one by one, with a specified delay between each character. If the random parameter is set to TRUE, the delay will be a random value between 0.0001 and 0.3 seconds. Otherwise, the delay will be the value specified by the delay parameter.

Usage

```
slow_print_v2(text, random = FALSE, delay = 0.125)
```

Arguments

text	A string representing the text to be printed. Must be a non-NA string.
random	A logical value indicating whether the delay between characters should be random. Default is FALSE.
delay	A numeric value representing the fixed delay between characters in seconds. Default is 0.125. Must be a non-negative number.

Details

Slowly Print Text

This function prints the characters of a given text string one by one, with a specified delay between each character. The delay can be either fixed or random.

Value

Invisible NULL. The function prints the text to the console.

Author(s)

Satoshi Kume

Examples

```
## Not run:
slow_print_v2("Hello, World!")
slow_print_v2("Hello, World!", random = TRUE)
slow_print_v2("Hello, World!", delay = 0.1)

## End(Not run)
```

---

speakInEN *Speak Selected Text in English*

Description

This function reads aloud the selected text in English using the MacOS system's 'say' command. It uses the voice specified by the parameter 'systemVoice'. The default voice is "Alex". It only works on MacOS systems.

Usage

```
speakInEN(systemVoice = "Alex")
```

Arguments

systemVoice	The voice to be used for speech. Default is "Alex".

Details

Speak Selected Text in English on MacOS System

Value

A message indicating the completion of the speech is returned.

Author(s)

Satoshi Kume

Examples

```
## Not run:
  # Select some text in RStudio and then run the rstudio addin

## End(Not run)
```

---

speakInJA *Speak Selected Text in Japanese*

Description

This function reads aloud the selected text in Japanese using the MacOS system's 'say' command. The function specifically uses the 'Kyoko' voice for the speech. It only works on MacOS systems.

Usage

```
speakInJA()
```

Details

Speak Selected Text in Japanese on MacOS System

Value

A message indicating the completion of the speech is returned.

Author(s)

Satoshi Kume

Examples

```
## Not run:
  # Select some text in RStudio and then run the rstudio addins

## End(Not run)
```

speakInJA_v2 *Speak Clipboard in Japanese*

Description

This function reads aloud the clipboard content in Japanese using the MacOS system's 'say' command. The function specifically uses the 'Kyoko' voice for the speech. It only works on MacOS systems.

Usage

```
speakInJA_v2()
```

Details

Speak Clipboard in Japanese on MacOS System

Value

A message indicating the completion of the speech is returned.

Author(s)

Satoshi Kume

Examples

```
## Not run:
  # Copy some text into your clipboard in RStudio and then run the function
  speakInJA_v2()

## End(Not run)
```

supportIdeaGeneration *supportIdeaGeneration: Support Idea Generation from Selected Text or Clipboard.*

Description

Assist in generating ideas or concepts.

Usage

```
supportIdeaGeneration(
  Model = "gpt-4o-mini",
  SelectedCode = TRUE,
  verbose = TRUE,
  SlowTone = FALSE
)
```

Arguments

Model	The OpenAI GPT model to use for text generation. Default is "gpt-4o-mini".
SelectedCode	Logical flag to indicate whether to use the selected text in RStudio editor. Default is TRUE.
verbose	Logical flag to indicate whether to display the generated text. Default is TRUE.
SlowTone	Logical flag to indicate whether to print the text slowly. Default is FALSE.

Details

Support Idea Generation from Selected Text or Clipboard Input

This feature helps you generate ideas or concepts based on input from your selected text or clipboard. It uses the OpenAI GPT model for text generation to assist in the idea generation process. The function reads the input from the clipboard.

Value

Prints the generated ideas or concepts based on the verbosity and tone speed settings.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Option 1
# Select some text in RStudio and then run the rstudio addins
# Option 2
# Copy the text into your clipboard then execute
supportIdeaGeneration()

## End(Not run)
```

---

textEmbedding *Text Embedding from OpenAI Embeddings API*

Description

This function calls the OpenAI Embeddings API to get the multidimensional vector via text embedding of the input text. This function uses the 'text-embedding-3-small' model by default, with options for 'text-embedding-3-large' and legacy 'text-embedding-ada-002'.

Usage

```
textEmbedding(
  text,
  model = "text-embedding-3-small",
  api_key = Sys.getenv("OPENAI_API_KEY")
)
```

Arguments

text	A string. The input text to get the embedding for. This should be a character string.
model	A string. The embedding model to use. Options: "text-embedding-3-small" (default), "text-embedding-3-large", "text-embedding-ada-002" (legacy).
api_key	A string. The API key for the OpenAI API. Defaults to the value of the environment variable "OPENAI_API_KEY".

Value

A vector representing the text embeddings.

Author(s)

Satoshi Kume

Examples

```
## Not run:
Sys.setenv(OPENAI_API_KEY = "Your API key")
textEmbedding("Hello, world!")
textEmbedding("Hello, world!", model = "text-embedding-3-large")

## End(Not run)
```

---

textFileInput4ai *Send Text File Content to OpenAI API and Retrieve Response*

Description

This function reads the content of a specified text file, sends it to the OpenAI API using the provided API key, and retrieves the generated response from the GPT model. If the text content exceeds the max_input_chars threshold, it will be automatically split into smaller chunks based on character count and processed separately, with results returned as a list. The function handles invalid multibyte strings automatically by cleaning and converting text encoding. It can also handle files with header rows and displays progress during processing.

Usage

```
textFileInput4ai(
  file_path,
  model = "gpt-4o-mini",
  system_prompt = "You are a helpful assistant to analyze your input.",
  max_tokens = 1000,
  max_input_chars = 10000,
  api_key = Sys.getenv("OPENAI_API_KEY"),
  has_header = TRUE,
```

```
  show_progress = TRUE,
  summarize_results = FALSE
)
```

Arguments

file_path	A string representing the path to the text or csv file to be read and sent to the API.
model	A string specifying the OpenAI model to be used (default is "gpt-4o-mini"). The function automatically handles parameter compatibility for newer models (o3, o1, gpt-4o series) that require max_completion_tokens instead of max_tokens.
system_prompt	Optional. A system-level instruction that can be used to guide the model's behavior (default is "You are a helpful assistant to analyze your input.").
max_tokens	A numeric value specifying the maximum number of tokens to generate (default is 50).
max_input_chars	A numeric value specifying the maximum number of characters to send in a single API request. If the text content exceeds this value, it will be split into chunks (default is 10000).
api_key	A string containing the OpenAI API key. Defaults to the "OPENAI_API_KEY" environment variable.
has_header	Logical indicating whether the input file has a header row (default is TRUE).
show_progress	Logical indicating whether to display progress information during processing (default is TRUE).
summarize_results	Logical indicating whether to summarize the final results using the system prompt (default is FALSE). Only applies when text content is split into multiple chunks.

Value

If the text content is within the max_input_chars limit, returns a character string containing the response from the OpenAI API. If the content exceeds the limit, returns a list of responses. If the text file contains invalid multibyte characters, the function will attempt to clean and normalize the text before processing. If summarize_results is TRUE and chunks are processed, an additional summarized response will be returned as the last element of the list.

Author(s)

Satoshi Kume

Examples

```
## Not run:
  # Example usage of the function
  api_key <- "YOUR_OPENAI_API_KEY"
  file_path <- "path/to/your/text_file.txt"
  response <- textFileInput4ai(file_path, api_key = api_key, max_tokens = 50)
```

```
## End(Not run)
```

`TextSummary` *Summarize Long Text*

Description

This function summarizes a long text using LLM. The development of this function started with the idea that it might be interesting to perform a copy-and-paste, sentence summarization and aims to be an evangelist for copy-and-paste LLM execution. It is recommended to run this function with GPT-4, but it is not cost effective and slow. This is still an experimental feature.

Usage

```
TextSummary(
  text = clipr::read_clip(),
  nch = 2000,
  verbose = TRUE,
  returnText = FALSE
)
```

Arguments

`text`	A character vector containing the text to be summarized. If not provided, the function will attempt to read from the clipboard.
`nch`	Integer specifying the number of characters at which to split the input text for processing.
`verbose`	A logical flag to print the message. Default is TRUE.
`returnText`	A logical flag to return summarized text results. Default is FALSE.

Details

Summarize Long Text

Value

The summarized text is placed into the clipboard and the function returns the result of `clipr::write_clip`.

Author(s)

Satoshi Kume

Examples

```
## Not run:
TextSummary(text = c("This is a long text to be summarized.",
                     "It spans multiple sentences and goes into much detail."),
            nch = 10)

## End(Not run)
```

`TextSummaryAsBullet` *Summarize Text into Bullet Points*

Description

This function takes a text input and summarizes it into a specified number of bullet points. It can either take the selected code from RStudio or read from the clipboard. The results are output to your clipboard.

Usage

```
TextSummaryAsBullet(
  Model = "gpt-4o-mini",
  temperature = 1,
  verbose = TRUE,
  SelectedCode = TRUE
)
```

Arguments

`Model`	A string specifying the machine learning model to use for text summarization. Default is "gpt-4o-mini".
`temperature`	A numeric value between 0 and 1 indicating the randomness of the text generation. Default is 1.
`verbose`	A logical value indicating whether to print the summary. Default is FALSE.
`SelectedCode`	A logical value indicating whether to use the selected code in RStudio. Default is TRUE.

Details

Summarize Selected Text into Bullet Points

Value

The summarized text in bullet points is returned.

Author(s)

Satoshi Kume

Examples

```
## Not run:
# Option 1
# Select some text in RStudio and then run the rstudio addins
# Option 2
# Copy the text into your clipboard then execute
TextSummaryAsBullet()

## End(Not run)
```

`vision4R` *Vision API Function using OpenAI's Vision API*

Description

This function sends a local image along with a text prompt to OpenAI's GPT-4 Vision API. The function encodes the image in Base64 format and constructs a JSON payload where the user's message contains both text and an image URL (data URI). This structure mimics the provided Python code.

Usage

```
vision4R(
  image_path,
  user_prompt = "What is depicted in this image?",
  Model = "gpt-4o-mini",
  temperature = 1,
  api_key = Sys.getenv("OPENAI_API_KEY")
)
```

Arguments

`image_path`	A string specifying the path to the image file. The image format should be png or jpeg.
`user_prompt`	A string containing the text prompt. Default: "What is in this image?".
`Model`	The model to use. Defaults to "gpt-4-turbo". Allowed values: "gpt-4-turbo", "gpt-4o-mini".
`temperature`	A numeric value controlling the randomness of the output (default: 1).
`api_key`	Your OpenAI API key. Default: environment variable 'OPENAI_API_KEY'.

Value

A data frame containing the model's response.

Author(s)

Satoshi Kume

Examples

```
## Not run:
  # Example usage of the function
  api_key <- "YOUR_OPENAI_API_KEY"
  file_path <- "path/to/your/text_file.txt"
  vision4R(image_path = file_path, api_key = api_key)

## End(Not run)
```

Index