Package ‘chatgpt’

May 6, 2023

Type   Package
Title   Interface to 'ChatGPT' from R
Version 0.2.3
Maintainer Juan Cruz Rodriguez <jcrodriguez@unc.edu.ar>
Description
 'OpenAI’s 'ChatGPT' <https://chat.openai.com/> coding assistant for 'RStudio'. A set
 of functions and 'RStudio' addins that aim to help the R developer in tedious coding tasks.
License GPL (>= 3)
BugReports https://github.com/jcrodriguez1989/chatgpt/issues
Encoding UTF-8
RoxygenNote 7.2.3
Imports clipr, httr, jsonlite, miniUI, rstudioapi, shiny, utils
NeedsCompilation no
Author Juan Cruz Rodriguez [aut, cre]
Repository CRAN
Date/Publication 2023-05-05 23:00:06 UTC

R topics documented:

chatgpt-package .................................................. 2
ask_chatgpt ..................................................... 2
comment_code ................................................... 3
complete_code .................................................. 3
create_unit_tests ................................................. 4
create_variable_name ........................................... 5
document_code .................................................. 5
explain_code .................................................... 6
find_issues_in_code ............................................. 6
gpt_get_completions ............................................ 7
Description

'OpenAI's 'ChatGPT' <https://chat.openai.com/> coding assistant for 'RStudio'. A set of functions and 'RStudio' addins that aim to help the R developer in tedious coding tasks.

Author(s)

Maintainer: Juan Cruz Rodriguez <jcrodriguez@unc.edu.ar>

See Also

Useful links:


ask_chatgpt

Ask ChatGPT

Description

Note: See also 'reset_chat_session'.

Usage

ask_chatgpt(question)

Arguments

question The question to ask ChatGPT.

Value

A character value with the response generated by ChatGPT.
comment_code

Examples

```r
## Not run:
cat(ask_chatgpt("What do you think about R language?"))
```

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

---

**Description**

ChatGPT: Comment Code

**Usage**

```r
comment_code(code = clipr::read_clip(allow_non_interactive = TRUE))
```

**Arguments**

- `code`: The code to be commented by ChatGPT. If not provided, it will use what’s copied on the clipboard.

**Value**

A character value with the response generated by ChatGPT.

**Examples**

```r
## Not run:
cat(comment_code("for (i in 1:10) {
print(i ** 2)\n""))
```

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

---

**complete_code**

ChatGPT: Complete Code

**Description**

ChatGPT: Complete Code

**Usage**

```r
complete_code(code = clipr::read_clip(allow_non_interactive = TRUE))
```
Arguments

- **code**: The code to be completed by ChatGPT. If not provided, it will use what’s copied on the clipboard.

Value

A character value with the response generated by ChatGPT.

Examples

```r
## Not run:
cat(complete_code("# A function to square each element of a vector\nsquare_each <- function()\n\n## End(Not run)
```

---

**create_unit_tests**

*ChatGPT: Create Unit Tests*

Description

Create ‘testthat’ test cases for the code.

Usage

```r
create_unit_tests(code = clir::read_clip(allow_non_interactive = TRUE))
```

Arguments

- **code**: The code for which to create unit tests by ChatGPT. If not provided, it will use what’s copied on the clipboard.

Value

A character value with the response generated by ChatGPT.

Examples

```r
## Not run:
cat(create_unit_tests("squared_numbers <- function(numbers) {\n    numbers ^ 2\n}"))
## End(Not run)
```
**create_variable_name**  
*ChatGPT: Create Variable Name*

**Description**  
ChatGPT: Create Variable Name

**Usage**  
create_variable_name(code = clipr::read_clip(allow_non_interactive = TRUE))

**Arguments**

- **code**  
The code for which to give a variable name to its result. If not provided, it will use what’s copied on the clipboard.

**Value**  
A character value with the response generated by ChatGPT.

**Examples**

```r  
## Not run:  
cat(create_variable_name("sapply(1:10, function(i) i ** 2)"))  
## End(Not run)
```

---

**document_code**  
*ChatGPT: Document Code (in roxygen2 format)*

**Description**  
ChatGPT: Document Code (in roxygen2 format)

**Usage**  
document_code(code = clipr::read_clip(allow_non_interactive = TRUE))

**Arguments**

- **code**  
The code to be documented by ChatGPT. If not provided, it will use what’s copied on the clipboard.

**Value**  
A character value with the response generated by ChatGPT.
find_issues_in_code

### Examples

```r
## Not run:
cat(document_code("square_numbers <- function(numbers) numbers ** 2"))
## End(Not run)
```

---

**explain_code**  
*ChatGPT: Explain Code*

#### Description

ChatGPT: Explain Code

#### Usage

```r
explain_code(code = clipr::read_clip(allow_non_interactive = TRUE))
```

#### Arguments

- **code**
  
The code to be explained by ChatGPT. If not provided, it will use what’s copied on the clipboard.

#### Value

A character value with the response generated by ChatGPT.

---

**find_issues_in_code**  
*ChatGPT: Find Issues in Code*

#### Description

ChatGPT: Find Issues in Code

#### Usage

```r
find_issues_in_code(code = clipr::read_clip(allow_non_interactive = TRUE))
```
gpt_get_completions

Arguments

code The code to be analyzed by ChatGPT. If not provided, it will use what’s copied on the clipboard.

Value

A character value with the response generated by ChatGPT.

Examples

```r
## Not run:
cat(find_issues_in_code("i <- 0
while (i < 0) {
 i <- i - 1
}"))

## End(Not run)
```

---

gpt_get_completions  Get GPT Completions Endpoint

Description

Get GPT Completions Endpoint

Usage

```r
gpt_get_completions(
prompt,
openai_api_key = Sys.getenv("OPENAI_API_KEY"),
messages = NULL
)
```

Arguments

prompt The prompt to generate completions for.
openai_api_key OpenAI’s API key.
messages Available variable, to send the needed messages list to ChatGPT.
**optimize_code**  
*ChatGPT: Optimize Code*

**Description**
ChatGPT: Optimize Code

**Usage**
```r
optimize_code(code = clipr::read_clip(allow_non_interactive = TRUE))
```

**Arguments**
- `code`
  The code to be optimized by ChatGPT. If not provided, it will use what’s copied on the clipboard.

**Value**
A character value with the response generated by ChatGPT.

**Examples**
```r
## Not run:
cat(optimize_code("i <- 10
while (i > 0) {
  i <- i - 1
  print(i)
""))

## End(Not run)
```

**parse_response**  
*Parse OpenAI API Response*

**Description**
Takes the raw response from the OpenAI API and extracts the text content from it.

**Usage**
```r
parse_response(raw_responses)
```

**Arguments**
- `raw_responses`
  The raw response object returned by the OpenAI API.

**Value**
Returns a character vector containing the text content of the response.
**refactor_code**

**ChatGPT: Refactor Code**

**Description**

ChatGPT: Refactor Code

**Usage**

```r
defrefactor_code(code = clipr::read_clip(allow_non_interactive = TRUE))
```

**Arguments**

- `code` The code to be refactored by ChatGPT. If not provided, it will use what's copied on the clipboard.

**Value**

A character value with the response generated by ChatGPT.

**Examples**

```r
## Not run:
cat(refactor_code("i <- 10
while (i > 0) {
 i <- i - 1
 print(i)""))
## End(Not run)
```

---

**reset_chat_session**

**Reset Chat Session**

**Description**

This function is intended to be used with ‘ask_chatgpt’. If we are using ‘ask_chatgpt’ to chat with ChatGPT, and we want to start a new conversation, we must call ‘reset_chat_session’.

**Usage**

```r
defreset_chat_session(system_role = "You are a helpful assistant.")
```

**Arguments**

- `system_role` ChatGPT’s role as an AI assistant.
run_addin  

*Run a ChatGPT RStudio Addin*

**Description**

Run a ChatGPT RStudio Addin

**Usage**

`run_addin(addin_name)`

**Arguments**

*addin_name*  The name of the adding to execute.

---

run_addin_ask_chatgpt  *Ask ChatGPT*

**Description**

Opens an interactive chat session with ChatGPT

**Usage**

`run_addin_ask_chatgpt()`
Index

ask_chatgpt, 2

chatgpt (chatgpt-package), 2
chatgpt-package, 2
comment_code, 3
complete_code, 3
create_unit_tests, 4
create_variable_name, 5
document_code, 5
explain_code, 6
find_issues_in_code, 6


gpt_get_completions, 7

optimize_code, 8

parse_response, 8

refactor_code, 9
reset_chat_session, 9
run_addin, 10
run_addin_ask_chatgpt, 10