# Package 'chattr'

January 30, 2025

**Title** Interact with Large Language Models in 'RStudio'

**Version** 0.2.1

**Description** Enables user interactivity with large-language models ('LLM') inside the 'RStudio' integrated development environment (IDE). The user can interact with the model using the 'shiny' app included in this package, or directly in the 'R' console. It comes with back-ends for 'OpenAI', 'GitHub' 'Copilot', and 'LlamaGPT'.

**URL** https://github.com/mlverse/chattr, https://mlverse.github.io/chattr/

**BugReports** https://github.com/mlverse/chattr/issues

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Imports** rstudioapi, lifecycle, processx, jsonlite, config, httr2 (>= 1.1.0), purrr, rlang, bslib, shiny, clipr, callr, yaml, glue, cli, fs

**Depends** R (>= 2.10)

**Suggests** covr, knitr, rmarkdown, testthat (>= 3.0.0), shinytest2, withr

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Edgar Ruiz [aut, cre], Posit Software, PBC [cph, fnd]

**Maintainer** Edgar Ruiz <edgar@posit.co>

**Repository** CRAN

**Date/Publication** 2025-01-30 20:30:05 UTC

# Contents



---

`chattr` *Submits prompt to LLM*

---

## Description

Submits prompt to LLM

## Usage

```
chattr(prompt = NULL, preview = FALSE, prompt_build = TRUE, stream = NULL)
```

## Arguments

| | |
|---|---|
| `prompt` | Request to send to LLM. Defaults to NULL |
| `preview` | Primarily used for debugging. It indicates if it should send the prompt to the LLM (FALSE), or if it should print out the resulting prompt (TRUE) |
| `prompt_build` | Include the context and additional prompt as part of the request |
| `stream` | To output the response from the LLM as it happens, or wait until the response is complete. Defaults to TRUE. |

## Value

The output of the LLM to the console, document or script.

## Examples

```
library(chattr)
chattr_use("test")
chattr("hello")
chattr("hello", preview = TRUE)
```

## chattr_app *Starts a Shiny app interface to the LLM*

### Description

Starts a Shiny app interface to the LLM

### Usage

```
chattr_app(
  viewer = c("viewer", "dialog"),
  as_job = getOption("chattr.as_job", FALSE),
  as_job_port = getOption("shiny.port", 7788),
  as_job_host = getOption("shiny.host", "127.0.0.1")
)
```

### Arguments

| | |
|---|---|
| viewer | Specifies where the Shiny app is going to display |
| as_job | App runs as an RStudio IDE Job. Defaults to FALSE. If set to TRUE, the Shiny app will not be able to transfer the code blocks directly to the document, or console, in the IDE. |
| as_job_port | Port to use for the Shiny app. Applicable only if as_job is set to TRUE. |
| as_job_host | Host IP to use for the Shiny app. Applicable only if as_job is set to TRUE. |

### Value

A chat interface inside the 'RStudio' IDE

## chattr_defaults *Default arguments to use when making requests to the LLM*

### Description

Default arguments to use when making requests to the LLM

### Usage

```
chattr_defaults(
  type = "default",
  prompt = NULL,
  max_data_files = NULL,
  max_data_frames = NULL,
  include_doc_contents = NULL,
  include_history = NULL,
```

```
  provider = NULL,
  path = NULL,
  model = NULL,
  model_arguments = NULL,
  system_msg = NULL,
  yaml_file = "chattr.yml",
  force = FALSE,
  label = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| type | Entry point to interact with the model. Accepted values: 'notebook', chat' |
| prompt | Request to send to LLM. Defaults to NULL |
| max_data_files | Sets the maximum number of data files to send to the model. It defaults to 20. To send all, set to NULL |
| max_data_frames | Sets the maximum number of data frames loaded in the current R session to send to the model. It defaults to 20. To send all, set to NULL |
| include_doc_contents | Send the current code in the document |
| include_history | Indicates whether to include the chat history when every time a new prompt is submitted |
| provider | The name of the provider of the LLM. Today, only "openai" is is available |
| path | The location of the model. It could be an URL or a file path. |
| model | The name or path to the model to use. |
| model_arguments | Additional arguments to pass to the model as part of the request, it requires a list. Examples of arguments: temperature, top_p, max_tokens |
| system_msg | For OpenAI GPT 3.5 or above, the system message to send as part of the request |
| yaml_file | The path to a valid config YAML file that contains the defaults to use in a session |
| force | Re-process the base and any work space level file defaults |
| label | Label to display in the Shiny app, and other locations |
| ... | Additional model arguments that are not standard for all models/backends |

## Details

The idea is that because we will use addin shortcut to execute the request, all of the other arguments can be controlled via this function. By default, it will try to load defaults from a config YAML file, if none are found, then the defaults for GPT 3.5 will be used. The defaults can be modified by calling this function, even after the interactive session has started.

### Value

An 'ch_model' object that contains the current defaults that will be used to communicate with the LLM.

---

## chattr_defaults_save *Saves the current defaults in a yaml file that is compatible with the config package*

---

### Description

Saves the current defaults in a yaml file that is compatible with the config package

### Usage

```
chattr_defaults_save(path = "chattr.yml", overwrite = FALSE, type = NULL)
```

### Arguments

| | |
|---|---|
| path | Path to the file to save the configuration to |
| overwrite | Indicates to replace the file if it exists |
| type | The type of UI to save the defaults for. It defaults to NULL which will save whatever types had been used during the current R session |

### Value

It creates a YAML file with the defaults set in the current R session.

---

## chattr_test *Confirms connectivity to LLM interface*

---

### Description

Confirms connectivity to LLM interface

### Usage

```
chattr_test(defaults = NULL)

ch_test(defaults = NULL)
```

### Arguments

| | |
|---|---|
| defaults | Defaults object, generally puled from `chattr_defaults()` |

### Value

It returns console massages with the status of the test.

`chattr_use` *Sets the LLM model to use in your session*

## Description

Sets the LLM model to use in your session

## Usage

```
chattr_use(x = NULL, ...)
```

## Arguments

| | |
|---|---|
| `x` | The label of the LLM model to use, or the path of a valid YAML default file . Valid values are 'copilot', 'gpt4', 'gpt35', 'llamagpt', 'databricks-dbrx', 'databricks-meta-llama3-70b', and 'databricks-mixtral8x7b'. The value 'test' is also acceptable, but it is meant for package examples, and internal testing. |
| `...` | Default values to modify. |

## Details

If the error "No model setup found" was returned, that is because none of the expected setup for Copilot, OpenAI or LLama was automatically detected. Here is how to setup a model:

- OpenAI - The main thing `chattr` checks is the presence of the R user's OpenAI PAT (Personal Access Token). It looks for it in the 'OPENAI_API_KEY' environment variable. Get a PAT from the OpenAI website, and save it to that environment variable. Then restart R, and try again.
- GitHub Copilot - Setup GitHub Copilot in your RStudio IDE, and restart R. `chattr` will look for the default location where RStudio saves the Copilot authentication information.
- Databricks - `chattr` checks for presence of R user's Databricks host and token ('DATABRICKS_HOST' and 'DATABRICKS TOKEN' environment variables).

Use the 'CHATTR_MODEL' environment variable to set it for the R session, or create a YAML file named 'chattr.yml' in your working directory to control the model, and the defaults it will use to communicate with such model.

## Value

It returns console messages to allow the user select the model to use.

# Index