# Package ‘rollama’

April 26, 2025

**Title** Communicate with 'Ollama' to Run Large Language Models Locally

**Version** 0.2.2

**Description** Wraps the 'Ollama' <https://ollama.com> API, which can be used to communicate with generative large language models locally.

**License** GPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Imports** callr, cli, dplyr, httr2, jsonlite, methods, prettyunits, purrr, rlang, tibble, withr

**Suggests** base64enc, covr, glue, knitr, rmarkdown, spelling, testthat (>= 3.0.0)

**Depends** R (>= 4.1.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Language** en-US

**URL** https://jbgruber.github.io/rollama/,

https://github.com/JBGruber/rollama

**BugReports** https://github.com/JBGruber/rollama/issues

**NeedsCompilation** no

**Author** Johannes B. Gruber [aut, cre] (<https://orcid.org/0000-0001-9177-1772>), Maximilian Weber [aut, ctb] (<https://orcid.org/0000-0002-1174-449X>)

**Maintainer** Johannes B. Gruber <JohannesB.Gruber@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-04-25 23:20:02 UTC

# Contents

chat_history . . . 2
check_model_installed . . . 2
create_model . . . 3
embed_text . . . 4
list_models . . . 5
make_query . . . 5
ping_ollama . . . 7
pull_model . . . 7
query . . . 8
rollama-options . . . 12

**Index** **13**

---

`chat_history` *Handle conversations*

---

## Description

Shows and deletes (`new_chat`) the local prompt and response history to start a new conversation.

## Usage

```
chat_history()

new_chat()
```

## Value

chat_history: tibble with chat history

new_chat: Does not return a value

---

`check_model_installed` *Check if one or several models are installed on the server*

---

## Description

Check if one or several models are installed on the server

## Usage

```
check_model_installed(
  model,
  check_only = FALSE,
  auto_pull = FALSE,
  server = getOption("rollama_server", default = "http://localhost:11434")
)
```

### Arguments

| | |
|---|---|
| `model` | names of one or several models as character vector. |
| `check_only` | only return TRUE/FALSE and don't download models. |
| `auto_pull` | if FALSE, the default, asks before downloading models. |
| `server` | URL to one or several Ollama servers (not the API). Defaults to "http://localhost:11434". |

### Value

invisible TRUE/FALSE

## `create_model` *Create a model from a Modelfile*

### Description

Create a model from a Modelfile

### Usage

```
create_model(model, modelfile, server = NULL)
```

### Arguments

| | |
|---|---|
| `model` | name of the model to create |
| `modelfile` | either a path to a model file to be read or the contents of the model file as a character vector. |
| `server` | URL to one or several Ollama servers (not the API). Defaults to "http://localhost:11434". |

### Details

Custom models are the way to save your system message and model parameters in a dedicated shareable way. If you use `show_model()`, you can look at the configuration of a model in the column modelfile. To get more information and a list of valid parameters, check out `https://github.com/ollama/ollama/blob/main/docs/modelfile.md`. Most options are also available through the `query` and `chat` functions, yet are not persistent over sessions.

### Value

Nothing. Called to create a model on the Ollama server.

### Examples

```
modelfile <- system.file("extdata", "modelfile.txt", package = "rollama")
## Not run: create_model("mario", modelfile)
modelfile <- "FROM llama3.1\nSYSTEM You are mario from Super Mario Bros."
## Not run: create_model("mario", modelfile)
```

## embed_text *Generate Embeddings*

### Description

Generate Embeddings

### Usage

```
embed_text(
  text,
  model = NULL,
  server = NULL,
  model_params = NULL,
  verbose = getOption("rollama_verbose", default = interactive())
)
```

### Arguments

| | |
|---|---|
| `text` | text vector to generate embeddings for. |
| `model` | which model to use. See `https://ollama.com/library` for options. Default is "llama3.1". Set option(rollama_model = "modelname") to change default for the current session. See pull_model for more details. |
| `server` | URL to one or several Ollama servers (not the API). Defaults to "http://localhost:11434". |
| `model_params` | a named list of additional model parameters listed in the documentation for the Modelfile. |
| `verbose` | Whether to print status messages to the Console (`TRUE`/`FALSE`). The default is to have status messages in interactive sessions. Can be changed with `options(rollama_verbose = FALSE)`. |

### Value

a tibble with embeddings.

### Examples

```
## Not run:
embed_text(c(
  "Here is an article about llamas...",
  "R is a language and environment for statistical computing and graphics."))

## End(Not run)
```

## `list_models` *List models that are available locally.*

### Description

List models that are available locally.

### Usage

```
list_models(server = NULL)
```

### Arguments

| | |
|---|---|
| `server` | URL to one or several Ollama servers (not the API). Defaults to "http://localhost:11434". |

### Value

a tibble of installed models

## `make_query` *Generate and format queries for a language model*

### Description

`make_query` generates structured input for a language model, including system prompt, user messages, and optional examples (assistant answers).

### Usage

```
make_query(
  text,
  prompt,
  template = "{prefix}{text}\n{prompt}\n{suffix}",
  system = NULL,
  prefix = NULL,
  suffix = NULL,
  examples = NULL
)
```

## Arguments

| | |
|---|---|
| `text` | A character vector of texts to be annotated. |
| `prompt` | A string defining the main task or question to be passed to the language model. |
| `template` | A string template for formatting user queries, containing placeholders like `{text}`, `{prefix}`, and `{suffix}`. |
| `system` | An optional string to specify a system prompt. |
| `prefix` | A prefix string to prepend to each user query. |
| `suffix` | A suffix string to append to each user query. |
| `examples` | A `tibble` with columns `text` and `answer`, representing example user messages and corresponding assistant responses. |

## Details

The function supports the inclusion of examples, which are dynamically added to the structured input. Each example follows the same format as the primary user query.

## Value

A list of tibbles, one for each input `text`, containing structured rows for system messages, user messages, and assistant responses.

## Examples

```
template <- "{prefix}{text}\n\n{prompt}{suffix}"
examples <- tibble::tribble(
  ~text, ~answer,
  "This movie was amazing, with great acting and story.", "positive",
  "The film was okay, but not particularly memorable.", "neutral",
  "I found this movie boring and poorly made.", "negative"
)
queries <- make_query(
  text = c("A stunning visual spectacle.", "Predictable but well-acted."),
  prompt = "Classify sentiment as positive, neutral, or negative.",
  template = template,
  system = "Provide a sentiment classification.",
  prefix = "Review: ",
  suffix = " Please classify.",
  examples = examples
)
print(queries)
if (ping_ollama()) { # only run this example when Ollama is running
  query(queries, screen = TRUE, output = "text")
}
```

## `ping_ollama` *Ping server to see if Ollama is reachable*

### Description

Ping server to see if Ollama is reachable

### Usage

```
ping_ollama(server = NULL, silent = FALSE, version = FALSE)
```

### Arguments

| | |
|---|---|
| `server` | URL to one or several Ollama servers (not the API). Defaults to "http://localhost:11434". |
| `silent` | suppress warnings and status (only return `TRUE`/`FALSE`). |
| `version` | return version instead of `TRUE`. |

### Value

TRUE if server is running

## `pull_model` *Pull, show and delete models*

### Description

Pull, show and delete models

### Usage

```
pull_model(
  model = NULL,
  server = NULL,
  insecure = FALSE,
  verbose = getOption("rollama_verbose", default = interactive())
)

show_model(model = NULL, server = NULL)

delete_model(model, server = NULL)

copy_model(model, destination = paste0(model, "-copy"), server = NULL)
```

### Arguments

| | |
|---|---|
| `model` | name of the model(s). Defaults to "llama3.1" when `NULL` (except in `delete_model`). |
| `server` | URL to one or several Ollama servers (not the API). Defaults to "http://localhost:11434". |
| `insecure` | allow insecure connections to the library. Only use this if you are pulling from your own library during development. |
| `verbose` | Whether to print status messages to the Console. Either `TRUE`/`FALSE` or see httr2::progress_bars. The default is to have status messages in interactive sessions. Can be changed with `options(rollama_verbose = FALSE)`. |
| `destination` | name of the copied model. |

### Details

- `pull_model()`: downloads model
- `show_model()`: displays information about a local model
- `copy_model()`: creates a model with another name from an existing model
- `delete_model()`: deletes local model

**Model names**: Model names follow a model:tag format, where model can have an optional namespace such as example/model. Some examples are orca-mini:3b-q4_1 and llama3.1:70b. The tag is optional and, if not provided, will default to latest. The tag is used to identify a specific version.

### Value

(invisible) a tibble with information about the model (except in `delete_model`)

### Examples

```
## Not run:
# download a model and save information in an object
model_info <- pull_model("mixtral")
# after you pull, you can get the same information with:
model_info <- show_model("mixtral")
# pulling models from Hugging Face Hub is also possible
pull_model("https://huggingface.co/oxyapi/oxy-1-small-GGUF:Q2_K")

## End(Not run)
```

## query *Chat with a LLM through Ollama*

### Description

Chat with a LLM through Ollama

## Usage

```
query(
  q,
  model = NULL,
  screen = TRUE,
  server = NULL,
  images = NULL,
  model_params = NULL,
 output = c("response", "text", "list", "data.frame", "httr2_response", "httr2_request"),
  format = NULL,
  template = NULL,
  verbose = getOption("rollama_verbose", default = interactive())
)

chat(
  q,
  model = NULL,
  screen = TRUE,
  server = NULL,
  images = NULL,
  model_params = NULL,
  template = NULL,
  verbose = getOption("rollama_verbose", default = interactive())
)
```

## Arguments

| | |
|---|---|
| `q` | the question as a character string or a conversation object. |
| `model` | which model(s) to use. See `https://ollama.com/library` for options. Default is "llama3.1". Set `option(rollama_model = "modelname")` to change default for the current session. See pull_model for more details. |
| `screen` | Logical. Should the answer be printed to the screen. |
| `server` | URL to one or several Ollama servers (not the API). Defaults to "http://localhost:11434". |
| `images` | path(s) to images (for multimodal models such as llava). |
| `model_params` | a named list of additional model parameters listed in the documentation for the Modelfile such as temperature. Use a seed and set the temperature to zero to get reproducible results (see examples). |
| `output` | what the function should return. Possible values are "response", "text", "list", "data.frame", "httr2_response" or "httr2_request" or a function see details. |
| `format` | the format to return a response in. Currently the only accepted value is `"json"`. |
| `template` | the prompt template to use (overrides what is defined in the Modelfile). |
| `verbose` | Whether to print status messages to the Console. Either `TRUE`/`FALSE` or see httr2::progress_bars. The default is to have status messages in interactive sessions. Can be changed with `options(rollama_verbose = FALSE)`. |

## Details

query sends a single question to the API, without knowledge about previous questions (only the config message is relevant). chat treats new messages as part of the same conversation until new_chat is called.

To make the output reproducible, you can set a seed with options(rollama_seed = 42). As long as the seed stays the same, the models will give the same answer, changing the seed leads to a different answer.

For the output of query, there are a couple of options:

- response: the response of the Ollama server
- text: only the answer as a character vector
- data.frame: a data.frame containing model and response
- list: a list containing the prompt to Ollama and the response
- httr2_response: the response of the Ollama server including HTML headers in the httr2 response format
- httr2_request: httr2_request objects in a list, in case you want to run them with httr2::req_perform(), httr2::req_perform_sequential(), or httr2::req_perform_parallel() yourself.
- a custom function that takes the httr2_response(s) from the Ollama server as an input.

## Value

list of objects set in output parameter.

## Examples

```
# ask a single question
query("why is the sky blue?")

# hold a conversation
chat("why is the sky blue?")
chat("and how do you know that?")

# save the response to an object and extract the answer
resp <- query(q = "why is the sky blue?")
answer <- resp[[1]]$message$content

# or just get the answer directly
answer <- query(q = "why is the sky blue?", output = "text")

# besides the other output options, you can also supply a custom function
query_duration <- function(resp) {
nanosec <- purrr::map(resp, httr2::resp_body_json) |>
  purrr::map_dbl("total_duration")
  round(nanosec * 1e-9, digits = 2)
}
# this function only returns the number of seconds a request took
res <- query("why is the sky blue?", output = query_duration)
res
```

```
# ask question about images (to a multimodal model)
images <- c("https://avatars.githubusercontent.com/u/23524101?v=4", # remote
            "/path/to/your/image.jpg") # or local images supported
query(q = "describe these images",
      model = "llava",
      images = images[1]) # just using the first path as the second is not real

# set custom options for the model at runtime (rather than in create_model())
query("why is the sky blue?",
      model_params = list(
        num_keep = 5,
        seed = 42,
        num_predict = 100,
        top_k = 20,
        top_p = 0.9,
        min_p = 0.0,
        tfs_z = 0.5,
        typical_p = 0.7,
        repeat_last_n = 33,
        temperature = 0.8,
        repeat_penalty = 1.2,
        presence_penalty = 1.5,
        frequency_penalty = 1.0,
        mirostat = 1,
        mirostat_tau = 0.8,
        mirostat_eta = 0.6,
        penalize_newline = TRUE,
        numa = FALSE,
        num_ctx = 1024,
        num_batch = 2,
        num_gpu = 0,
        main_gpu = 0,
        low_vram = FALSE,
        vocab_only = FALSE,
        use_mmap = TRUE,
        use_mlock = FALSE,
        num_thread = 8
      ))

# use a seed to get reproducible results
query("why is the sky blue?", model_params = list(seed = 42))

# to set a seed for the whole session you can use
options(rollama_seed = 42)

# this might be interesting if you want to turn off the GPU and load the
# model into the system memory (slower, but most people have more RAM than
# VRAM, which might be interesting for larger models)
query("why is the sky blue?",
       model_params = list(num_gpu = 0))

# Asking the same question to multiple models is also supported
```

```
query("why is the sky blue?", model = c("llama3.1", "orca-mini"))

# And if you have multiple Ollama servers in your network, you can send
# requests to them in parallel
if (ping_ollama(c("http://localhost:11434/",
                  "http://192.168.2.45:11434/"))) { # check if servers are running
  query("why is the sky blue?", model = c("llama3.1", "orca-mini"),
        server = c("http://localhost:11434/",
                   "http://192.168.2.45:11434/"))
}
```

---

## rollama-options *rollama Options*

---

### Description

The behaviour of `rollama` can be controlled through `options()`. Specifically, the options below can be set.

### Details

**rollama_server** This controls the default server where Ollama is expected to run. It assumes that you are running Ollama locally in a Docker container.

`"http://localhost:11434"`

**default** **rollama_model** The default model is llama3.1, which is a good overall option with reasonable performance and size for most tasks. You can change the model in each function call or globally with this option.

`"llama3.1"`

**default** **rollama_verbose** Whether the package tells users what is going on, e.g., showing a spinner while the models are thinking or showing the download speed while pulling models. Since this adds some complexity to the code, you might want to disable it when you get errors (it won't fix the error, but you get a better error trace).

`TRUE`

**default** **rollama_config** The default configuration or system message. If NULL, the system message defined in the used model is employed.

None

**default** **rollama_seed** As long as the seed stays the same, the models will give the same answer, changing the seed leads to a different answer. Per default, no seed is set and each call to `query()` or `chat()` will give you a different answer.

None

### Examples

**default:** `options(rollama_config = "You make answers understandable to a 5 year old")`

# Index

chat *(*query*)*, 8
chat_history, 2
check_model_installed, 2
copy_model *(*pull_model*)*, 7
create_model, 3

delete_model *(*pull_model*)*, 7

embed_text, 4

httr2::progress_bars, *8*, *9*
httr2::req_perform(), *10*
httr2::req_perform_parallel(), *10*
httr2::req_perform_sequential(), *10*

list_models, 5

make_query, 5

new_chat, *10*
new_chat *(*chat_history*)*, 2

ping_ollama, 7
pull_model, *4*, 7, *9*

query, 8

rollama-options, 12

show_model *(*pull_model*)*, 7