

# Package ‘ollamar’

May 9, 2026

**Title** 'Ollama' Language Models

**Version** 1.2.2

**Description**

An interface to easily run local language models with 'Ollama' <<https://ollama.com>> server and API endpoints (see <<https://github.com/ollama/ollama/blob/main/docs/api.md>> for details). It lets you run open-source large language models locally on your machine.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Imports** base64enc, crayon, glue, httr2, jsonlite, tibble

**BugReports** <https://github.com/hauselin/ollama-r/issues>

**URL** <https://hauselin.github.io/ollama-r/>,  
<https://github.com/hauselin/ollama-r>

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Hause Lin [aut, cre, cph] (ORCID:  
<<https://orcid.org/0000-0003-4590-7039>>),  
Tawab Safi [aut, ctb] (ORCID: <<https://orcid.org/0009-0000-5659-9890>>)

**Maintainer** Hause Lin <[hauselin@gmail.com](mailto:hauselin@gmail.com)>

**Repository** CRAN

**Date/Publication** 2025-01-08 05:40:09 UTC

## Contents

append_message . . . . .	2
chat . . . . .	3

check_options . . . . .	5
check_option_valid . . . . .	5
copy . . . . .	6
create . . . . .	6
create_message . . . . .	7
create_messages . . . . .	8
create_request . . . . .	9
delete . . . . .	9
delete_message . . . . .	10
embed . . . . .	11
embeddings . . . . .	12
encode_images_in_messages . . . . .	13
generate . . . . .	13
image_encode_base64 . . . . .	15
insert_message . . . . .	16
list_models . . . . .	16
model_avail . . . . .	17
model_options . . . . .	18
ohelp . . . . .	18
package_config . . . . .	19
prepend_message . . . . .	19
ps . . . . .	20
pull . . . . .	20
push . . . . .	21
resp_process . . . . .	22
search_options . . . . .	23
show . . . . .	24
test_connection . . . . .	25
validate_message . . . . .	25
validate_messages . . . . .	26
validate_options . . . . .	26
<b>Index</b>	<b>28</b>

---

append_message	<i>Append message to a list</i>
----------------	---------------------------------

---

### Description

Appends a message (add to end of a list) to a list of messages. The role and content will be converted to a list and appended to the input list.

### Usage

```
append_message(content, role = "user", x = NULL, ...)
```

**Arguments**

content	The content of the message.
role	The role of the message. Can be "user", "system", "assistant". Default is "user".
x	A list of messages. Default is NULL.
...	Additional arguments such as images.

**Value**

A list of messages with the new message appended.

**Examples**

```
append_message("user", "Hello")
append_message("system", "Always respond nicely")
```

---

chat	<i>Generate a chat completion with message history</i>
------	--

---

**Description**

Generate a chat completion with message history

**Usage**

```
chat(
  model,
  messages,
  tools = list(),
  stream = FALSE,
  format = list(),
  keep_alive = "5m",
  output = c("resp", "jsonlist", "raw", "df", "text", "req", "tools", "structured"),
  endpoint = "/api/chat",
  host = NULL,
  ...
)
```

**Arguments**

model	A character string of the model name such as "llama3".
messages	A list with list of messages for the model (see examples below).
tools	Tools for the model to use if supported. Requires stream = FALSE. Default is an empty list.
stream	Enable response streaming. Default is FALSE.
format	Format to return a response in. Format can be json/list (structured response).

keep_alive	The duration to keep the connection alive. Default is "5m".
output	The output format. Default is "resp". Other options are "jsonlist", "raw", "df", "text", "req" (httr2_request object), "tools" (tool calling), "structured" (structured output)
endpoint	The endpoint to chat with the model. Default is "/api/chat".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.
...	Additional options to pass to the model.

### Value

A response in the format specified in the output parameter.

### References

[API documentation](#)

### Examples

```
# one message
messages <- list(
  list(role = "user", content = "How are you doing?")
)
chat("llama3", messages) # returns response by default
chat("llama3", messages, output = "text") # returns text/vector
chat("llama3", messages, temperature = 2.8) # additional options
chat("llama3", messages, stream = TRUE) # stream response
chat("llama3", messages, output = "df", stream = TRUE) # stream and return dataframe

# multiple messages
messages <- list(
  list(role = "user", content = "Hello!"),
  list(role = "assistant", content = "Hi! How are you?"),
  list(role = "user", content = "Who is the prime minister of the uk?"),
  list(role = "assistant", content = "Rishi Sunak"),
  list(role = "user", content = "List all the previous messages.")
)
chat("llama3", messages, stream = TRUE)

# image
image_path <- file.path(system.file("extdata", package = "ollamar"), "image1.png")
messages <- list(
  list(role = "user", content = "What is in the image?", images = image_path)
)
chat("benzie/llava-phi-3", messages, output = 'text')
```

---

check_options	<i>Check if a vector of options are valid</i>
---------------	---

---

**Description**

Check if a vector of options are valid

**Usage**

```
check_options(opts = NULL)
```

**Arguments**

opts                   A vector of options to check.

**Value**

Returns a list with two elements: valid\_options and invalid\_options.

**Examples**

```
check_options(c("mirostat", "invalid_option"))
check_options(c("mirostat", "num_predict"))
```

---

check_option_valid	<i>Check if an option is valid</i>
--------------------	------------------------------------

---

**Description**

Check if an option is valid

**Usage**

```
check_option_valid(opt)
```

**Arguments**

opt                    An option (character) to check.

**Value**

Returns TRUE if the option is valid, FALSE otherwise.

**Examples**

```
check_option_valid("mirostat")
check_option_valid("invalid_option")
```

---

copy	<i>Copy a model</i>
------	---------------------

---

### Description

Creates a model with another name from an existing model.

### Usage

```
copy(source, destination, endpoint = "/api/copy", host = NULL)
```

### Arguments

source	The name of the model to copy.
destination	The name for the new model.
endpoint	The endpoint to copy the model. Default is "/api/copy".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

### Value

A htr2 response object.

### References

[API documentation](#)

### Examples

```
copy("llama3", "llama3_copy")
delete("llama3_copy") # delete the model was just got copied
```

---

create	<i>Create a model from a Modelfile</i>
--------	--

---

### Description

It is recommended to set `modelfile` to the content of the Modelfile rather than just set path.

**Usage**

```
create(
  name,
  modelfile = NULL,
  stream = FALSE,
  path = NULL,
  endpoint = "/api/create",
  host = NULL
)
```

**Arguments**

name	Name of the model to create.
modelfile	Contents of the Modelfile as character string. Default is NULL.
stream	Enable response streaming. Default is FALSE.
path	The path to the Modelfile. Default is NULL.
endpoint	The endpoint to create the model. Default is "/api/create".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

**Value**

A response in the format specified in the output parameter.

**References**

[API documentation](#)

**Examples**

```
create("mario", "FROM llama3\nSYSTEM You are mario from Super Mario Bros.")
generate("mario", "who are you?", output = "text") # model should say it's Mario
delete("mario") # delete the model created above
```

---

create_message	<i>Create a message</i>
----------------	-------------------------

---

**Description**

Create a message

**Usage**

```
create_message(content, role = "user", ...)
```

**Arguments**

content            The content of the message.  
role                The role of the message. Can be "user", "system", "assistant". Default is "user".  
...                 Additional arguments such as images.

**Value**

A list of messages.

**Examples**

```
create_message("Hello", "user")  
create_message("Always respond nicely", "system")  
create_message("I am here to help", "assistant")
```

---

create_messages	<i>Create a list of messages</i>
-----------------	----------------------------------

---

**Description**

Create messages for chat() function.

**Usage**

```
create_messages(...)
```

**Arguments**

...                 A list of messages, each of list class.

**Value**

A list of messages, each of list class.

**Examples**

```
messages <- create_messages(  
  create_message("be nice", "system"),  
  create_message("tell me a 3-word joke")  
)  
  
messages <- create_messages(  
  list(role = "system", content = "be nice"),  
  list(role = "user", content = "tell me a 3-word joke")  
)
```

---

create_request	<i>Create a htr2 request object</i>
----------------	-------------------------------------

---

**Description**

Creates a htr2 request object with base URL, headers and endpoint. Used by other functions in the package and not intended to be used directly.

**Usage**

```
create_request(endpoint, host = NULL)
```

**Arguments**

endpoint	The endpoint to create the request
host	The base URL to use. Default is NULL, which uses http://127.0.0.1:11434

**Value**

A htr2 request object.

**Examples**

```
create_request("/api/tags")
create_request("/api/chat")
create_request("/api/embeddings")
```

---

delete	<i>Delete a model and its data</i>
--------	------------------------------------

---

**Description**

Delete a model from your local machine that you downloaded using the pull() function. To see which models are available, use the list\_models() function.

**Usage**

```
delete(name, endpoint = "/api/delete", host = NULL)
```

**Arguments**

name	A character string of the model name such as "llama3".
endpoint	The endpoint to delete the model. Default is "/api/delete".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

**Value**

A httr2 response object.

**References**

[API documentation](#)

**Examples**

```
## Not run:  
delete("llama3")  
  
## End(Not run)
```

---

delete\_message

*Delete a message in a specified position from a list*

---

**Description**

Delete a message using positive or negative positions/indices. Negative positions/indices can be used to refer to elements/messages from the end of the sequence.

**Usage**

```
delete_message(x, position = -1)
```

**Arguments**

x                    A list of messages.  
position            The position of the message to delete.

**Value**

A list of messages with the message at the specified position removed.

**Examples**

```
messages <- list(  
  list(role = "system", content = "Be friendly"),  
  list(role = "user", content = "How are you?")  
)  
delete_message(messages, 1) # delete first message  
delete_message(messages, -2) # same as above (delete first message)  
delete_message(messages, 2) # delete second message  
delete_message(messages, -1) # same as above (delete second message)
```

---

embed	<i>Generate embedding for inputs</i>
-------	--------------------------------------

---

## Description

Supercedes the embeddings() function.

## Usage

```
embed(  
  model,  
  input,  
  truncate = TRUE,  
  normalize = TRUE,  
  keep_alive = "5m",  
  endpoint = "/api/embed",  
  host = NULL,  
  ...  
)
```

## Arguments

model	A character string of the model name such as "llama3".
input	A vector of characters that you want to get the embeddings for.
truncate	Truncates the end of each input to fit within context length. Returns error if FALSE and context length is exceeded. Defaults to TRUE.
normalize	Normalize the vector to length 1. Default is TRUE.
keep_alive	The time to keep the connection alive. Default is "5m" (5 minutes).
endpoint	The endpoint to get the vector embedding. Default is "/api/embeddings".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.
...	Additional options to pass to the model.

## Value

A numeric matrix of the embedding. Each column is the embedding for one input.

## References

[API documentation](#)

**Examples**

```
embed("nomic-embed-text:latest", "The quick brown fox jumps over the lazy dog.")
# pass multiple inputs
embed("nomic-embed-text:latest", c("Good bye", "Bye", "See you.))
# pass model options to the model
embed("nomic-embed-text:latest", "Hello!", temperature = 0.1, num_predict = 3)
```

---

embeddings	<i>Generate embeddings for a single prompt - deprecated in favor of embed()</i>
------------	---

---

**Description**

This function will be deprecated over time and has been superceded by `embed()`. See `embed()` for more details.

**Usage**

```
embeddings(
  model,
  prompt,
  normalize = TRUE,
  keep_alive = "5m",
  endpoint = "/api/embeddings",
  host = NULL,
  ...
)
```

**Arguments**

<code>model</code>	A character string of the model name such as "llama3".
<code>prompt</code>	A character string of the prompt that you want to get the vector embedding for.
<code>normalize</code>	Normalize the vector to length 1. Default is TRUE.
<code>keep_alive</code>	The time to keep the connection alive. Default is "5m" (5 minutes).
<code>endpoint</code>	The endpoint to get the vector embedding. Default is "/api/embeddings".
<code>host</code>	The base URL to use. Default is NULL, which uses Ollama's default base URL.
<code>...</code>	Additional options to pass to the model.

**Value**

A numeric vector of the embedding.

**References**

[API documentation](#)

**Examples**

```
embeddings("nomic-embed-text:latest", "The quick brown fox jumps over the lazy dog.")  
# pass model options to the model  
embeddings("nomic-embed-text:latest", "Hello!", temperature = 0.1, num_predict = 3)
```

---

encode\_images\_in\_messages

*Encode images in messages to base64 format*

---

**Description**

Encode images in messages to base64 format

**Usage**

```
encode_images_in_messages(messages)
```

**Arguments**

messages            A list of messages, each of list class. Generally used in the chat() function.

**Value**

A list of messages with images encoded in base64 format.

**Examples**

```
image <- file.path(system.file("extdata", package = "ollamar"), "image1.png")  
message <- create_message(content = "what is in the image?", images = image)  
message_updated <- encode_images_in_messages(message)
```

---

generate

*Generate a response for a given prompt*

---

**Description**

Generate a response for a given prompt

**Usage**

```

generate(
  model,
  prompt,
  suffix = "",
  images = "",
  format = list(),
  system = "",
  template = "",
  context = list(),
  stream = FALSE,
  raw = FALSE,
  keep_alive = "5m",
  output = c("resp", "jsonlist", "raw", "df", "text", "req", "structured"),
  endpoint = "/api/generate",
  host = NULL,
  ...
)

```

**Arguments**

model	A character string of the model name such as "llama3".
prompt	A character string of the prompt like "The sky is..."
suffix	A character string after the model response. Default is "".
images	A path to an image file to include in the prompt. Default is "".
format	Format to return a response in. Format can be json/list (structured response).
system	A character string of the system prompt (overrides what is defined in the Modelfile). Default is "".
template	A character string of the prompt template (overrides what is defined in the Modelfile). Default is "".
context	A list of context from a previous response to include previous conversation in the prompt. Default is an empty list.
stream	Enable response streaming. Default is FALSE.
raw	If TRUE, no formatting will be applied to the prompt. You may choose to use the raw parameter if you are specifying a full templated prompt in your request to the API. Default is FALSE.
keep_alive	The time to keep the connection alive. Default is "5m" (5 minutes).
output	A character vector of the output format. Default is "resp". Options are "resp", "jsonlist", "raw", "df", "text", "req" (httr2_request object).
endpoint	The endpoint to generate the completion. Default is "/api/generate".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.
...	Additional options to pass to the model.

**Value**

A response in the format specified in the output parameter.

**References**

[API documentation](#)

**Examples**

```
# text prompt
generate("llama3", "The sky is...", stream = FALSE, output = "df")
# stream and increase temperature
generate("llama3", "The sky is...", stream = TRUE, output = "text", temperature = 2.0)

# image prompt
# something like "image1.png"
image_path <- file.path(system.file("extdata", package = "ollamar"), "image1.png")
# use vision or multimodal model such as https://ollama.com/benzie/llava-phi-3
generate("benzie/llava-phi-3:latest", "What is in the image?", images = image_path, output = "text")
```

---

image\_encode\_base64    *Read image file and encode it to base64*

---

**Description**

Read image file and encode it to base64

**Usage**

```
image_encode_base64(image_path)
```

**Arguments**

image\_path    The path to the image file.

**Value**

A base64 encoded string.

**Examples**

```
image_path <- file.path(system.file("extdata", package = "ollamar"), "image1.png")
substr(image_encode_base64(image_path), 1, 5) # truncate output
```

---

insert_message	<i>Insert message into a list at a specified position</i>
----------------	---

---

**Description**

Inserts a message at a specified position in a list of messages. The role and content are converted to a list and inserted into the input list at the given position.

**Usage**

```
insert_message(content, role = "user", x = NULL, position = -1, ...)
```

**Arguments**

content	The content of the message.
role	The role of the message. Can be "user", "system", "assistant". Default is "user".
x	A list of messages. Default is NULL.
position	The position at which to insert the new message. Default is -1 (end of list).
...	Additional arguments such as images.

**Value**

A list of messages with the new message inserted at the specified position.

**Examples**

```
messages <- list(
  list(role = "system", content = "Be friendly"),
  list(role = "user", content = "How are you?")
)
insert_message("INSERT MESSAGE AT THE END", "user", messages)
insert_message("INSERT MESSAGE AT THE BEGINNING", "user", messages, 2)
```

---

list_models	<i>List models that are available locally</i>
-------------	---

---

**Description**

List models that are available locally

**Usage**

```
list_models(
  output = c("df", "resp", "jsonlist", "raw", "text"),
  endpoint = "/api/tags",
  host = NULL
)
```

**Arguments**

output	The output format. Default is "df". Other options are "resp", "jsonlist", "raw", "text".
endpoint	The endpoint to get the models. Default is "/api/tags".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

**Value**

A response in the format specified in the output parameter.

**References**

[API documentation](#)

**Examples**

```
list_models() # returns dataframe
list_models("df") # returns dataframe
list_models("resp") # httr2 response object
list_models("jsonlist")
list_models("raw")
```

---

model\_avail

*Check if model is available locally*

---

**Description**

Check if model is available locally

**Usage**

```
model_avail(model)
```

**Arguments**

model	A character string of the model name such as "llama3".
-------	--

**Value**

A logical value indicating if the model exists.

**Examples**

```
model_avail("codegemma:7b")
model_avail("abc")
model_avail("llama3")
```

---

model_options	<i>Model options</i>
---------------	----------------------

---

**Description**

Model options

**Usage**

```
model_options
```

**Format**

An object of class list of length 13.

---

ohelp	<i>Chat with a model in real-time in R console</i>
-------	--

---

**Description**

Chat with a model in real-time in R console

**Usage**

```
ohelp(model = "codegemma:7b", ...)
```

**Arguments**

model	A character string of the model name such as "llama3". Defaults to "codegemma:7b" which is a decent coding model as of 2024-07-27.
...	Additional options. No options are currently available at this time.

**Value**

Does not return anything. It prints the conversation in the console.

**Examples**

```
ohelp(first_prompt = "quit")  
# regular usage: ohelp()
```

---

package_config	<i>Package configuration</i>
----------------	------------------------------

---

**Description**

Package configuration

**Usage**

package\_config

**Format**

An object of class list of length 3.

---

prepend_message	<i>Prepend message to a list</i>
-----------------	----------------------------------

---

**Description**

Prepends a message (add to beginning of a list) to a list of messages. The role and content will be converted to a list and prepended to the input list.

**Usage**

```
prepend_message(content, role = "user", x = NULL, ...)
```

**Arguments**

content	The content of the message.
role	The role of the message. Can be "user", "system", "assistant".
x	A list of messages. Default is NULL.
...	Additional arguments such as images.

**Value**

A list of messages with the new message prepended.

**Examples**

```
prepend_message("user", "Hello")
prepend_message("system", "Always respond nicely")
```

---

ps *List models that are currently loaded into memory*

---

### Description

List models that are currently loaded into memory

### Usage

```
ps(  
  output = c("df", "resp", "jsonlist", "raw", "text"),  
  endpoint = "/api/ps",  
  host = NULL  
)
```

### Arguments

output	The output format. Default is "df". Supported formats are "df", "resp", "jsonlist", "raw", and "text".
endpoint	The endpoint to list the running models. Default is "/api/ps".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

### Value

A response in the format specified in the output parameter.

### References

[API documentation](#)

### Examples

```
ps("text")
```

---

pull *Pull/download a model from the Ollama library*

---

### Description

See <https://ollama.com/library> for a list of available models. Use the `list_models()` function to get the list of models already downloaded/installed on your machine. Cancelled pulls are resumed from where they left off, and multiple calls will share the same download progress.

**Usage**

```
pull(  
  name,  
  stream = FALSE,  
  insecure = FALSE,  
  endpoint = "/api/pull",  
  host = NULL  
)
```

**Arguments**

name	A character string of the model name to download/pull, such as "llama3".
stream	Enable response streaming. Default is FALSE.
insecure	Allow insecure connections Only use this if you are pulling from your own library during development. Default is FALSE.
endpoint	The endpoint to pull the model. Default is "/api/pull".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

**Value**

A htr2 response object.

**References**

[API documentation](#)

**Examples**

```
pull("llama3")  
pull("all-minilm", stream = FALSE)
```

---

push

*Push or upload a model to a model library*

---

**Description**

Push or upload a model to an Ollama model library. Requires registering for ollama.ai and adding a public key first.

**Usage**

```
push(
  name,
  insecure = FALSE,
  stream = FALSE,
  output = c("resp", "jsonlist", "raw", "text", "df"),
  endpoint = "/api/push",
  host = NULL
)
```

**Arguments**

name	A character string of the model name to upload, in the form of <namespace>/<model>:<tag>
insecure	Allow insecure connections. Only use this if you are pushing to your own library during development. Default is FALSE.
stream	Enable response streaming. Default is FALSE.
output	The output format. Default is "resp". Other options are "jsonlist", "raw", "text", and "df".
endpoint	The endpoint to push the model. Default is "/api/push".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

**Value**

A htr2 response object.

**References**

[API documentation](#)

**Examples**

```
push("mattw/pygmalion:latest")
```

---

resp\_process

*Process htr2 response object*

---

**Description**

Process htr2 response object

**Usage**

```
resp_process(
  resp,
  output = c("df", "jsonlist", "raw", "resp", "text", "tools")
)
```

**Arguments**

resp	A httr2 response object.
output	The output format. Default is "df". Other options are "jsonlist", "raw", "resp" (httr2 response object), "text", "tools" (tool_calls), "structured" (structured output).

**Value**

A data frame, json list, raw or httr2 response object.

**Examples**

```
resp <- list_models("resp")
resp_process(resp, "df") # parse response to dataframe/tibble
resp_process(resp, "jsonlist") # parse response to list
resp_process(resp, "raw") # parse response to raw string
resp_process(resp, "text") # return text/character vector
resp_process(resp, "tools") # return tool_calls
```

---

search_options	<i>Search for options based on a query</i>
----------------	--

---

**Description**

Search for options based on a query

**Usage**

```
search_options(query)
```

**Arguments**

query	A query (character) to search for in the options.
-------	---

**Value**

Returns a list of matching options.

**Examples**

```
search_options("learning rate")
search_options("tokens")
search_options("invalid query")
```

---

show	<i>Show model information</i>
------	-------------------------------

---

## Description

Model information includes details, model file, template, parameters, license, system prompt.

## Usage

```
show(  
  name,  
  verbose = FALSE,  
  output = c("jsonlist", "resp", "raw"),  
  endpoint = "/api/show",  
  host = NULL  
)
```

## Arguments

name	Name of the model to show
verbose	Returns full data for verbose response fields. Default is FALSE.
output	The output format. Default is "jsonlist". Other options are "resp", "raw".
endpoint	The endpoint to show the model. Default is "/api/show".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

## Value

A response in the format specified in the output parameter.

## References

[API documentation](#)

## Examples

```
# show("llama3") # returns jsonlist  
show("llama3", output = "resp") # returns response object
```

---

test_connection	<i>Test connection to Ollama server</i>
-----------------	---

---

**Description**

Tests whether the Ollama server is running or not.

**Usage**

```
test_connection(url = "http://localhost:11434", logical = FALSE)
```

**Arguments**

url	The URL of the Ollama server. Default is http://localhost:11434
logical	Logical. If TRUE, returns a boolean value. Default is FALSE.

**Value**

Boolean value or httr2 response object, where status\_code is either 200 (success) or 503 (error).

**Examples**

```
test_connection(logical = TRUE)
test_connection("http://localhost:11434") # default url
test_connection("http://127.0.0.1:11434")
```

---

validate_message	<i>Validate a message</i>
------------------	---------------------------

---

**Description**

Validate a message to ensure it has the required fields and the correct data types for the chat() function.

**Usage**

```
validate_message(message)
```

**Arguments**

message	A list with a single message of list class.
---------	---

**Value**

TRUE if message is valid, otherwise an error is thrown.

**Examples**

```
validate_message(create_message("Hello"))
validate_message(list(role = "user", content = "Hello"))
```

---

validate_messages	<i>Validate a list of messages</i>
-------------------	------------------------------------

---

**Description**

Validate a list of messages to ensure they have the required fields and the correct data types for the chat() function.

**Usage**

```
validate_messages(messages)
```

**Arguments**

messages            A list of messages, each of list class.

**Value**

TRUE if all messages are valid, otherwise warning messages are printed and FALSE is returned.

**Examples**

```
validate_messages(create_messages(
  create_message("Be friendly", "system"),
  create_message("Hello")
))
```

---

validate_options	<i>Validate additional options or parameters provided to the API call</i>
------------------	---

---

**Description**

Validate additional options or parameters provided to the API call

**Usage**

```
validate_options(...)
```

**Arguments**

...                  Additional options or parameters provided to the API call

**Value**

TRUE if all additional options are valid, FALSE otherwise

**Examples**

```
validate_options(mirostat = 1, mirostat_eta = 0.2, num_ctx = 1024)
validate_options(mirostat = 1, mirostat_eta = 0.2, invalid_opt = 1024)
```

# Index

- \* **datasets**
  - model\_options, [18](#)
  - package\_config, [19](#)
- append\_message, [2](#)
- chat, [3](#)
- check\_option\_valid, [5](#)
- check\_options, [5](#)
- copy, [6](#)
- create, [6](#)
- create\_message, [7](#)
- create\_messages, [8](#)
- create\_request, [9](#)
  
- delete, [9](#)
- delete\_message, [10](#)
  
- embed, [11](#)
- embeddings, [12](#)
- encode\_images\_in\_messages, [13](#)
  
- generate, [13](#)
  
- image\_encode\_base64, [15](#)
- insert\_message, [16](#)
  
- list\_models, [16](#)
  
- model\_avail, [17](#)
- model\_options, [18](#)
  
- ohelp, [18](#)
  
- package\_config, [19](#)
- prepend\_message, [19](#)
- ps, [20](#)
- pull, [20](#)
- push, [21](#)
  
- resp\_process, [22](#)
  
- search\_options, [23](#)
- show, [24](#)
  
- test\_connection, [25](#)
  
- validate\_message, [25](#)
- validate\_messages, [26](#)
- validate\_options, [26](#)