Package: ask (via r-universe)

May 28, 2026

Title ask R anything

Version 0.0.0.9000

Description Interface to do useful things in R with LLMs.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2.9000

Imports base64enc, curl, diffviewer, dplyr, fs, htmltools, htmlwidgets, httr, jsonlite, knitr, memoise, miniUI, rlang, rmarkdown, rstudioapi, shiny, shinyjs, vctrs, withr

URL https://github.com/moodymudskipper/ask

BugReports https://github.com/moodymudskipper/ask/issues

Suggests constructive, clipr, gmailr, testthat (>= 3.0.0), rvest, pdftools

Config/testthat/edition 3

Config/pak/sysreqs cmake make libuv1-dev libssl-dev zlib1g-dev

Repository https://cynkra.r-universe.dev

Date/Publication 2024-11-10 20:48:15 UTC

RemoteUrl https://github.com/moodymudskipper/ask

RemoteRef HEAD

RemoteSha f96abf915e62b6e55c8ea8b1212c89703fba573a

Contents

`again` *Ask again*

Description

Ask the same thing again, by default with the same parameters, if `cache` is used a new request will be sent for the given arguments, and cached instead of the older value.

Usage

```
again(
  conversation = last_conversation(),
  model = NULL,
  cache = getOption("ask.cache"),
  api_args = NULL,
  api_key = NULL
)
```

Arguments

conversation	A conversation, initiated by ask() or followed up by follow_up()
model, api_args	inherited from the last item of conversation by default
cache	A path where to cache the outputs, or "ram" to store them in RAM. useful to spare tokens and to have reproducible code.
api_key	API key

---

all_models *List all current models*

Description

These are all current models from openai, anthropic and ollama. We don't guarantee that they all work with ask though as we've tested a limited amount. ChatGPT, Claude, and llama models seem to work ok, please open a ticket to request further support.

Usage

```
all_models(
  openai_api_key = Sys.getenv("OPENAI_API_KEY"),
  only_supported = TRUE
)
```

Arguments

openai_api_key	an api key for open ai (for other models we use webscraping rather than rest APIs to get the list of models)

Value

A tibble

---

ask *Ask anything*

Description

Ask anything

Usage

```
ask(
  prompt = listen(),
  context = NULL,
  model = getOption("ask.model", "gpt-4o"),
  image = NULL,
  cache = getOption("ask.cache"),
  api_args = NULL,
  api_key = NULL
)
```

Arguments

prompt	Your request, a string or a character vector that will be concatenated to a string with line breaks as separators.
context	An object of class "ask_context" usually built from a call to `context()` or a `context_*()` function. It is used to define a "system" message that define the behavior, tone or focus of the assistant.
model	The model to choose, see https://platform.openai.com/docs/models or call `all_models()` for chatgpt model, or use ollama models such as "llama3.1".
image	Path or URL to image to provide. Only considered for gpt models.
cache	A path where to cache the outputs, or "ram" to store them in RAM. useful to spare tokens and to have reproducible code.
api_args	Additional arguments to the api, depend on the model and rarely needed, most useful ones include `temperature` and `seed`.
api_key	API key

Value

a converstaion object

`ask-options` *Options for the 'ask' package*

Description

- `ask.cache`: For optimal reproducibility, a path to a directory where to store the cache, or `ram` to store it into the RAM.

ask_boolean *Ask a boolean question*

Description

Ask a boolean question

Usage

```
ask_boolean(prompt = listen(), context = NULL, ...)

ask_numeric(prompt = listen(), unit = NULL, context = NULL, ...)
```

Arguments

prompt	Your request, a string or a character vector that will be concatenated to a string with line breaks as separators.
context	An object of class "ask_context" usually built from a call to `context()` or a `context_*()` function. It is used to define a "system" message that define the behavior, tone or focus of the assistant.
...	forwarded to `ask()`
unit	Optional, if not the SI is used, but better provide it.

ask_clipboard *Ask and copy output to clipboard*

Description

This function behaves like `ask()` but copies the output to the clipboard.

Usage

```
ask_clipboard(
  prompt = listen(),
  context = NULL,
  model = getOption("ask.model", "gpt-4o"),
  image = NULL,
  cache = getOption("ask.cache"),
  api_args = NULL,
  api_key = NULL
)
```

Arguments

prompt	Your request, a string or a character vector that will be concatenated to a string with line breaks as separators.
context	An object of class "ask_context" usually built from a call to `context()` or a `context_*()` function. It is used to define a "system" message that define the behavior, tone or focus of the assistant.
model	The model to choose, see https://platform.openai.com/docs/models or call `all_models()` for chatgpt model, or use ollama models such as "llama3.1".
image	Path or URL to image to provide. Only considered for gpt models.
cache	A path where to cache the outputs, or "ram" to store them in RAM. useful to spare tokens and to have reproducible code.
api_args	Additional arguments to the api, depend on the model and rarely needed, most useful ones include `temperature` and `seed`.
api_key	API key

---

ask_console *Ask in the Console*

Description

These are simple wrappers around `ask()` and `follow_up()`, that print to the console rather than the viewer (Something that can also be achieved with `print(ask(...), venue = "console")`). Only the last answer is printed but the output object is still the entire conversation.

Usage

```
ask_console(
  prompt = listen(),
  context = NULL,
  model = getOption("ask.model", "gpt-4o"),
  image = NULL,
  cache = getOption("ask.cache"),
  api_args = NULL,
  api_key = NULL,
  follow_up = FALSE
)

follow_up_console(
  prompt = listen(),
  context = NULL,
  conversation = last_conversation(),
  model = NULL,
  image = NULL,
  cache = getOption("ask.cache"),
  ap_args = NULL,
  api_key = NULL
)
```

Arguments

prompt	Your request, a string or a character vector that will be concatenated to a string with line breaks as separators.
context	An object of class "ask_context" usually built from a call to `context()` or a `context_*()` function. It is used to define a "system" message that define the behavior, tone or focus of the assistant.
model	The model to choose, see https://platform.openai.com/docs/models or call `all_models()` for chatgpt model, or use ollama models such as "llama3.1".
image	Path or URL to image to provide. Only considered for gpt models.
cache	A path where to cache the outputs, or "ram" to store them in RAM. useful to spare tokens and to have reproducible code.
api_args	Additional arguments to the api, depend on the model and rarely needed, most useful ones include `temperature` and `seed`.
api_key	API key
follow_up	Whether to automatically follow up in the console, this triggers an interactive prompt that you can exit by pressing Esc or Ctrl+C..
conversation	A conversation, initiated by `ask()` or followed up by `follow_up()`

Value

The result from the `ask()` function.

---

ask_in_place *Ask for script updates in place*

Description

This will change files in place a terminal command in a new terminal, and print extra information in the R console. `context = context_repo()` is often useful here unless the package is not too big.

Usage

```
ask_in_place(
  prompt = listen(),
  context = NULL,
  model = getOption("ask.model", "gpt-4o"),
  ...
)

follow_up_in_place(
  prompt = listen(),
  context = NULL,
  conversation = last_conversation(),
  ...
)
```

Arguments

prompt	Your request, a string or a character vector that will be concatenated to a string with line breaks as separators.
context	An object of class "ask_context" usually built from a call to `context()` or a `context_*()` function. It is used to define a "system" message that define the behavior, tone or focus of the assistant.
model	The model to choose, see https://platform.openai.com/docs/models or call `all_models()` for chatgpt model, or use ollama models such as "llama3.1".
...	Forwarded to `ask()` or `follow_up()`
conversation	A conversation, initiated by `ask()` or followed up by `follow_up()`

Examples

```
## Not run:
ask_in_place("update the existing readme with useful missing info", context = context_repo())

## End(Not run)
```

---

ask_open *Ask to open one or several files*

Description

Ask to open one or several files

Usage

```
ask_open(
  prompt,
  context = NULL,
  model = getOption("ask.model", "gpt-4o"),
  image = NULL,
  cache = getOption("ask.cache"),
  api_args = NULL,
  api_key = NULL
)
```

Arguments

prompt	Your request, a string or a character vector that will be concatenated to a string with line breaks as separators.
context	An object of class "ask_context" usually built from a call to `context()` or a `context_*()` function. It is used to define a "system" message that define the behavior, tone or focus of the assistant.
model	The model to choose, see https://platform.openai.com/docs/models or call `all_models()` for chatgpt model, or use ollama models such as "llama3.1".

image	Path or URL to image to provide. Only considered for gpt models.
cache	A path where to cache the outputs, or "ram" to store them in RAM. useful to spare tokens and to have reproducible code.
api_args	Additional arguments to the api, depend on the model and rarely needed, most useful ones include temperature and seed.
api_key	API key

Value

A conversation object

Examples

```
## Not run:
ask_open("the file(s) touched by the last commit", context_commits())
ask_open("the script were roxygen2 imports are defined", context_repo())

## End(Not run)
```

---

ask_smart *ask for anything or any change*

Description

ask_smart() guesses which ask*() function and which context*() function is appropriate, so you don't need to think too much or know the package in depth.

Usage

```
ask_smart(
  prompt = listen(),
  context = NULL,
  model = getOption("ask.model", "gpt-4o"),
  cache = getOption("ask.cache"),
  api_args = NULL,
  api_key = NULL
)
```

Arguments

prompt	Your request, a string or a character vector that will be concatenated to a string with line breaks as separators.
context	An object of class "ask_context" usually built from a call to context() or a context_*() function. It is used to define a "system" message that define the behavior, tone or focus of the assistant.
model	The model to choose, see https://platform.openai.com/docs/models or call all_models() for chatgpt model, or use ollama models such as "llama3.1".

cache	A path where to cache the outputs, or "ram" to store them in RAM. useful to spare tokens and to have reproducible code.
api_args	Additional arguments to the api, depend on the model and rarely needed, most useful ones include temperature and seed.
api_key	API key

Details

It works by calling a llm twice, first to figure out the proper functions to use, then through these functions. As a consequence it is slower and spends more token (with an increased chance of reaching the limit of maximum tokens per minute)

---

ask_terminal *Ask for a terminal command*

Description

This will suggest a terminal command in a new terminal, and print extra information in the R console.

Usage

```
ask_terminal(prompt = listen(), context = NULL, ...)
```

Arguments

prompt	Your request, a string or a character vector that will be concatenated to a string with line breaks as separators.
context	An object of class "ask_context" usually built from a call to context() or a context_*() function. It is used to define a "system" message that define the behavior, tone or focus of the assistant.
...	Forwarded to ask()

Examples

```
## Not run:
ask_terminal("who contributed to this project?")
ask_terminal("show the latest 5 changes in compact form")

## End(Not run)
```

ask_tibble *Ask for a tibble*

Description

Ask for a tibble

Usage

```
ask_tibble(prompt = listen(), context = NULL, ...)
```

Arguments

prompt	Your request, a string or a character vector that will be concatenated to a string with line breaks as separators.
context	An object of class "ask_context" usually built from a call to `context()` or a `context_*()` function. It is used to define a "system" message that define the behavior, tone or focus of the assistant.
...	forwarded to `ask()`

capture_plot *Capture plot*

Description

Capture current plot (can be a base plot, a ggplot2 plot or anything else) and saves it into a temp png file whose path is returned. meant to be used with the `image` argument of `ask*()` functions.

Usage

```
capture_plot(width = 800, height = 600)
```

Arguments

width	image width
height	image height

Value

The path to the created file

`context` *Build a context object*

Description

Build an object of class "ask_context" to be used in the `context` argument. of `ask*` functions. Most of the time using the provided `context_*()` functions is sufficient.

Usage

```
context(...)
```

Arguments

`...`	named character vectors, or other context ojects (named or not). These dots are dynamic in the rlang sense (`?rlang::`dyn-dots``) so they can be conveniently named using glue syntax and `:=`.

Details

`context()` can be used :

- To combine existing contexts, as in `context(context1, context2)`
- To define a completely new context as in
- To nest contexts as in `context("new label" = context(context1, context2))`
- For any combination of the above

They are implemented as nested lists so the labels can be numbered when flattened internally in `ask*()` functions or when printing them. Run examples below to see how this works.

Value

An object of class `"ask_context"`

Examples

```
context_smith <- context(
  "Smith family" =
    "The Smith family is John, Wendy, and their 3 kids Boris, Judith and Michael."
)
context_johnson <- context("Johnson family" = "The Johnson family is Robert and his sister Brenda.")
context_house <- context(
  "House" =
    "The Smiths and the Johnsons live in the same house, called 'the happy place',"
)
context_johnson
full_context <- context(
  Families = context(context_smith, context_johnson),
```

```
  context_house
)
full_context
## Not run:
ask("What are the names of the kids that live in the happy place", context = full_context)
ask_numeric("How many people live in the happy place?", context = full_context)
ask_boolean("Are Robert and Brenda related?", context = full_context)

## End(Not run)
```

`context_clipboard` *Contextualize clipboard content*

Description

Contextualize clipboard content

Usage

```
context_clipboard()
```

Value

An object of class "ask_context"

`context_commits` *Contextualize git commits*

Description

Contextualize git commits

Usage

```
context_commits(n = 5)
```

Arguments

`n`	The number of commits to return context for (default is 5)

Value

An object of class "ask_context"

context_diff *Contextualize uncommited changes*

Description

Contextualize uncommited changes

Usage

```
context_diff()
```

Value

An object of class "ask_context"

context_github *Contextualize GitHub issues*

Description

Contextualize GitHub issues

Usage

```
context_github(
  repo = NULL,
  state = c("open", "closed", "all"),
  filter = c("assigned", "created", "mentioned", "subscribed", "all"),
  labels = NULL,
  sort = c("created", "updated", "comments"),
  direction = c("asc", "desc"),
  since = NULL,
  n = 30
)
```

Arguments

repo	A string in the form 'my.org/my.repo' specifying the repository. If not provided, will be fetched from the URL field of the DESCRIPTION file.
state	State of the issues to return ('open', 'closed', or 'all').
filter	A string, see details.
labels	A vector of labels to filter the issues by.
sort	What to sort the results by
direction	The direction to sort the results by.
since	A date or time used to only show results that were last updated after the given time.
n	Number of results (max 100).

Value

An object of class "ask_context"

values for filter

- **assigned** (default): Issues assigned to the authenticated user.
- **created**: Issues created by the authenticated user.
- **mentioned**: Issues mentioning the authenticated user.
- **subscribed**: Issues the user is subscribed to.
- **all**: All issues owned by the authenticated user.

---

context_gmail *Contextualize Gmail messages from threads*

Description

Contextualize Gmail messages from threads

Usage

```
context_gmail(
  search = NULL,
  num_results = 5,
  page_token = NULL,
  label_ids = NULL,
  include_spam_trash = NULL,
  user_id = "me"
)
```

Arguments

`search`, `num_results`, `page_token`, `label_ids`, `include_spam_trash`, `user_id`
Forwarded to `gmailr::gm_threads()`

Value

An object of class "ask_context"

context_objects *Contextualize R objects*

Description

Contextualize R objects

Usage

```
context_objects(objects)
```

Arguments

objects	A list of R objects to be contextualized.

Value

An object of class "ask_context"

context_package *Contextualize package*

Description

Contextualize package

Usage

```
context_package(
  pkg,
  DESCRIPTION = TRUE,
  NAMESPACE = TRUE,
  README = TRUE,
  NEWS = TRUE,
  LICENSE = TRUE,
  help_files = TRUE,
  code = FALSE,
  vignettes = FALSE
)
```

Arguments

pkg	Name of the package.
DESCRIPTION	Boolean, whether to include the DESCRIPTION file.
NAMESPACE	Boolean, whether to include the NAMESPACE file.
README	Boolean, whether to include the README file.
NEWS	Boolean, whether to include the NEWS.md file.
LICENSE	Boolean, whether to include the LICENSE file.
help_files	Boolean, whether to include help files.
code	Boolean, whether to include code files.
vignettes	Boolean, whether to include vignettes.

Value

An object of class "ask_context"

---

context_pdf *Contextualize pdf text content*

Description

Contextualize pdf text content

Usage

```
context_pdf(
  file,
  pages = NULL,
  opw = "",
  upw = "",
  dpi = 600,
  language = "eng",
  options = NULL,
  type = "text"
)
```

Arguments

file	file path or raw vector with pdf data
pages	which pages of the pdf file to extract
opw	string with owner password to open pdf
upw	string with user password to open pdf
dpi	resolution to render image that is passed to pdf_convert.
language	passed to tesseract to specify the languge of the engine.
options	passed to tesseract to specify OCR parameters
type	one or more from "text", "ocr_text", "ocr_data". Decides wether to use `pdftools::pdf_text()`, `pdftools::pdf_ocr_text()`, or/and `pdftools::pdf_ocr_data()`.

Value

An object of class "ask_context"

Examples

```
## Not run:
ask("what is this about?", context_pdf("http://arxiv.org/pdf/1403.2805.pdf"))

## End(Not run)
```

---

context_repo *Contextualize all R files in the repository*

Description

Contextualize all R files in the repository

Usage

```
context_repo()
```

Value

An object of class "ask_context"

---

context_script *Contextualize a script*

Description

Contextualize a script

Usage

```
context_script(file = NULL)
```

Arguments

file	Path to the file, if NULL we consider the active script

Value

An object of class "ask_context"

context_session_info *Contextualize the R session information*

Description

Contextualize the R session information

Usage

```
context_session_info()
```

Value

An object of class "ask_context"

context_url *Contextualize an url*

Description

Contextualize an url

Usage

```
context_url(url, only_text = TRUE)
```

Arguments

url	An url
only_text	Whether to cleanup the html and extract the text only

Value

An object of class "ask_context"

conversation_manager *Conversation Manager*

Description

A small shiny app to manage all conversations. Whenever an `ask*()` function is used (not just `ask()`), a new conversation is recorded and can be accessed through the conversation manager. There we can expand or remove conversations, or even start a new one from scratch. When you pick a conversation you leave the shiny app and the chosen conversation is printed in the viewer and stored as `last_conversation()` so you can use `follow_up()` or `again()` to expand it..

Usage

```
conversation_manager()
```

Value

A conversation

follow_up *Follow up a request*

Description

Continue a conversation, by default with the same parameters.

Usage

```
follow_up(
  prompt = listen(),
  context = NULL,
  conversation = last_conversation(),
  image = NULL,
  cache = getOption("ask.cache"),
  api_args = NULL,
  api_key = NULL
)
```

Arguments

`prompt`	Your request, a string or a character vector that will be concatenated to a string with line breaks as separators.
`context`	An object of class "ask_context" usually built from a call to `context()` or a `context_*()` function. It is used to define a "system" message that define the behavior, tone or focus of the assistant.

conversation	A conversation, initiated by ask() or followed up by follow_up()
image	Path or URL to image to provide. Only considered for gpt models.
cache	A path where to cache the outputs, or "ram" to store them in RAM. useful to spare tokens and to have reproducible code.
api_key	API key
model, api_args	inherited from the last item of conversation by default

Value

a conversation object

---

last_conversation *Fetch the last updated conversation*

Description

Fetch the last updated conversation

Usage

```
last_conversation()
```

---

listen *listen to voice input*

Description

listen to voice input

Usage

```
listen(verbose = TRUE)
```

Arguments

verbose	whether to print the recorded text as a message

Value

A string

Index