# Package: ragR (via r-universe)

July 22, 2026

**Title** Retrieval-Augmented Generation and RAG Evaluation Tools

**Version** 0.1.0

**Description** Provides tools for document ingestion, embedding storage, retrieval-augmented generation (RAG), and evaluation of question-answering systems. The package includes an R-native vector store, wrappers for OpenAI embedding and chat-completion application programming interfaces (APIs), question-answering logging utilities, and large language model (LLM)-based evaluation metrics for context precision, context recall, answer relevance, and faithfulness. These metrics are based on the Retrieval-Augmented Generation Assessment (RAGAS) framework. The retrieval-augmented generation methodology is described by Lewis et al. (2020) ``Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" <doi:10.48550/arXiv.2005.11401>. The evaluation metrics are based on Es et al. (2024) ``RAGAS: Automated Evaluation of Retrieval Augmented Generation" <doi:10.18653/v1/2024.eacl-demo.16>.

**License** GPL-3

**URL** https://github.com/aimalrehman92/ragR

**BugReports** https://github.com/aimalrehman92/ragR/issues

**Encoding** UTF-8

**Language** en-US

**Depends** R (>= 4.1.0)

**Imports** dplyr, httr2, jsonlite, pdftools, readtext, tibble

**Suggests** plumber, stringr, yaml, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Config/roxygen2/version** 8.0.0

**NeedsCompilation** no

**Author** Muhammad Aimal Rehman [aut, cre], Zhili Lu [aut], Chi-Kuang Yeh [aut]

**Maintainer** Muhammad Aimal Rehman <rehman.aimal@gmail.com>

**Repository** https://cran.r-universe.dev

**Date/Publication** 2026-07-22 07:40:02 UTC

**RemoteUrl** https://github.com/cran/ragR

**RemoteRef** HEAD

**RemoteSha** f01a408bd5d8935e6500b18833e581ed89998b6e

# Contents

ragR-package *ragR: Retrieval-Augmented Generation and Evaluation Tools*

## Description

The **ragR** package provides tools for building and evaluating retrieval-augmented generation systems in R. It includes:

- A data-ingestion pipeline for PDF, text, and Word documents
- An R-native vector store for document chunks and embeddings
- A configurable retrieval-augmented generation pipeline
- Persistent question-answer logging for reproducible evaluation
- Large-language-model-scored evaluation metrics, including context precision, context recall, answer relevance, and faithfulness
- A Plumber-based HTTP API suitable for chatbot and evaluation frontends

## Architecture

The package is organized into several modules:

- **Ingestion** (`ingestion_files.R`)
  Extracts text from PDF, TXT, and DOCX files, divides the text into chunks, and prepares those chunks for embedding and storage. The vector-store path must be supplied explicitly by the caller.
- **Embeddings and chat** (`embeddings_openai.R`)
  Wraps the OpenAI REST API for embedding models, such as `"text-embedding-3-small"`, and chat models, such as `"gpt-4o-mini"`. The API key is read from the `OPENAI_API_KEY` environment variable.
- **Vector store** (`vectorstore_interface.R`)
  Implements an R-native vector store using tibbles and numeric matrices. Functions are provided to load the store, add or update records, retrieve similar chunks, delete collections, and clear the complete store.
  All vector-store functions require an explicit storage path.
- **Retrieval-augmented generation pipeline** (`rag_pipeline.R`)
  Given a user question, the pipeline:
  1. Computes an embedding for the question
  2. Retrieves similar chunks from the selected vector-store collection
  3. Builds a grounded, context-aware prompt
  4. Calls a chat model to generate an answer
- **Question-answer logging and persistence** (`qa_logging.R`, `qa_persistence.R`)
  Stores interactions including the question, model answer, retrieved chunks, final grounded prompt, model names, collection name, and timestamp. Helpers are provided to load, save, and clear logs.
  Log locations must be supplied explicitly.

- **Evaluation metrics and summaries** (`ragas_metrics.R`, `ragas_summary.R`)
  Computes large-language-model-scored evaluation metrics, including context precision, context recall, answer relevance, faithfulness, and an overall score. Summary helpers aggregate metric results across question-answer pairs.
- **Reports** (`ragas_report.R`)
  Generates an evaluation report by:
  - Computing metrics from a saved question-answer log
  - Saving the resulting metrics to an explicitly supplied RDS path
  - Writing a summary CSV to an explicitly supplied output directory
  - Writing a bar chart of mean metric values to that output directory
- **API handlers** (`api_handlers.R`)
  Connect the core package functions to HTTP endpoints. Storage paths are supplied by the server configuration rather than by client request bodies or hidden package defaults.

## HTTP API

The development server in `scripts/dev/run_api.R` exposes:

- `POST /chat`
  Runs the retrieval-augmented generation pipeline for a user question and appends the interaction to the configured question-answer log.
- `GET /ragas`
  Computes and returns evaluation metrics and a summary for the logged question-answer interactions.
- `POST /ragas/report`
  Computes metrics from the configured question-answer log and writes the metrics, summary CSV, and plot to server-configured locations.
- `POST /ragas/clear`
  Clears the configured question-answer log, metrics file, and report directory.
- `POST /clear`
  Clears one collection from the configured vector store.
- `POST /clear_all`
  Clears all collections from the configured vector store using `api_clear_all_handler()` and `vectorstore_clear_all()`.

The development server uses a temporary storage directory unless a persistent location is supplied through the `RAGR_API_DATA_DIR` environment variable.

## Usage

Programmatic usage inside R:

```
library(ragR)

vectorstore_path <- tempfile(fileext = ".rds")
```

```
# Documents must first be ingested into this vector store.
res <- query_rag(
  question = "What is this document about?",
  collection = "default",
  vectorstore_path = vectorstore_path,
  top_k = 5L,
  embedding_model = "text-embedding-3-small",
  chat_model = "gpt-4o-mini",
  system_prompt = "You are a helpful assistant."
)

cat(res$answer)
```

To run the development HTTP server from the project root:

```
source("scripts/dev/run_api.R")
```

Interactive API documentation is then available at `http://127.0.0.1:8000/__docs__/`.

## Author(s)

**Maintainer**: Muhammad Aimal Rehman <rehman.aimal@gmail.com>

Authors:

- Muhammad Aimal Rehman <rehman.aimal@gmail.com>
- Zhili Lu
- Chi-Kuang Yeh

## See Also

Useful links:

- https://github.com/aimalrehman92/ragR
- Report bugs at https://github.com/aimalrehman92/ragR/issues

---

`api_chat_handler` *API handler: chat with the RAG pipeline*

---

## Description

This handler receives a parsed JSON request body, runs the RAG pipeline, appends the interaction to the QA log, and returns the model answer.

## Usage

```
api_chat_handler(body, qa_log_path, vectorstore_path)
```

### Arguments

| | |
|---|---|
| `body` | Parsed JSON request body. |
| `qa_log_path` | Character scalar; path to the QA log RDS file. |
| `vectorstore_path` | Character scalar; path to the vector-store RDS file. |

### Details

File locations must be supplied explicitly. The handler does not choose default locations in the user's working directory.

### Value

A list suitable for JSON serialization.

---

## `api_clear_all_handler` *API handler: clear the entire vector store*

---

### Description

Removes all collections and stored embeddings from the vector store. The vector-store path must be supplied explicitly.

### Usage

```
api_clear_all_handler(vectorstore_path)
```

### Arguments

| | |
|---|---|
| `vectorstore_path` | Character scalar; path to the vector-store RDS file. |

### Value

A list containing `status` and `message`.

### Examples

```
path <- tempfile(fileext = ".rds")

api_clear_all_handler(
  vectorstore_path = path
)
```

## api_clear_handler *API handler: clear a vector-store collection*

### Description

Deletes a collection from the R-native vector store. The request body may specify a collection name; otherwise, `"default"` is used. The vector-store path must be supplied explicitly.

### Usage

```
api_clear_handler(body, vectorstore_path)
```

### Arguments

| | |
|---|---|
| `body` | A list parsed from a JSON request body, or `NULL`. |
| `vectorstore_path` | Character scalar; path to the vector-store RDS file. |

### Value

A list suitable for JSON serialization, containing `status`, `collection`, and `message`.

### Examples

```
path <- tempfile(fileext = ".rds")

api_clear_handler(
  body = list(collection = "example"),
  vectorstore_path = path
)
```

## api_ragas_clear_handler *API handler: clear RAG evaluation files*

### Description

Clears the saved question-answering log, saved evaluation metrics, and generated report directory. All locations must be supplied explicitly.

### Usage

```
api_ragas_clear_handler(qa_log_path, qa_metrics_path, output_dir)
```

### Arguments

| | |
|---|---|
| qa_log_path | Character scalar; path to the question-answering log RDS file. |
| qa_metrics_path | Character scalar; path to the saved metrics RDS file. |
| output_dir | Character scalar; directory containing generated report files. |

### Value

A list containing status and message.

### Examples

```
qa_log_path <- tempfile(fileext = ".rds")
qa_metrics_path <- tempfile(fileext = ".rds")
output_dir <- tempfile()

save_qa_log(qa_log_empty(), qa_log_path)
save_qa_metrics(qa_metrics_empty(), qa_metrics_path)
dir.create(output_dir)

api_ragas_clear_handler(
  qa_log_path = qa_log_path,
  qa_metrics_path = qa_metrics_path,
  output_dir = output_dir
)
```

---

## api_ragas_handler *API handler: compute RAG evaluation metrics*

---

### Description

Loads a saved question-answering log, computes evaluation metrics, and returns both the per-interaction metrics and their summary. The QA-log path must be supplied explicitly.

### Usage

```
api_ragas_handler(path, judge_model = "gpt-4o-mini")
```

### Arguments

| | |
|---|---|
| path | Character scalar; path to the QA log RDS file. |
| judge_model | Character scalar; large language model used to score the evaluation metrics. |

### Value

A list containing status, n_qa, metrics, and summary.

## Examples

```
path <- tempfile(fileext = ".rds")
save_qa_log(qa_log_empty(), path)

api_ragas_handler(
  path = path,
  judge_model = "gpt-4o-mini"
)
```

---

`api_ragas_report_handler` *API handler: generate a RAG evaluation performance report*

---

## Description

Loads a saved question-answering log, computes evaluation metrics, and writes the metrics, summary CSV, and plot to explicitly supplied locations.

## Usage

```
api_ragas_report_handler(
  qa_log_path,
  qa_metrics_path,
  output_dir,
  judge_model = "gpt-4o-mini"
)
```

## Arguments

| | |
|---|---|
| `qa_log_path` | Character scalar; path to the question-answering log RDS file. |
| `qa_metrics_path` | Character scalar; path to the metrics RDS output file. |
| `output_dir` | Character scalar; directory where the summary CSV and plot PNG are written. |
| `judge_model` | Character scalar; large language model used to score the evaluation metrics. |

## Value

A list containing `status`, `n_qa`, `qa_metrics_path`, `summary_csv_path`, and `plot_path`. If the QA log is empty, the returned list instead includes an explanatory `message`.

## chunk_text_sentence *Chunk text into sentences (strict)*

### Description

Splits text into individual sentences. Each returned element is intended to be exactly one sentence (best-effort based on punctuation).

### Usage

```
chunk_text_sentence(text)
```

### Arguments

| | |
|---|---|
| text | Character scalar. |

### Value

Character vector; one sentence per element.

### Examples

```
text <- paste(
  "Retrieval-augmented generation retrieves relevant information.",
  "The retrieved information is then supplied to a language model.",
  "This can help produce more grounded responses."
)

chunk_text_sentence(text)
```

## clear_qa_log *Clear the QA log on disk*

### Description

This helper overwrites the QA log file with an empty QA log, as created by qa_log_empty(). It is intended for starting a fresh evaluation session.

### Usage

```
clear_qa_log(path)
```

### Arguments

| | |
|---|---|
| path | Character scalar; path to the QA log RDS file. |

## Value

Invisibly, the path to the saved file.

## Examples

```
path <- tempfile(fileext = ".rds")

clear_qa_log(path)
load_qa_log(path)
```

---

`clear_qa_metrics` *Clear QA metrics on disk*

---

## Description

This helper overwrites the QA metrics file with an empty metrics tibble, as created by `qa_metrics_empty()`.

## Usage

```
clear_qa_metrics(path)
```

## Arguments

| | |
|---|---|
| `path` | Character scalar; path to the QA metrics RDS file. |

## Value

Invisibly, the path to the saved file.

## Examples

```
path <- tempfile(fileext = ".rds")

clear_qa_metrics(path)
load_qa_metrics(path)
```

## compute_ragas_metrics *Compute RAGAS metrics*

### Description

Convenience wrapper for computing LLM-scored RAGAS-style metrics.

### Usage

```
compute_ragas_metrics(
  qa_log,
  judge_model = "gpt-4o-mini",
  seed = NULL,
  embedding_model = "text-embedding-3-small",
  answer_relevance_strictness = 3L
)
```

### Arguments

| | |
|---|---|
| qa_log | QA log tibble. |
| judge_model | Judge model for LLM-scored metrics. |
| seed | Optional integer forwarded to LLM-scored metrics. |
| embedding_model | Embedding model for answer relevance. |
| answer_relevance_strictness | Number of reverse questions generated for answer relevance. |

### Value

A metrics tibble.

## compute_ragas_metrics_llm *Compute RAGAS-style metrics (LLM scored)*

### Description

This implementation mirrors the structured Python RAGAS workflow rather than asking the judge model for one direct scalar score per metric.

## Usage

```
compute_ragas_metrics_llm(
  qa_log,
  judge_model = "gpt-4o-mini",
  seed = NULL,
  embedding_model = "text-embedding-3-small",
  answer_relevance_strictness = 3L
)
```

## Arguments

| | |
|---|---|
| qa_log | A tibble created and populated by log_rag_interaction(). |
| judge_model | Character scalar; judge model name. |
| seed | Optional integer forwarded to judge model calls. |
| embedding_model | Embedding model name used for answer relevance. |
| answer_relevance_strictness | Number of generated reverse questions for answer relevance. Python RAGAS defaults to 3. |

## Details

Notes:

- context_precision uses answer_reference if available; otherwise it uses answer_model, matching the Python with-reference / without-reference split.
- answer_relevance requires an embedding helper to be available in the package environment.

## Value

A tibble with one row per qa_id and metric columns.

---

# generate_openai_chat *Generate a chat completion from OpenAI*

---

## Description

Used by the RAG pipeline and RAGAS-style metric prompts to generate text from a constructed prompt.

### Usage

```
generate_openai_chat(
  prompt,
  model = "gpt-4o-mini",
  system_message = NULL,
  temperature = 0,
  max_output_tokens = 512L,
  seed = NULL,
  api_key = NULL
)
```

### Arguments

| | |
|---|---|
| prompt | Character scalar; the user/content prompt. |
| model | Character scalar; chat model name, e.g., "gpt-4o-mini". |
| system_message | Optional API-level system message. If NULL or empty, no system message is sent. |
| temperature | Numeric; sampling temperature. |
| max_output_tokens | Integer or NULL; maximum tokens to generate. If NULL, the parameter is omitted from the request. |
| seed | Optional integer. If provided and supported by the model/provider, it is sent as seed to encourage reproducible outputs. |
| api_key | OpenAI API key. If NULL, uses the OPENAI_API_KEY environment variable. |

### Value

Character scalar containing the model's answer.

---

## generate_ragas_report *Generate a RAG evaluation performance report*

---

### Description

Loads a question-answering log from disk, computes large language model evaluation metrics, and writes:

### Usage

```
generate_ragas_report(
  qa_log_path,
  qa_metrics_path,
  output_dir,
  judge_model = "gpt-4o-mini"
)
```

### Arguments

| | |
|---|---|
| qa_log_path | Character scalar; path to the question-answering log RDS file. |
| qa_metrics_path | Character scalar; path to the metrics RDS output file. |
| output_dir | Character scalar; directory where the summary CSV and plot PNG are written. |
| judge_model | Character scalar; large language model used to score the evaluation metrics. |

### Details

- metrics to an RDS file;
- a summary to `ragas_summary.csv`;
- a bar plot of mean metrics to `ragas_means.png`.

All input and output locations must be supplied explicitly.

### Value

A list containing `status`, `n_qa`, `qa_metrics_path`, `summary_csv_path`, and `plot_path`.

---

## generate_ragas_report_llm

*Generate a RAG evaluation report using large language model scoring*

---

### Description

This is an explicit alias for `generate_ragas_report()`.

### Usage

```
generate_ragas_report_llm(
  qa_log_path,
  qa_metrics_path,
  output_dir,
  judge_model = "gpt-4o-mini"
)
```

### Arguments

| | |
|---|---|
| qa_log_path | Character scalar; path to the question-answering log RDS file. |
| qa_metrics_path | Character scalar; path to the metrics RDS output file. |
| output_dir | Character scalar; directory where the summary CSV and plot PNG are written. |
| judge_model | Character scalar; large language model used to score the evaluation metrics. |

### Details

All input and output locations must be supplied explicitly.

### Value

A list containing `status`, `n_qa`, `qa_metrics_path`, `summary_csv_path`, and `plot_path`.

---

## `get_openai_embeddings` *Get embeddings from OpenAI*

---

### Description

Calls the OpenAI embeddings endpoint to obtain vector representations for one or more input texts. Used by both ingestion and query-time retrieval.

### Usage

```
get_openai_embeddings(texts, model = "text-embedding-3-small", api_key = NULL)
```

### Arguments

| | |
|---|---|
| `texts` | Character vector of texts to embed. |
| `model` | Character scalar; embedding model name, e.g., `"text-embedding-3-small"`. |
| `api_key` | OpenAI API key. If NULL, uses the `OPENAI_API_KEY` environment variable. |

### Value

A numeric matrix with one row per input text.

---

## `ingest_documents` *Ingest documents into the vector store*

---

### Description

Reads PDF, DOCX, or TXT files, extracts text, cleans it lightly, chunks it, embeds chunks, and stores them in a vector-store collection.

## Usage

```
ingest_documents(
  paths,
  vectorstore_path,
  collection = "default",
  chunk_size = 500L,
  chunk_overlap = 50L,
  chunking_strategy = c("character", "sentence"),
  embedding_model = "text-embedding-3-small",
  embedding_batch_size = 128L,
  embedding_max_chars = 8000L,
  retry = TRUE,
  max_retries = 5L,
  resume = TRUE,
  checkpoint_path = NULL,
  use_openai = TRUE,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `paths` | Character vector of file paths to ingest. |
| `vectorstore_path` | Character scalar; path to the vector-store RDS file. |
| `collection` | Character scalar; name of the collection. |
| `chunk_size` | Integer; target chunk size in characters. Used when `chunking_strategy = "character"`. |
| `chunk_overlap` | Integer; overlap between consecutive chunks. Used when `chunking_strategy = "character"`. |
| `chunking_strategy` | Character; either `"character"` or `"sentence"`. |
| `embedding_model` | Character; OpenAI embedding model name. |
| `embedding_batch_size` | Integer; number of chunks per embedding request. Internally capped at 256. |
| `embedding_max_chars` | Integer; maximum number of characters retained in each chunk before embedding. |
| `retry` | Logical; whether to retry transient API failures. |
| `max_retries` | Integer; maximum number of retries for each batch. |
| `resume` | Logical; whether to use a checkpoint to skip chunks embedded during an earlier run. |
| `checkpoint_path` | Optional character scalar; path to the checkpoint RDS file. If `NULL`, a temporary file based on the collection name is used. |

| | |
|---|---|
| use_openai | Logical; if TRUE, use OpenAI embeddings. If FALSE, use the local dummy embedding generator. |
| verbose | Logical; whether to display progress messages. |

## Details

Chunking strategies:

- "character": overlapping fixed-size character windows.
- "sentence": strict sentence splitting via chunk_text_sentence().

Cleaning strategy:

- Replace carriage returns and newlines with spaces
- Remove control characters
- Collapse multiple whitespace characters into a single space
- Trim leading and trailing whitespace

Embedding strategy:

- Embeddings are computed in batches.
- Batches are retried with exponential backoff after transient failures.
- A batch is automatically divided after certain request-size failures.
- Each chunk is truncated to a specified maximum length before embedding.
- Ingestion can optionally be resumed using a local checkpoint file.

## Value

A tibble containing a per-file ingestion summary with columns collection, path, and n_chunks.

## Examples

```
document_path <- tempfile(fileext = ".txt")
writeLines(
  "Retrieval-augmented generation combines retrieval and generation.",
  document_path
)

store_path <- tempfile(fileext = ".rds")

result <- ingest_documents(
  paths = document_path,
  vectorstore_path = store_path,
  collection = "example",
  use_openai = FALSE,
  resume = FALSE,
  verbose = FALSE
)

result
```

```
vectorstore_load(store_path)
```

---

`load_qa_log` *Load a QA log from disk*

---

## Description

This helper loads a QA log tibble from an RDS file. If the file does not exist, it returns an empty QA log created by `qa_log_empty()`.

## Usage

```
load_qa_log(path)
```

## Arguments

| | |
|---|---|
| `path` | Character scalar; path to the RDS file. |

## Value

A tibble containing the QA log.

## Examples

```
path <- tempfile(fileext = ".rds")

load_qa_log(path)
```

---

`load_qa_metrics` *Load QA metrics from disk*

---

## Description

This helper loads QA metrics from an RDS file. If the file does not exist, it returns an empty metrics tibble created by `qa_metrics_empty()`.

## Usage

```
load_qa_metrics(path)
```

## Arguments

| | |
|---|---|
| `path` | Character scalar; path to the RDS file. |

## Value

A tibble containing the QA metrics.

## Examples

```
path <- tempfile(fileext = ".rds")

load_qa_metrics(path)
```

---

`load_rag_config` *Load RAG configuration*

---

## Description

Placeholder for a configuration loader. In the future, this can read a YAML file and merge it with environment variables and defaults.

## Usage

```
load_rag_config(path = NULL)
```

## Arguments

| | |
|---|---|
| `path` | Optional path to a YAML config file. Currently unused. |

## Value

A list of configuration values. Currently returns an empty list.

---

`log_rag_interaction` *Append one RAG interaction to the QA log*

---

## Description

Adds one row to an in-memory QA log tibble.

## Usage

```
log_rag_interaction(
  qa_log,
  question,
  rag_result,
  collection,
  chat_model = "gpt-4o-mini",
  embedding_model = "text-embedding-3-small",
  qa_id = NULL,
  timestamp = Sys.time()
)
```

### Arguments

| | |
|---|---|
| qa_log | Existing QA log tibble, or NULL. |
| question | Character scalar. |
| rag_result | List returned by query_rag(). Must include answer; may include retrieved and prompt. |
| collection | Collection name used for retrieval. |
| chat_model | Chat model name. |
| embedding_model | Embedding model name. |
| qa_id | Optional integer QA id. If NULL, auto-increments. |
| timestamp | POSIXct timestamp. |

### Value

Updated QA log tibble.

---

## plot_ragas_means *Plot mean RAG evaluation metrics to a PNG*

---

### Description

Creates a bar chart of mean metric values and writes it to a PNG file. The output path must be supplied explicitly.

### Usage

```
plot_ragas_means(summary_df, output_path, width = 1200, height = 700)
```

### Arguments

| | |
|---|---|
| summary_df | Output of summarize_ragas(). |
| output_path | Character scalar; path where the PNG file is written. |
| width, height | Positive numeric values giving the plot dimensions in pixels. |

### Value

Invisibly, output_path.

## Examples

```
summary_df <- tibble::tibble(
  metric = c("Context precision", "Faithfulness"),
  mean = c(0.8, 0.9)
)

output_path <- tempfile(fileext = ".png")

plot_ragas_means(
  summary_df = summary_df,
  output_path = output_path
)

file.exists(output_path)
```

---

# qa_log_empty *Create an empty QA log tibble*

---

## Description

Standard schema for storing Q/A interactions produced by the RAG pipeline.

## Usage

```
qa_log_empty()
```

## Value

A tibble with zero rows and the standard QA log columns.

---

# qa_metrics_empty *Create an empty QA metrics tibble*

---

## Description

This helper creates an empty tibble with the columns used to store RAGAS-style evaluation metrics for each QA interaction.

## Usage

```
qa_metrics_empty()
```

## Details

All metric values are expected to be in $[0, 1]$.

### Value

A tibble with zero rows and the standard QA metrics columns.

---

`query_rag` *Query the RAG pipeline*

---

### Description

Takes a user question, embeds it, retrieves relevant chunks from a vector store, constructs a grounded RAG prompt, and calls a large language model to generate an answer.

### Usage

```
query_rag(
  question,
  collection,
  vectorstore_path,
  top_k = 4L,
  embedding_model = "text-embedding-3-small",
  chat_model = "gpt-4o-mini",
  temperature = 0,
  max_output_tokens = NULL,
  score_threshold = 0,
  system_prompt = ""
)
```

### Arguments

| | |
|---|---|
| `question` | Character scalar. |
| `collection` | Vector-store collection name. |
| `vectorstore_path` | Character scalar; path to the vector-store RDS file. |
| `top_k` | Integer; number of chunks to retrieve. |
| `embedding_model` | Character; embedding model for the question. |
| `chat_model` | Character; chat model for answer generation. |
| `temperature` | Numeric; sampling temperature for the chat model. |
| `max_output_tokens` | Integer or NULL; maximum output tokens for the chat model. |
| `score_threshold` | Numeric; minimum similarity score to keep a retrieved chunk. Applied only if `retrieved` includes a `score` column. |
| `system_prompt` | Character; system instruction passed to the chat model. |

## Value

A list with:

| | |
|---|---|
| answer | Model-generated answer |
| retrieved | Tibble or data frame of retrieved chunks after filtering |
| prompt | Final grounded RAG prompt sent to the chat model |
| model | Chat model used |

---

# save_qa_log *Save a QA log to disk*

---

## Description

This helper saves a QA log tibble, as produced by log_rag_interaction(), to an RDS file. The output path must be supplied explicitly.

## Usage

```
save_qa_log(qa_log, path)
```

## Arguments

| | |
|---|---|
| qa_log | A tibble created by qa_log_empty() and populated by log_rag_interaction(). |
| path | Character scalar; path to the RDS file. |

## Value

Invisibly, the path to the saved file.

## Examples

```
qa_log <- qa_log_empty()
path <- tempfile(fileext = ".rds")

save_qa_log(qa_log, path)
file.exists(path)
```

## save_qa_metrics *Save QA metrics to disk*

### Description

This helper saves a QA metrics tibble, as produced by `compute_ragas_metrics()`, to an RDS file. The output path must be supplied explicitly.

### Usage

```
save_qa_metrics(qa_metrics, path)
```

### Arguments

| | |
|---|---|
| `qa_metrics` | A tibble created by `compute_ragas_metrics()`. |
| `path` | Character scalar; path to the RDS file. |

### Value

Invisibly, the path to the saved file.

### Examples

```
qa_metrics <- qa_metrics_empty()
path <- tempfile(fileext = ".rds")

save_qa_metrics(qa_metrics, path)
file.exists(path)
```

## summarize_ragas *Summarize RAGAS metrics across QA pairs*

### Description

Computes mean, standard deviation, minimum, and maximum for each metric column in a QA-metrics tibble.

### Usage

```
summarize_ragas(qa_metrics)
```

### Arguments

| | |
|---|---|
| `qa_metrics` | A tibble returned by `compute_ragas_metrics()` or `compute_ragas_metrics_llm()` with metric columns such as `context_precision`, `context_recall`, `answer_relevance`, `faithfulness`, and `ragas_overall`. |

## Value

A tibble with columns: `metric`, `mean`, `sd`, `min`, `max`.

---

`vectorstore_clear_all` *Delete all collections from the R-native vector store*

---

## Description

Replaces the vector store with an empty tibble. The vector-store path must be supplied explicitly.

## Usage

```
vectorstore_clear_all(path)
```

## Arguments

| | |
|---|---|
| `path` | Character scalar; path to the vector-store RDS file. |

## Value

Invisibly, `TRUE`.

## Examples

```
path <- tempfile(fileext = ".rds")

embeddings <- matrix(c(1, 0), nrow = 1)

vectorstore_upsert(
  collection = "example",
  ids = "doc_1",
  embeddings = embeddings,
  documents = "Example document",
  path = path
)

vectorstore_clear_all(path)
vectorstore_load(path)
```

`vectorstore_delete_collection` *Delete a collection from the R-native vector store*

## Description

Removes all stored records belonging to a specified collection. The vector-store path must be supplied explicitly.

## Usage

```
vectorstore_delete_collection(collection, path)
```

## Arguments

| | |
|---|---|
| `collection` | Character scalar; name of the collection to delete. |
| `path` | Character scalar; path to the vector-store RDS file. |

## Value

Invisibly, `TRUE` on success.

## Examples

```
path <- tempfile(fileext = ".rds")

embeddings <- matrix(c(1, 0), nrow = 1)

vectorstore_upsert(
  collection = "example",
  ids = "doc_1",
  embeddings = embeddings,
  documents = "Example document",
  path = path
)

vectorstore_delete_collection(
  collection = "example",
  path = path
)

vectorstore_load(path)
```

## vectorstore_load *Load the R-native vector store*

### Description

Loads the current contents of an R-native vector store from an RDS file. If the file does not exist, the function returns an empty tibble with the expected columns: `collection`, `id`, `text`, `embedding`, and `metadata`.

### Usage

```
vectorstore_load(path)
```

### Arguments

| | |
|---|---|
| `path` | Character scalar; path to the vector-store RDS file. |

### Details

The vector-store path must be supplied explicitly.

### Value

A tibble with one row per stored chunk, or zero rows if the store does not yet exist.

### Examples

```
path <- tempfile(fileext = ".rds")

vectorstore_load(path)
```

## vectorstore_query *Query the R-native vector store for nearest neighbors*

### Description

Retrieves the stored chunks whose embeddings have the highest cosine similarity to a supplied query embedding. The vector-store path must be supplied explicitly.

### Usage

```
vectorstore_query(collection, query_embedding, path, top_k = 4L)
```

### Arguments

| | |
|---|---|
| collection | Character scalar; name of the collection. |
| query_embedding | Numeric vector representing the query. |
| path | Character scalar; path to the vector-store RDS file. |
| top_k | Integer; maximum number of neighbors to retrieve. |

### Value

A tibble with columns collection, id, text, score, and metadata.

### Examples

```
path <- tempfile(fileext = ".rds")

embeddings <- matrix(
  c(
    1, 0,
    0, 1
  ),
  nrow = 2,
  byrow = TRUE
)

vectorstore_upsert(
  collection = "example",
  ids = c("doc_1", "doc_2"),
  embeddings = embeddings,
  documents = c("First document", "Second document"),
  path = path
)

vectorstore_query(
  collection = "example",
  query_embedding = c(1, 0),
  path = path,
  top_k = 1L
)
```

---

## vectorstore_upsert *Upsert embeddings into the R-native vector store*

---

### Description

Adds new records to a vector store or replaces existing records having the same collection and identifier. The vector-store path must be supplied explicitly.

## Usage

```
vectorstore_upsert(
  collection,
  ids,
  embeddings,
  documents,
  metadatas = NULL,
  path
)
```

## Arguments

| | |
|---|---|
| collection | Character scalar; name of the collection. |
| ids | Character vector of identifiers, one for each embedding. |
| embeddings | Numeric matrix or data frame; one row per identifier. |
| documents | Character vector of document text, one element per embedding. |
| metadatas | Optional list or data frame containing metadata for each row. |
| path | Character scalar; path to the vector-store RDS file. |

## Value

Invisibly, TRUE on success.

## Examples

```
path <- tempfile(fileext = ".rds")

embeddings <- matrix(
  c(
    1, 0,
    0, 1
  ),
  nrow = 2,
  byrow = TRUE
)

vectorstore_upsert(
  collection = "example",
  ids = c("doc_1", "doc_2"),
  embeddings = embeddings,
  documents = c("First document", "Second document"),
  path = path
)

vectorstore_load(path)
```

# Index