Package 'statcanR'

July 25, 2026

Type Package

Title Client for Statistics Canada's Open Economic Data

Version 0.3.9

Description Provides an R client for Statistics Canada's Web Data Service. Users can describe the data they need in natural language, search the official table catalogue, and download complete data tables in English or French as data frames. Tables formerly known as CANSIM tables are identified by Product IDs. Warin (2024) <doi:10.5070/T5.1868>.

Depends R (>= 4.0.0)

License MIT + file LICENSE

Encoding UTF-8

Language en-US

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

VignetteBuilder knitr

Imports data.table, jsonlite, DT, httr

URL https://warint.github.io/statcanR/, https://github.com/warint/statcanR

BugReports https://github.com/warint/statcanR/issues

Config/testthat/edition 3

Config/Needs/website pkgdown

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Thierry Warin [aut, cre] (ORCID: <https://orcid.org/0000-0002-5921-3428>)

Maintainer Thierry Warin <thierry.warin@hec.ca>

Repository CRAN

Date/Publication 2026-07-25 05:10:02 UTC

Contents

`statcan_chat` *Get an LLM's help interpreting a natural-language table search*

Description

Sends the query and the ranked candidates from `statcan_find()` to a user-configured language-model provider, which explains which candidate(s) best match and asks a clarifying question when the query is ambiguous. The candidate table numbers and rankings always come from `statcan_find()` itself; the language model only interprets and explains them, and is never allowed to propose a table number of its own.

Usage

```
statcan_chat(
  query,
  lang = c("eng", "fra"),
  n = 5L,
  refresh = FALSE,
  endpoint = NULL,
  api_key = NULL,
  model = NULL,
  provider = c("openai", "anthropic")
)
```

Arguments

`query`	One non-empty character string describing the desired data. Passed to `statcan_find()`.
`lang`	Language of the table titles and of the model's reply: `"eng"` or `"fra"`.
`n`	Maximum number of candidates to request from `statcan_find()`.
`refresh`	Logical; forwarded to `statcan_find()`.
`endpoint`	Provider endpoint URL. Defaults to `getOption("statcanR.llm_endpoint")`, then `Sys.getenv("STATCANR_LLM_ENDPOINT")`, then the chosen provider's own endpoint. Must be `https://`, except for loopback hosts (for example, `http://localhost`).
`api_key`	API key. Defaults to `Sys.getenv("STATCANR_LLM_API_KEY")`, then the provider's native variable (`OPENAI_API_KEY` for `"openai"`, `ANTHROPIC_API_KEY` for `"anthropic"`). For safety it is not read from `options()`. It is sent as an `Authorization: Bearer` header for `"openai"` and as an `x-api-key` header for `"anthropic"`.

model	Model name sent to the provider (for example, "gpt-4o-mini" for OpenAI or "claude-opus-4-8" for Anthropic). Defaults to getOption("statcanR.llm_model"), then Sys.getenv("STATCANR_LLM_MODEL").
provider	Which LLM provider to use: "openai" (the default, also covering OpenAI-compatible and local servers) or "anthropic" (Claude).

Details

Two providers ship built in, selected with the provider argument:

- "openai" (the default): the OpenAI chat-completions format, using an Authorization: Bearer API key. This also covers any OpenAI-compatible server – Groq, Together, OpenRouter, Mistral, vLLM, or a local open-source model served by Ollama or LM Studio – by pointing endpoint at it (a loopback http://localhost endpoint is accepted for local models; pass any placeholder api_key for servers that ignore it).
- "anthropic": the Claude Messages format, using an x-api-key header.

This is an optional feature. It requires no additional packages beyond what statcanR already imports, but it does require you to configure an API key and model (the endpoint defaults to the chosen provider), either as arguments or through options() / environment variables:

- endpoint: options(statcanR.llm_endpoint = ...) or Sys.setenv(STATCANR_LLM_ENDPOINT = ...). Defaults to the provider's own endpoint when unset.
- api_key: Sys.setenv(STATCANR_LLM_API_KEY = ...) (or the api_key argument, or the provider's native variable – OPENAI_API_KEY / ANTHROPIC_API_KEY). Because it is a secret, the key is **not** read from options(), which can be dumped, saved with a session, or recorded in .Rhistory.
- model: options(statcanR.llm_model = ...) or Sys.setenv(STATCANR_LLM_MODEL = ...)

The endpoint must use https:// so the key is never sent in cleartext; plain http:// is accepted only for loopback hosts (for example, http://localhost for a local model).

No network request is made unless statcan_chat() is called directly.

Value

A statcan_chat_result object: a list with query, candidates (the statcan_find() data frame), explanation, and clarifying_question (NA when the model had none).

Examples

```
## Not run:
# OpenAI (the default provider)
Sys.setenv(OPENAI_API_KEY = "sk-...")
result <- statcan_chat(
  "R&D expenditures in Quebec since 2020",
  model = "gpt-4o-mini"
)

# Anthropic (Claude)
Sys.setenv(ANTHROPIC_API_KEY = "sk-ant-...")
```

```
result <- statcan_chat(
  "R&D expenditures in Quebec since 2020",
  provider = "anthropic", model = "claude-opus-4-8"
)

# A local open-source model served by Ollama (no key over loopback http)
result <- statcan_chat(
  "R&D expenditures in Quebec since 2020",
  endpoint = "http://localhost:11434/v1/chat/completions",
  api_key = "ollama", model = "llama3.1"
)

# statcan_chat() returns several ranked candidates, not a single table:
# the model explains them but never picks or invents one for you.
# The candidates are already a data frame, so you never retype an id.
result$candidates          # the full statcan_find() data frame
result$candidates$id       # every candidate id, best-ranked first
result$candidates$id[1]    # just the top-ranked id
result$explanation         # the model's plain-language explanation

# Feed the chosen id straight into statcan_data() -- nothing copied by hand.
table_data <- statcan_data(result$candidates$id[1], lang = "eng")

## End(Not run)
```

`statcan_chat_continue` *Continue a* `statcan_chat()` *conversation*

Description

Sends a follow-up `message` – typically an answer to the `clarifying_question` from a previous turn – and returns an updated result. The follow-up stays scoped to the **same candidate tables** that `statcan_chat()` already found: it never re-runs `statcan_find()` and the model still never proposes a table number of its own. To search the catalogue again, start a new conversation with `statcan_chat()`.

Usage

```
statcan_chat_continue(result, message, api_key = NULL)
```

Arguments

`result`	A `statcan_chat_result` returned by `statcan_chat()` (or by a previous `statcan_chat_continue()` call).
`message`	One non-empty character string: your follow-up to the model.
`api_key`	API key, re-resolved exactly as in `statcan_chat()`. Defaults to `Sys.getenv("STATCANR_LLM_API_KEY")` then the provider's native variable. Not read from `options()`.

Details

The returned object is itself continuable, so you can chain several follow-ups. The API key is re-resolved on each call (from the `api_key` argument, the `STATCANR_LLM_API_KEY` environment variable, or the provider's native variable) because, for safety, it is never stored in the result object.

Value

An updated `statcan_chat_result` with the same `candidates`, a new `explanation` and `clarifying_question`, and an extended conversation.

Examples

```
## Not run:
Sys.setenv(ANTHROPIC_API_KEY = "sk-ant-...")
chat <- statcan_chat(
  "R&D spending in Quebec",
  provider = "anthropic", model = "claude-opus-4-8"
)
chat$clarifying_question

# Answer it and keep the same shortlist of candidates. Reassigning the same
# variable keeps the latest turn; you never need r1/r2/r3-style names.
chat <- statcan_chat_continue(chat, "annual data, since 2015")
chat$explanation

# Chain another follow-up the same way:
chat <- statcan_chat_continue(chat, "just the total, not by industry")

## End(Not run)
```

`statcan_data` *Download a Statistics Canada data table*

Description

Downloads a complete table from Statistics Canada's Web Data Service (WDS) and returns it as a data frame. Product IDs can be supplied in the familiar hyphenated form (for example, `"27-10-0014-01"`) or as an eight-digit PID (for example, `"27100014"`).

Usage

```
statcan_data(tableNumber, lang)
```

Arguments

`tableNumber`	A Statistics Canada table number or Product ID. Both `"27-10-0014-01"` and `"27100014"` are accepted.
`lang`	Language of the downloaded table: `"eng"` or `"fra"`.

Details

The function keeps the interface used by earlier statcanR releases. English and French downloads share the same processing rules. In particular, the first column is named `REF_DATE`, coordinates are stored as character, and the table title from the metadata file is added as `INDICATOR`.

Value

A data frame containing the complete Statistics Canada table.

Examples

```
## Not run:
science <- statcan_data("27-10-0014-01", "eng")
science_fr <- statcan_data("27100014", "fra")

## End(Not run)
```

`statcan_download_data` *Download a Statistics Canada table and save a CSV file*

Description

Calls `statcan_data()` and also writes the returned table to a CSV file. Existing calls with only `tableNumber` and `lang` remain supported; `path` can be used to select a different output directory.

Usage

```
statcan_download_data(tableNumber, lang, path = ".")
```

Arguments

`tableNumber`	A Statistics Canada table number or Product ID. Both `"27-10-0014-01"` and `"27100014"` are accepted.
`lang`	Language of the downloaded table: `"eng"` or `"fra"`.
`path`	Directory in which to save the CSV file. The directory must already exist. Defaults to the current working directory.

Value

The downloaded table. The CSV path is available in the `statcan_file` attribute of the returned data frame.

Examples

```
## Not run:
science <- statcan_download_data("27-10-0014-01", "eng")
science <- statcan_download_data(
  "27-10-0014-01", "eng", path = tempdir()
)
attr(science, "statcan_file")

## End(Not run)
```

`statcan_find` *Find Statistics Canada tables using a natural-language query*

Description

Interprets a short description of the data needed and returns ranked Statistics Canada table candidates. The query can contain a topic, Canadian geography, and reference year or range. For example, `"R&D expenditures in Quebec since 2020"` is interpreted as a research and development expenditure topic, a Quebec geography constraint, and coverage beginning in 2020.

Usage

```
statcan_find(query, lang = c("eng", "fra"), n = 5L, refresh = FALSE)
```

Arguments

`query`	One non-empty character string describing the desired data.
`lang`	Language of the table titles: `"eng"` or `"fra"`.
`n`	Maximum number of candidates to return, from 1 to 20.
`refresh`	Logical; if `TRUE`, request a fresh catalogue and fresh candidate metadata from WDS.

Details

This function finds tables; it does not download or filter their observations. Use the returned `id` with `statcan_data()`, then apply any required geography and date filters to the downloaded data. Rankings are a discovery aid, so review the candidate title before downloading a large table.

The catalogue is cached for 24 hours. When a geography is present in the query, metadata for a small set of leading candidates is retrieved through WDS to confirm that the geography is a table member. That metadata is cached for seven days. If metadata is temporarily unavailable, candidates can still be returned with `geography_match = NA`.

Value

A data frame of ranked candidates. `id` contains the table number, `score` is the relevance score, and `match_reason` explains the ranking. Coverage dates describe the table as a whole. `geography_match` is `TRUE` when WDS metadata confirms every geography in the query, `FALSE` when it does not, and `NA` when no geography was requested or validation was not possible.

Examples

```
## Not run:
matches <- statcan_find(
  "R&D expenditures in Quebec since 2020",
  lang = "eng"
)
matches[, c("title", "id", "match_reason")]

## End(Not run)
```

`statcan_search` *Search Statistics Canada data tables*

Description

Searches the catalogue of tables published through Statistics Canada's Web Data Service (WDS). The catalogue is cached for 24 hours in the user's cache directory. If WDS is temporarily unavailable, the most recent valid cache is used.

Usage

```
statcan_search(keywords, lang = c("eng", "fra"), refresh = FALSE)
```

Arguments

`keywords`	Character vector of words that must appear in the table title. Matching is case-insensitive.
`lang`	Language of the returned titles: `"eng"` or `"fra"`.
`refresh`	Logical; if `TRUE`, ignore a fresh cache and request the catalogue from WDS.

Value

A `DT::datatable` containing matching table titles, Product IDs, release dates, and language.

Examples

```
## Not run:
statcan_search(c("economy", "export"), "eng")
statcan_search("population", "fra")

## End(Not run)
```

Index