Package 'sparqlr'

April 8, 2026

Title A SPARQL Client for R

Version 0.1.0

Description Provides a client for running SPARQL queries directly from R. SPARQL (short for SPARQL Protocol and RDF Query Language) is a query language used to retrieve and manipulate data stored in RDF (Resource Description Framework) format.

License GPL-3

URL https://github.com/sib-swiss/sparqlr

BugReports https://github.com/sib-swiss/sparqlr/issues

Depends R (>= 4.1.0)

Imports dplyr, httr2, purrr, rlang, stringr, tibble, tidyr

Suggests knitr, lintr, rcmdcheck, rmarkdown, styler, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Marco Pagni [aut] (ORCID: <https://orcid.org/0000-0001-9292-9463>), Robin Engler [aut, cre] (ORCID: <https://orcid.org/0009-0001-7090-6196>), SIB Swiss Institute of Bioinformatics [cph]

Maintainer Robin Engler <robin.engler@sib.swiss>

Repository CRAN

Date/Publication 2026-04-08 14:10:08 UTC

Contents

`convert_iri_style` *Convert the style of IRIs from "long" or "short" into another form.*

Description

Convert the style of IRIs in all columns of a tibble.

Usage

```
convert_iri_style(
  t,
  prefixes,
  iri_style = c("short", "long", "mdlink", "html"),
  replace_in_literal = FALSE
)
```

Arguments

`t`	Tibble whose IRIs are to be modified.
`prefixes`	Tibble with "short" and "long" versions of the prefixes for which the IRI style should be modified.
`iri_style`	One of "short", "mdlink", "html" or "long".
`replace_in_literal`	If TRUE, then replacement is also done within literal strings, not just in IRIs.

Value

An copy of the input tibble `t` where the IRI style was modified.

Examples

```
# Create a sample tibble with long-form IRIs.
data <- tibble::tibble(
  composer = c(
    "<http://www.wikidata.org/entity/Q254>",
    "<http://www.wikidata.org/entity/Q255>",
    "<http://www.wikidata.org/entity/Q1268>"
```

```
  ),
  country = c(
    "<http://www.wikidata.org/entity/Q701614>",
    "<http://www.wikidata.org/entity/Q131964>",
    "<http://www.wikidata.org/entity/Q34266>"
  ),
  label = c(
    "Wolfgang Amadeus Mozart - http://example.org/Mozart",
    "Ludwig van Beethoven - http://example.org/Beethoven",
    "Frederic Chopin - http://example.org/Chopin)"
  )
)

# Define prefixes long-to-short correspondence.
prefixes <- tibble::tibble(
  short = c("wde", "ex"),
  long = c("http://www.wikidata.org/entity/", "http://example.org/")
)

# Convert IRIs to short form.
convert_iri_style(data, prefixes, iri_style = "short")

# Convert IRIs to markdown link format.
convert_iri_style(data, prefixes, iri_style = "mdlink")

# Convert IRIs to HTML link format.
convert_iri_style(data, prefixes, iri_style = "html")

# Also apply conversion to string literals.
convert_iri_style(
  data, prefixes,
  iri_style = "short",
  replace_in_literal = TRUE
)

# Apply different IRI styles to different columns.
convert_iri_style(
  data, prefixes,
  iri_style = c("short", "short", "mdlink"),
  replace_in_literal = TRUE
)
```

`load_prefixes_from_file` *Load SPARQL prefixes from a text file.*

Description

Extracts the list of PREFIXes from a SPARQL query file, and returns it as a tibble object.

Usage

```
load_prefixes_from_file(path)
```

Arguments

path	Path (string) of the SPARQL file from which to load prefixes.

Value

A tibble with short and long columns.

Examples

```
# Load a query file
file_path <- system.file("extdata", "example_select.rq", package = "sparqlr")
prefixes <- load_prefixes_from_file(file_path)
```

---

load_query_from_file *Load a SPARQL query from a file.*

Description

Reads a SPARQL query from a file and returns it as a single multi-line string. Optionally removes comment lines (lines starting with a '#' character).

Usage

```
load_query_from_file(path, remove_comments = FALSE)
```

Arguments

path	String specifying the path to the SPARQL query file.
remove_comments	Boolean. If TRUE, removes lines that start with a comment character '#'. Defaults to FALSE.

Value

A character string containing the loaded SPARQL query.

Examples

```
# Load a query file
file_path <- system.file("extdata", "example_select.rq", package = "sparqlr")
query <- load_query_from_file(file_path, remove_comments = TRUE)
cat(query)
```

`sparql_construct` *Run a SPARQL CONSTRUCT query.*

Description

Executes a SPARQL CONSTRUCT query and returns the results as a list containing two tibbles: one for edges (triples with IRIs) and one for nodes with properties. IRIs can be formatted in different styles using the `iri_style` argument.

Usage

```
sparql_construct(
  endpoint,
  query,
  prefixes = NULL,
  request_method = c("POST", "GET"),
  http_extra_params = list(),
  verbose = FALSE
)
```

Arguments

`endpoint`	URL of SPARQL endpoint.
`query`	SPARQL query as a string.
`prefixes`	Optional data frame whose first two columns are taken for short and long versions of base IRIs.
`request_method`	HTTP method to use to submit the request.
`http_extra_params`	Additional parameter/value pair to pass to the HTTP request. E.g. some endpoints accept a "timeout" argument, which could be passed via this argument.
`verbose`	If `TRUE`, print query execution time.

Value

A list with two tibbles: `edges` and `nodes`.

Examples

```
# Load a SPARQL construct query from an example file.
query <- system.file(
  "extdata", "example_construct.rq",
  package = "sparqlr"
) |> load_query_from_file()

# Run the SPARQL query on the specified endpoint.
sparql_construct(
```

```
  endpoint = "https://query.wikidata.org/sparql",
  query = query
)
```

`sparql_describe` *Run a SPARQL DESCRIBE query.*

Description

Executes a SPARQL DESCRIBE query and returns the results tibble with 3 columns: subject, predicate, object.

IRIs can be formatted in different styles using the `iri_style` argument.

Usage

```
sparql_describe(
  endpoint,
  query,
  prefixes = NULL,
  request_method = c("POST", "GET"),
  http_extra_params = list(),
  verbose = FALSE
)
```

Arguments

`endpoint`	URL of SPARQL endpoint.
`query`	SPARQL query as a string.
`prefixes`	Optional data frame whose first two columns are taken for short and long versions of base IRIs.
`request_method`	HTTP method to use to submit the request.
`http_extra_params`	Additional parameter/value pair to pass to the HTTP request. E.g. some endpoints accept a "timeout" argument, which could be passed via this argument.
`verbose`	If `TRUE`, print query execution time.

Value

```
A list with two tibbles: `edges` and `nodes`.
```

Examples

```
query <- "
# Retrieve all 'useful descriptions' associated with the 'cats' resource.
DESCRIBE <http://www.wikidata.org/entity/Q146>
"
endpoint <- "https://query.wikidata.org/sparql"

# Run the SPARQL query on the specified endpoint.
sparql_describe(endpoint, query)

# Same as above, but converting prefixes to their short form.
prefixes <- tibble::tibble(
  short = c("wd", "w3"),
  long = c(
    "http://www.wikidata.org/entity/",
    "http://www.w3.org/2000/01/rdf-schema#"
  )
)
sparql_describe(endpoint, query, prefixes = prefixes)
```

`sparql_select` *Run a SPARQL SELECT query.*

Description

Run a SPARQL query, either SELECT, CONSTRUCT or DESCRIBE and return the results as a tibble. Returned column names are the same as SPARQL variables. Detection of column types relies on R built-in methods, not RDF data types.

In the HTTP request, the "application/sparql-results+json" MIME type is used, which is supported by most SPARQL endpoints.

Usage

```
sparql_select(
  endpoint,
  query,
  prefixes = NULL,
  request_method = c("POST", "GET"),
  http_extra_params = list(),
  verbose = FALSE
)
```

Arguments

`endpoint`	URL of SPARQL endpoint.
`query`	SPARQL query as a string.

prefixes	Optional data frame whose first two columns are taken for short and long versions of base IRIs.
request_method	HTTP method to use to submit the request.
http_extra_params	Additional parameter/value pair to pass to the HTTP request. E.g. some endpoints accept a "timeout" argument, which could be passed via this argument.
verbose	If TRUE, print query execution time.

Value

A tibble with the query results or NULL if the query returns nothing.

Examples

```
# Define the SPARQL query to run.
query <- "
# Query classical music composers and their country of origin.
SELECT ?composer ?composerLabel ?country ?countryLabel WHERE {

  ?composer wdt:P106 wd:Q36834 ;     # Occupation: composer
            wdt:P101  wd:Q9730 .     # Field of work: classical music

  OPTIONAL { ?composer wdt:P27 ?country . }  # country of citizenship

  SERVICE wikibase:label {
    bd:serviceParam wikibase:language '[AUTO_LANGUAGE],en' .
  }
}
"

# Run the SPARQL query on the specified endpoint.
sparql_select(
  endpoint = "https://query.wikidata.org/sparql",
  query = query
)

# Same as above, but converting prefixes to their short form.
prefixes <- tibble::tibble(
  short = c("wde"),
  long = c("http://www.wikidata.org/entity/")
)
sparql_select(
  endpoint = "https://query.wikidata.org/sparql",
  query = query,
  prefixes = prefixes
)
```

Index