Package 'SLIMER'

October 15, 2024

Type Package

Title A package to retrieve trophic information from SLIME, the semantic Soil-Life Metaweb

Version 1.0.0

Author Author 1 <email@mail.com>

Maintainer Author 1 <email@mail.com>

Description The SLIMER package provides functions for executing SPARQL queries and retrieving trophic information from the SLIME knowledge graph.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Config/Needs/website usethis, pkgdown

RoxygenNote 7.3.1

Imports stringr,
purrr,
dplyr,
httr2,
anytime,
tibble,
memoise,
rlang

Depends R (>= 4.3)

Suggests knitr,
rmarkdown,
pkgdown

VignetteBuilder knitr

Contents

`add_prefixes` *Add SPARQL prefixes to a query.*

Description

The 'add_prefixes' function is designed to add SPARQL prefixes to a given query. It takes a query as input and prefixes it with the specified namespace prefixes.

Usage

```
add_prefixes(query)
```

Arguments

`query`	A character string representing the SPARQL query to which prefixes need to be added.

Details

The function utilizes an inner function 'format.namespace' to format each namespace prefix and its corresponding URI as a SPARQL PREFIX statement. The prefixes are defined in a global variable named 'prefixes'. The format of the global variable should be a named list where each name corresponds to a prefix, and the corresponding value is the URI.

Value

A character string, which is the input query with SPARQL prefixes added.

Author(s)

<Author 1>

Examples

```
# Define namespace prefixes
prefixes <- list(
  ex = "http://example.org/",
  foaf = "http://xmlns.com/foaf/0.1/",
  rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
)

# Example query
query <- "SELECT ?person WHERE { ?person foaf:name ?name . }"

# Add prefixes to the query
result_query <- add_prefixes(query)

# Print the result
cat(result_query)
```

`fill.query` *Fill in placeholders in a SPARQL query based on provided parameters.*

Description

The ‘fill.query‘ function replaces placeholders in a SPARQL query with specified values based on taxonomy ID or scientific name. It utilizes the ‘get.sparql.values‘, ‘get.named.graphs‘, ‘query.by.name‘, and ‘query.by.taxid‘ functions.

Usage

```
fill.query(query, taxid = NULL, sciName = NULL)
```

Arguments

`query`	A character string representing the SPARQL query with placeholders.
`taxid`	An optional parameter representing the Taxonomy ID to be used in the query.
`sciName`	An optional parameter representing the scientific name to be used in the query.

Details

The function takes a SPARQL query with placeholders and optional parameters for Taxonomy ID and scientific name. It uses the ‘get.sparql.values‘ function to generate VALUES clauses for the provided parameters and replaces the placeholders in the query using ‘sprintf‘. The function also includes information from ‘get.named.graphs‘, ‘query.by.name‘, and ‘query.by.taxid‘ in the resulting query.

Value

A character string representing the filled-in SPARQL query with specified values.

Author(s)

<Author 1>

Examples

```
# Example for filling in placeholders in a SPARQL query
sparql_query_template <- "SELECT * WHERE { %s }"
filled_query <- fill.query(sparql_query_template, taxid = c("GBIF:2130185", "NCBI:55786"))
cat(filled_query)
```

`format.result.df` *Format and process a data frame of results.*

Description

The ‘format.result.df‘ function transforms IRIs in a data frame to their short, prefixed form using the ‘to.short.iris‘ function. It also groups matching taxids into a list if specified, and then deduplicates the data frame.

Usage

```
## S3 method for class 'result.df'
format(df, iri.cols, with.matchId = TRUE, with.reference = TRUE)
```

Arguments

`df`	A data frame containing query results.
`iri.cols`	A vector of character strings specifying the columns containing IRIs to be transformed.
`with.matchId`	Logical, indicating whether to group matching taxids into a list.
`with.reference`	Logical, indicating whether to group bibliographic references into a list.

Details

The function transforms specified columns containing IRIs to their short, prefixed form using the ‘to.short.iris‘ function. It also provides an option to group matching taxids into a list if the ‘with.matchId‘ parameter is set to TRUE. The resulting data frame is then deduplicated.

Value

A modified data frame with transformed IRIs, grouped taxids, and deduplicated rows.

Author(s)

<Author 1>

Examples

```
# Example for formatting and processing a data frame of results
result_df <- data.frame(matchId = c(1, 1, 2, 3),
                iri_col = c("http://example.org/1", "http://example.org/2", "http://example.org/3", "http://exam
formatted_df <- format.result.df(result_df, iri.cols = "iri_col", with.matchId = TRUE, with.reference=FALSE)
print(formatted_df)
```

`get.diets` *Retrieve diet information based on Taxonomy ID or scientific name from a specific SPARQL endpoint.*

Description

The ‘get.diets‘ function constructs and executes a SPARQL query to retrieve diet information based on specified Taxonomy ID or scientific name. It utilizes the ‘fill.query‘, ‘send.sparql‘, and ‘format.result.df‘ functions.

Usage

```
get.diets(taxid = NULL, sciName = NULL, endpoint)
```

Arguments

`taxid`	An optional parameter representing the Taxonomy ID(s) to be used in the query.
`sciName`	An optional parameter representing the scientific name(s) to be used in the query.
`endpoint`	A character string representing the SPARQL endpoint to send the query to.

Details

The function constructs a SPARQL query template, fills in placeholders based on provided parameters using the ‘fill.query‘ function, sends the query to the specified SPARQL endpoint using ‘send.sparql‘, and formats the result data frame using ‘format.result.df‘. The resulting data frame contains the following information: query name, query ID, match name, match ID, diet ID, diet name, reference, source, and inferred.

Value

A data frame containing diet information based on the specified parameters.

Author(s)

<Author 1>

Examples

```
# Example for retrieving diet information based on Taxonomy ID
diet_info <- get.diets(taxid = c("GBIF:2130185", "NCBI:55786"), endpoint = "http://129.88.204.79:7200/repositorie
print(diet_info)
```

---

get.guilds — *Retrieve guild information based on Taxonomy ID or scientific name from a specific SPARQL endpoint.*

Description

The ‘get.guilds‘ function constructs and executes a SPARQL query to retrieve guild information based on specified Taxonomy ID or scientific name. It utilizes the ‘fill.query‘, ‘send.sparql‘, and ‘format.result.df‘ functions.

Usage

```
get.guilds(taxid = NULL, sciName = NULL, endpoint)
```

Arguments

`taxid`	An optional parameter representing the Taxonomy ID(s) to be used in the query.
`sciName`	An optional parameter representing the scientific name(s) to be used in the query.
`endpoint`	A character string representing the SPARQL endpoint to send the query to.

Details

The function constructs a SPARQL query template, fills in placeholders based on provided parameters using the ‘fill.query‘ function, sends the query to the specified SPARQL endpoint using ‘send.sparql‘, and formats the result data frame using ‘format.result.df‘. The resulting data frame contains the following information: query name, query ID, match name, match ID, guild ID, guild name, reference, source, and inferred.

Value

A data frame containing guild information based on the specified parameters.

Author(s)

<Author 1>

Examples

```
# Example for retrieving guild information based on Taxonomy ID from a specific SPARQL endpoint
guild_info <- get.guilds(taxid = c("GBIF:2130185", "NCBI:55786"), endpoint = "http://129.88.204.79:7200/repositor
print(guild_info)
```

`get.interactions` *Retrieve interaction information based on Taxonomy ID or scientific name from a specific SPARQL endpoint.*

Description

The ‘get.interactions‘ function constructs and executes a SPARQL query to retrieve interaction information based on specified Taxonomy ID or scientific name. It utilizes the ‘fill.query‘, ‘send.sparql‘, and ‘format.result.df‘ functions.

Usage

```
get.interactions(taxid = NULL, sciName = NULL, endpoint = NULL)
```

Arguments

`taxid`	An optional parameter representing the Taxonomy ID(s) to be used in the query.
`sciName`	An optional parameter representing the scientific name(s) to be used in the query.
`endpoint`	A character string representing the SPARQL endpoint to send the query to.

Details

The function constructs a SPARQL query template, fills in placeholders based on provided parameters using the ‘fill.query‘ function, sends the query to the specified SPARQL endpoint using ‘send.sparql‘, and formats the result data frame using ‘format.result.df‘. The resulting data frame contains the following information: query name, query ID, match name, match ID, interaction ID, interaction name, resource ID, resource name, reference, source, and inferred.

Value

A data frame containing interaction information based on the specified parameters.

Author(s)

<Author 1>

Examples

```
# Example for retrieving interaction information based on Taxonomy ID from a specific SPARQL endpoint
interaction_info <- get.interactions(taxid = c("GBIF:2130185", "NCBI:55786"), endpoint = "http://129.88.204.79:72
print(interaction_info)
```

`get.named.graphs` *Generate a SPARQL FROM NAMED clause for named graphs.*

Description

The ‘get.named.graphs‘ function generates a SPARQL FROM NAMED clause with specified named graphs. The resulting SPARQL snippet can be used to specify named graphs for a query.

Usage

```
get.named.graphs()
```

Details

The function returns a static SPARQL snippet with FROM NAMED clauses for three example named graphs.

Value

A character string representing the SPARQL FROM NAMED clause with specified named graphs.

Author(s)

<Author 1>

Examples

```
# Example for generating a SPARQL FROM NAMED clause
named_graphs <- get.named.graphs()
cat(named_graphs)
```

`get.potential.interactions` *Retrieve potential interaction information based on Taxonomy ID or scientific name from a specific SPARQL endpoint.*

Description

The ‘get.potential.interactions‘ function constructs and executes a SPARQL query to retrieve potential interaction information based on specified Taxonomy ID or scientific name. It utilizes the ‘fill.query‘, ‘send.sparql‘, and ‘format.result.df‘ functions.

Usage

```
get.potential.interactions(taxid = NULL, sciName = NULL, endpoint = NULL)
```

Arguments

`taxid`	An optional parameter representing the Taxonomy ID(s) to be used in the query.
`sciName`	An optional parameter representing the scientific name(s) to be used in the query.
`endpoint`	A character string representing the SPARQL endpoint to send the query to.

Details

Potential interactions are relationships between a consumer and potential trophic resources which are inferred from its diet, e.g. a bacterivorous organism potentially interacts with Bacteria and Archaea.

The function constructs a SPARQL query template, fills in placeholders based on provided parameters using the ‘fill.query‘ function, sends the query to the specified SPARQL endpoint using ‘send.sparql‘, and formats the result data frame using ‘format.result.df‘. The resulting data frame contains the following information: query name, query ID, match name, match ID, interaction ID, interaction name, resource ID, resource name, reference, source, and inferred.

Value

A data frame containing potential interaction information based on the specified parameters.

Author(s)

<Author 1>

Examples

```
# Example for retrieving potential interaction information based on Taxonomy ID from a specific SPARQL endpoint
interaction_info <- get.potential.interactions(taxid = c("GBIF:2130185", "NCBI:55786"), endpoint = "http://129.88
print(interaction_info)
```

`get.source.metadata` *Retrieve metadata for a specified data source from a specific SPARQL endpoint.*

Description

The ‘get.source.metadata‘ function constructs and executes a SPARQL query to retrieve metadata for a specified data source. It utilizes the ‘get.sparql.values‘, ‘send.sparql‘, and ‘format.result.df‘ functions.

Usage

```
get.source.metadata(source, endpoint = NULL)
```

Arguments

`source`	A character string representing the shortened IRI of the data source for which metadata is to be retrieved.
`endpoint`	A character string representing the SPARQL endpoint to send the query to.

Details

The function constructs a SPARQL query template, fills in placeholders based on provided parameters using the ‘get.sparql.values‘ function, sends the query to the specified SPARQL endpoint using ‘send.sparql‘, and formats the result data frame using ‘format.result.df‘. The resulting data frame contains metadata information such as field and value for the specified data source.

Value

A data frame containing metadata for the specified data source.

Author(s)

<Author 1>

Examples

```
# Example for retrieving metadata for a specific data source from a SPARQL endpoint
source_metadata <- get.source.metadata(source = "gratin:nematrait", endpoint = "http://129.88.204.79:7200/reposi
print(source_metadata)
```

`get.sources` *Retrieve a list of data sources from a specific SPARQL endpoint.*

Description

The ‘get.sources‘ function constructs and executes a SPARQL query to retrieve a list of data sources from a specified SPARQL endpoint. It utilizes the ‘request‘, ‘req_retry‘, ‘req_perform‘, ‘resp_check_status‘, ‘read.table‘, and ‘format.result.df‘ functions.

Usage

```
get.sources(endpoint = NULL)
```

Arguments

`endpoint`	A character string representing the SPARQL endpoint to send the query to.

Details

The function constructs a SPARQL query to obtain the list of data sources, sends the query to the specified SPARQL endpoint using ‘request‘, ‘req_retry‘, and ‘req_perform‘, checks the response status with ‘resp_check_status‘, reads the response body with ‘read.table‘, and formats the result data frame using ‘format.result.df‘. The resulting data frame contains information about data sources, specifically the source shortened IRIs.

Value

A data frame containing a list of data sources available at the specified SPARQL endpoint.

Author(s)

<Author 1>

Examples

```
# Example for retrieving a list of data sources from a SPARQL endpoint
sources_list <- get.sources(endpoint = "http://129.88.204.79:7200/repositories/gratin")
print(sources_list)
```

`get.sparql.values` *Generate a SPARQL VALUES clause for a variable with specified values.*

Description

The ‘get.sparql.values‘ function generates a SPARQL VALUES clause for a specified variable with associated values. The resulting SPARQL snippet can be used to filter query results based on the specified variable values.

Usage

```
get.sparql.values(var.name, var.values, is.iri = TRUE)
```

Arguments

`var.name`	A character string representing the variable name for which values are specified.
`var.values`	A vector of values for the specified variable.
`is.iri`	Logical, indicating whether the values are IRIs. If FALSE, the values are wrapped in double quotes.

Details

If ‘is.iri‘ is set to FALSE, the function wraps each value in double quotes. The resulting VALUES clause is then constructed and returned as a character string.

Value

A character string representing the SPARQL VALUES clause for the specified variable and values.

Author(s)

<Author 1>

Examples

```
# Example for generating a SPARQL VALUES clause
var_name <- "country"
var_values <- c("USA", "Canada", "UK")
sparql_values <- get.sparql.values(var_name, var_values, is.iri = FALSE)
cat(sparql_values)
```

`get.trophic.groups` *Retrieve trophic group information based on Taxonomy ID or scientific name from a specific SPARQL endpoint.*

Description

The ‘get.trophic.groups‘ function constructs and executes a SPARQL query to retrieve trophic group information based on specified Taxonomy ID or scientific name. It utilizes the ‘fill.query‘, ‘send.sparql‘, and ‘format.result.df‘ functions.

Usage

```
get.trophic.groups(taxid = NULL, sciName = NULL, endpoint = NULL)
```

Arguments

`taxid`	An optional parameter representing the Taxonomy ID(s) to be used in the query.
`sciName`	An optional parameter representing the scientific name(s) to be used in the query.
`endpoint`	A character string representing the SPARQL endpoint to send the query to.

Details

The function constructs a SPARQL query template, fills in placeholders based on provided parameters using the ‘fill.query‘ function, sends the query to the specified SPARQL endpoint using ‘send.sparql‘, and formats the result data frame using ‘format.result.df‘. The resulting data frame contains the following information: query name, query ID, match name, match ID, trophic group ID, trophic group name, reference, source, and inferred.

Value

A data frame containing trophic group information based on the specified parameters.

Author(s)

<Author 1>

Examples

```
# Example for retrieving trophic group information based on Taxonomy ID from a specific SPARQL endpoint
trophic_group_info <- get.trophic.groups(taxid = c("GBIF:2130185", "NCBI:55786"), endpoint = "http://129.88.204.7
print(trophic_group_info)
```

`query.by.name` *Generate a static SPARQL snippet for retrieving information by taxon scientific name.*

Description

The ‘query.by.name‘ function generates a static SPARQL snippet for retrieving information based on a taxon scientific name.

Usage

```
query.by.name()
```

Value

A character string representing the static SPARQL snippet for retrieving information by taxon scientific name.

Author(s)

<Author 1>

Examples

```
# Example for generating a SPARQL query by taxon scientific name
sparql_query <- query.by.name()
cat(sparql_query)
```

`query.by.taxid` *Generate a static SPARQL snippet for retrieving information by Taxonomy ID.*

Description

The ‘query.by.taxid‘ function generates a static SPARQL snippet for retrieving information based on Taxonomy ID.

Usage

```
query.by.taxid()
```

Value

A character string representing the static SPARQL snippet for retrieving information by Taxonomy ID.

Author(s)

<Author 1>

Examples

```
# Example for generating a SPARQL query by Taxonomy ID
sparql_query <- query.by.taxid()
cat(sparql_query)
```

`send.sparql` *Send a SPARQL query to an endpoint and retrieve results.*

Description

The ‘send.sparql‘ function sends a SPARQL query to a specified endpoint and retrieves the results. The function also supports adding prefixes to the query using the ‘add_prefixes‘ function.

Usage

```
send.sparql(query, endpoint, verbose = FALSE)
```

Arguments

`query`	A character string representing the SPARQL query to be sent.
`endpoint`	A character string representing the SPARQL endpoint to send the query to.
`verbose`	Logical, indicating whether to print the SPARQL query before sending.

Details

The function sends a GET request to the specified SPARQL endpoint with the provided query. It expects the response in the SPARQL Results JSON format and parses the results into a tibble. The function also checks the response type and raises an error if it's not in the expected format.

Value

A tibble containing the results of the SPARQL query.

Author(s)

<Author 1>

Examples

```
# Example for sending a SPARQL query to an endpoint
query <- "SELECT ?subject ?predicate ?object WHERE { ?subject ?predicate ?object } LIMIT 10"
endpoint <- "http://example.com/sparql"
result <- send.sparql(query, endpoint, verbose = TRUE)
print(result)
```

`source.tracking` *Generate a static SPARQL snippet for tracking the source of information about an organism*

Description

This function returns a SPARQL query string that retrieves the source of information about the focal organism, and includes a binding to check if the information has been inferred.

Usage

```
source.tracking()
```

Value

A character string representing the static SPARQL snippet for tracking the source of information about an organism.

Author(s)

<Author 1>

Examples

```
query <- source.tracking()
cat(query)
```

`to.short.iri` *Shorten a full IRI using specified namespaces.*

Description

The ‘to.short.iri‘ function shortens a full Internationalized Resource Identifier (IRI) using specified namespaces. It attempts to find the longest matching namespace and replaces it with its associated prefix.

Usage

```
to.short.iri(full.iri, namespaces = prefixes)
```

Arguments

`full.iri`	A character string representing the full IRI to be shortened.
`namespaces`	A named list containing namespace prefixes and their corresponding URIs.

Details

The function iterates through the provided namespaces to find the longest matching namespace for the given full IRI. If a matching namespace is found, it replaces the matching part with the associated prefix and returns the shortened IRI. If no matching namespace is found, the original full IRI is returned unchanged.

Value

A character string representing the shortened IRI with the associated prefix.

Author(s)

<Author 1>

Examples

```
# Example for shortening a full IRI using namespaces
full_iri <- "http://example.org/resource"
namespaces <- list(ex = "http://example.org/", foaf = "http://xmlns.com/foaf/0.1/")
shortened_iri <- to.short.iri(full_iri, namespaces)
cat(shortened_iri)
```

`to.short.iris` *Shorten multiple full IRIs using specified namespaces.*

Description

The ‘to.short.iris‘ function shortens a vector of full Internationalized Resource Identifiers (IRIs) using the ‘to.short.iri‘ function. It applies the ‘to.short.iri‘ function to each element of the vector.

Usage

```
to.short.iris(iris)
```

Arguments

`iris`	A vector of character strings representing the full IRIs to be shortened.

Details

The function applies the ‘to.short.iri‘ function to each element of the input vector of full IRIs. It returns a vector of shortened IRIs with associated prefixes based on the specified namespaces.

Value

A vector of character strings representing the shortened IRIs.

Author(s)

<Author 1>

Examples

```
# Example for shortening multiple full IRIs using namespaces
full_iris <- c("http://example.org/resource1", "http://example.org/resource2")
shortened_iris <- to.short.iris(full_iris)
cat(shortened_iris)
```

Index