Package ‘elasticsearchr’

July 30, 2019

Type Package

Version 0.3.1

Title A Lightweight Interface for Interacting with Elasticsearch from R

Date 2019-07-28

Author Alex Ioannides

Maintainer Alex Ioannides `<alex.ioannides@yahoo.co.uk>`

Description A lightweight R interface to 'Elasticsearch' - a NoSQL search-engine and column store database (see <https://www.elastic.co/products/elasticsearch> for more information). This package implements a simple Domain-Specific Language (DSL) for indexing, deleting, querying, sorting and aggregating data using 'Elasticsearch'.

License Apache License 2.0

URL `https://github.com/alexioannides/elasticsearchr`

BugReports `https://github.com/alexioannides/elasticsearchr/issues`

LazyData TRUE

Imports httr, jsonlite, dplyr

Suggests knitr, testthat, rmarkdown

VignetteBuilder knitr

RoxygenNote 6.1.1

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2019-07-30 21:00:02 UTC

R topics documented:

`+.elastic_api` *Define Elasticsearch aggregation on a secific subset of documents.*

Description

Sometimes it is necessary to perform an aggregation on the results of a query (i.e. on a subset of all the available documents). This is achieved by adding an `aggs` object to a `query` object.

Usage

```
## S3 method for class 'elastic_api'
x + y
```

Arguments

x	`elastic_query` object.
y	`elastic_aggs` or `elastic_sort` object.

Value

`elastic_aggs` object that contains the query information required for the aggregation.

Examples

```
all_docs <- query('{"match_all": {}}')
avg_sepal_width_per_cat <- aggs('{"avg_sepal_width_per_cat": {
      "terms": {"field": "species"},
      "aggs": {"avg_sepal_width": {"avg": {"field": "sepal_width"}}}}
}')
all_docs + avg_sepal_width_per_cat

sort_by_sepal_width <- sort_on('[{"sepal_width": {"order": "asc"}}]')
all_docs + sort_by_sepal_width
```

`aggs` *Define Elasticsearch aggregation.*

Description

Define Elasticsearch aggregation.

Usage

```
aggs(json)
```

Arguments

`json`	JSON object describing the aggregation that needs to be executed.

Value

An `elastic_aggs` object.

See Also

`https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations`
`html`

Examples

```
avg_sepal_width_per_cat <- aggs('{"avg_sepal_width_per_cat": {
      "terms": {"field": "species"},
      "aggs": {"avg_sepal_width": {"avg": {"field": "sepal_width"}}}}
}')
```

check_http_code_throw_error

HTTP response error handling.

Description

If an HTTP request returns a status code that is not 200, then this functions throws an exception and prints the prettified response contents to stderr.

Usage

```
check_http_code_throw_error(response)
```

Arguments

`response`	An HTTP response from a request to a search API request.

Value

Exception with prettified JSON response printed to stderr.

cleaned_field_names

Sanitise column names.

Description

Convert data frame column names into an Elasticsearch compatible format.

Usage

```
cleaned_field_names(colnames)
```

Arguments

`colnames`	A character vector containing data frame column names.

Details

Elasticsearch will not ingest field names with periods ("."), such as "Sepal.Width", as these are reserved for nested objects (in the JSON sense). This function replaces all period with underscores ("_") and the converts everything to lowercase for simplicity.

Value

A character vector with 'clean' column names.

Examples

```
## Not run:
df <- iris
colnames(df) <- cleaned_field_names(colnames(df))
colnames(df)
# "sepal_length" "sepal_width"  "petal_length" "petal_width"  "species"

## End(Not run)
```

---

create_bulk_upload_file

Create Bulk API data file.

Description

The fastest way to index, delete or update many documents, is via the Bulk API. This function assembles a text file comprising of data and/or actions in the format required by the Bulk API. This is ready to be POSTed to the Bulk API.

Usage

```
create_bulk_upload_file(metadata, df = NULL)

create_bulk_delete_file(metadata)
```

Arguments

`metadata`	A character vector of Bulk API document information objects, as generated by `create_metadata(...)`.
`df`	[optional] A data.frame with data for indexing or updating.

Value

The name of the temporary file containing the data for the Elasticsearch Bulk API.

See Also

`https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html` for more information on the information required by the Elasticsearch Bulk API.

Examples

```
## Not run:
bulk_upload_info <- create_metadata("index", "iris", "data", n = nrow(iris))
create_bulk_upload_file(bulk_upload_info, iris)
# "/var/folders/__/yz_l30s48xj6m_0059b_2twr0000gn/T//RtmpQnvUOt/file98194322b8"

bulk_delete_info <- create_metadata("delete", "iris", "data", n = nrow(iris))
```

```
create_bulk_delete_file(bulk_delete_info)
# "/var/folders/__/yz_l30s48xj6m_0059b_2twr0000gn/T//RtmpQnvUOt/file98194322b8"

## End(Not run)
```

`create_metadata` *Create Bulk API metadata.*

Description

The fastest way to index, delete or update many documents, is via the Bulk API. This requires that each document have the action combined with the document's metadata (index, type and id) sent to the API. This information is encapulated as a JSON object, that this function is responsible for generating.

Usage

```
create_metadata(action, index, doc_type, id = NULL, n = NULL)
```

Arguments

`action`	One of: "index", "create", "update" or "delete".
`index`	The name of the index where the documents reside (or will reside).
`doc_type`	The name of the document type where the documents reside (or will reside).
`id`	[optional] Character vector of document ids.
`n`	[optional] Integer number of repeated metadata description objects that need to be returned (if `id` is not specified).

Value

A character vector of Bulk API document information objects.

See Also

`https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html` for more information on the information required by the Elasticsearch Bulk API.

Examples

```
## Not run:
create_metadata("index", "iris", "data", n = 2)
'{\"index\": {\"_index\": \"iris\", \"_type\": \"data\"}}'
'{\"index\": {\"_index\": \"iris\", \"_type\": \"data\"}}'

## End(Not run)
```

`elastic` *elastic_resource class constructor.*

Description

Objects of this class contain all of the information required to locate documents in an Elasticsearch cluster.

Usage

```
elastic(cluster_url, index, doc_type = NULL)
```

Arguments

`cluster_url`	URL to the Elastic cluster.
`index`	The name of an index on the Elasticsearch cluster.
`doc_type`	[optional] The name of a document type within the index.

Value

An `elastic_rescource` object.

Examples

```
## Not run:
my_data <- elastic("http://localhost:9200", "iris", "data")

## End(Not run)
```

`elasticsearchr` *elasticsearchr: a lightweight Elasticsearch client for R.*

Description

Allows you to index, update and delete documents as well as run queries and aggregations.

`elastic_predicates` *elasticsearchr predicate functions.*

Description

Predicate functions for identifying different elasticsearchr object types.

Usage

```
is_elastic(x)

is_elastic_rescource(x)

is_elastic_api(x)

is_elastic_query(x)

is_elastic_aggs(x)

is_elastic_sort(x)

is_elastic_source_filter(x)

is_elastic_info(x)
```

Arguments

`x`	An elasticsearchr object.

Value

Boolean.

`elastic_version` *Elasticsearch version*

Description

Returns the major, minor and build version numbers for an Elasticsearch cluster, given a valid URL to an Elasticsearch cluster.

Usage

```
elastic_version(url)
```

Arguments

`url`	A valid URL to an Elasticsearch cluster.

Value

A list with the `major`, `minor` and `build` numbers.

Examples

```
## Not run:
elastic_version("http://localhost:9200")
$major
[1] 5

$minor
[1] 0

$build
[1] 1

## End(Not run)
```

`extract_query_results` *Elasticsearch HTTP response data extraction functions.*

Description

Functions for extracting the different types of data that can be contained in a response to a search API request.

Usage

```
extract_query_results(response)

extract_aggs_results(response)

extract_id_results(response)
```

Arguments

`response`	An HTTP response from a response to a search API request.

Value

A data.frame of response results.

from_size_search *Execute query with from-size search API.*

Description

The from-size search API allows a maximum of 10,000 search results (the maximum 'size') to be returned in one call to the API. The 'from' in the name of the API refers to where in the order of all qualifying documents (as ordered by their search score), should results start to be returned from. Anything larger than 10,000 and the results need to be fetched from using the scroll-search API (which is slower as it involves making multiple call-back requests). This API is particularly well suited to returning aggregation results.

Usage

```
from_size_search(rescource, api_call_payload)
```

Arguments

`rescource`	An `elastic` rescource object describing on what documents the query is to be execured on.
`api_call_payload`	A character string containing the JSON payload that described the query to be executed.

Value

A data.frame of documents returned from the query.

See Also

`https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-fro`
`html` for more information on the information required by the Elasticsearch from-size API.

Examples

```
## Not run:
elastic_rescource <- elastic("http://localhost:9200", "iris", "data")
query_json <- '{"query": {"match_all": {}}}'
results <- from_size_search(elastic_rescource, query_json)
head(results)
#   sepal_length sepal_width petal_length petal_width species
# 1          4.8         3.0          1.4         0.1  setosa
# 2          4.3         3.0          1.1         0.1  setosa
# 3          5.8         4.0          1.2         0.2  setosa
# 4          5.1         3.5          1.4         0.3  setosa
# 5          5.2         3.5          1.5         0.2  setosa
# 6          5.2         3.4          1.4         0.2  setosa

## End(Not run)
```

`index_bulk_dataframe` *Index data frame with Elasticsearch Bulk API*

Description

Helper function to orchestrate the assembly of the Bulk API upload file, http request to Elasticsearch and handling any subsequent respose errors. It's primary purpose is to be called repeatedly on 'chunks' of a data frame that is too bid to be indexed with a single call to the Bulk API (and hence the split into smaller more manageable chunks).

Usage

```
index_bulk_dataframe(rescource, df)
```

Arguments

`rescource`	An `elastic_rescource` object that contains the information on the Elasticsearch cluster, index and document type, where the indexed data will reside. If this does not already exist, it will be created automatically.
`df`	data.frame whose rows will be indexed as documents in the Elasticsearch cluster.

Examples

```
## Not run:
rescource <- elastic("http://localhost:9200", "iris", "data")
index_bulk_dataframe(rescource, iris)

## End(Not run)
```

`list_fields` *List of fields in index information.*

Description

List of fields in index information.

Usage

```
list_fields()
```

Value

An `elastic_info` object.

Examples

```
list_fields()
```

`list_indices` *List of indices in cluster information.*

Description

List of indices in cluster information.

Usage

```
list_indices()
```

Value

An `elastic_info` object.

Examples

```
list_indices()
```

`mapping_default_simple` *Simple Elasticsearch default mappings for non-text-search analytics*

Description

This mapping switches-off the text analyser for all fields of type 'string' (i.e. switches off free text search), allows all text search to work with case-insensitive lowercase terms, and maps any field with the name 'timestamp' to type 'date', so long as it has the appropriate string or long format.

Usage

```
mapping_default_simple()
```

`mapping_fielddata_true` *Elasticsearch 5.x default mappings enabling fielddata for text fields*

Description

A default mapping that enables fielddata for all string/text fields in Elasticsearch 5.x.

Usage

```
mapping_fielddata_true()
```

`print.elastic_api` *Pretty-print aggs and query JSON objects.*

Description

Pretty-print aggs and query JSON objects.

Usage

```
## S3 method for class 'elastic_api'
print(x, ...)
```

Arguments

`x`	`elastic_query` or `elastic_aggs` object.
`...`	For consitency with all other `print` methods.

Value

Character string of pretty-printed JSON object.

Examples

```
all_docs <- query('{"match_all": {}}')
print(all_docs)
```

`query` *Define Elasticsearch query.*

Description

Define Elasticsearch query.

Usage

```
query(json, size = 0)
```

Arguments

`json`	JSON object describing the query that needs to be executed.
`size`	[optional] The number of documents to return. If left unspecified, then the default if to return all documents.

Value

An `elastic_query` object.

See Also

`https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html`

Examples

```
all_docs <- query('{"match_all": {}}')
```

`scroll_search` *Execute a query with the scroll-search API.*

Description

The scroll-search API works by returning a 'token' to the user that allows search results to be returned one 'page' at a time. This, large query results (in excess of the 10,000 documents maximum size offered by the from-search API) can be retreived by making multiple calls after the initial query was sent. Although a slower process end-to-end, this API is particularly well suited to returning large query results.

Usage

```
scroll_search(rescource, api_call_payload,
  extract_function = extract_query_results)
```

Arguments

`rescource`	An `elastic` rescource object describing on what documents the query is to be execured on.
`api_call_payload`	A character string containing the JSON payload that described the query to be executed.
`extract_function`	A function to be used for extracting the data from the responses sent back from the scroll-search API. Defaults to `extract_query_results` that extracts query results, for when the scroll-search API is being used for retreiving query results (as opposed to aggregations or document ids, etc.).

Value

A data.frame of documents returned from the query.

See Also

`https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-scr`
`html` for more information on the information required by the Elasticsearch scroll-search API.

Examples

```
## Not run:
elastic_rescource <- elastic("http://localhost:9200", "iris", "data")
query_json <- '{"query": {"match_all": {}}}'
results <- scroll_search(elastic_rescource, query_json)
head(results)
#   sepal_length sepal_width petal_length petal_width species
# 1          4.8         3.0          1.4         0.1  setosa
# 2          4.3         3.0          1.1         0.1  setosa
# 3          5.8         4.0          1.2         0.2  setosa
# 4          5.1         3.5          1.4         0.3  setosa
# 5          5.2         3.5          1.5         0.2  setosa
# 6          5.2         3.4          1.4         0.2  setosa

## End(Not run)
```

`select_fields` *Define Elasticsearch query source filter.*

Description

Define Elasticsearch query source filter.

Usage

```
select_fields(json)
```

Arguments

`json`	JSON object describing the aggregation that needs to be executed.

Value

An `elastic_source_filter` object.

See Also

`https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-sou`
`html`

sort_on *Define Elasticsearch query sort*

Description

Define Elasticsearch query sort

Usage

```
sort_on(json)
```

Arguments

`json`	JSON object describing the sorting required on the query results.

Value

An `elastic_sort` object.

See Also

`https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-sort`
`html`

Examples

```
sort_by_key <- sort_on('[{"sort_key": {"order": "asc"}}]')
```

valid_connection *Validate healthy Elasticsearch connection.*

Description

Validates healthy Elasticsearch connections by attempting to call the cluster healthcheck endpoint. In doing to, it defends against incorrect URLs to Elasticsearch clusters. Requires that URLs point directly to a master node - i.e. the endpoint that would return the default Elasticsearch message, "You Know, for Search", e.g. ‘http://localhost:9200‘.

Usage

```
valid_connection(url)
```

Arguments

`url`	The URL to validate.

Value

Boolean

Examples

```
## Not run:
url <- "http://localhost:9200"
valid_connection(url)
# TRUE

url <- "http://localhost:9201"
valid_connection(url)
#  Error: Failed to connect to localhost port 9201: Connection refused

## End(Not run)
```

`valid_json` *Valid JSON string predicate function*

Description

Valid JSON string predicate function

Usage

```
valid_json(json)
```

Arguments

`json`	Candidate JSON object as a string.

Value

Boolean.

Examples

```
## Not run:
good_json <- '{"id": 1}'
valid_json(good_json)
# TRUE

bad_json <- '{"id": 1a}'
valid_json(bad_json)
# FALSE

## End(Not run)
```

`%create%` *Create Elasticsearch index with custom mapping.*

Description

Mappings are the closest concept to traditional database 'schema'. This function allows the creation of Elasticsearch indicies with custom mappings. If left unspecified, Elasticsearch will infer the type of each field based on the first document indexed.

Usage

```
rescource %create% mapping
```

Arguments

`rescource`	An `elastic_rescource` object that contains the information on the Elasticsearch cluster, index and document type, where the indexed data will reside. If this does not already exist, it will be created automatically.
`mapping`	A JSON object containing the mapping details required for the index.

See Also

`https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html`

Examples

```
## Not run:
elastic("http://localhost:9200", "iris", "data") %create% mapping_default_simple()

## End(Not run)
```

`%delete%` *Delete Elasticsearch index.*

Description

Delete all of the documents within a particular document type (if specified), or delete an entire index (if the document type is unspecified.)

Usage

```
rescource %delete% approve
```

Arguments

`rescource`	An `elastic_rescource` object that contains the information on the Elasticsearch cluster, index and document type, where the indexed data will reside. If this does not already exist, it will be created automatically.
`approve`	Must be equal to `"TRUE"` for deletion for all documents in a rescource, OR be a character vector of document ids if only specific documents need to be deleted.

Examples

```
## Not run:
elastic("http://localhost:9200", "iris", "data") %delete% TRUE

## End(Not run)
```

`%index%` *Index a data frame.*

Description

Inserting records (or documents) into Elasticsearch is referred to as "indexing' the data. This function considers each row of a data frame as a document to be indexed into an Elasticsearch index.

Usage

```
rescource %index% df
```

Arguments

`rescource`	An `elastic_rescource` object that contains the information on the Elasticsearch cluster, index and document type, where the indexed data will reside. If this does not already exist, it will be created automatically.
`df`	data.frame whose rows will be indexed as documents in the Elasticsearch cluster.

Details

If the data frame contains a column named 'id', then this will be used to assign document ids. Otherwise, Elasticsearch will automatically assigne the documents random ids.

See Also

`https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-index_.html`

Examples

```
## Not run:
elastic("http://localhost:9200", "iris", "data") %index% iris

## End(Not run)
```

`%info%` *Get cluster and index (meta) data.*

Description

An operator to be used with requests for information

Usage

```
rescource %info% info
```

Arguments

`rescource`	An `elastic_rescource` object that contains the information on the Elasticsearch cluster, index and document type, where the indexed data will reside. If this does not already exist, it will be created automatically.
`info`	`elastic_info` object.

Examples

```
## Not run:
elastic("http://localhost:9200", "iris", "data") %info% list_indices()
elastic("http://localhost:9200", "iris", "data") %info% list_fields()

## End(Not run)
```

`%search%` *Execute query or search.*

Description

Execute query or search.

Usage

```
rescource %search% search
```

Arguments

`rescource`	An `elastic_rescource` object that contains the information on the Elasticsearch cluster, index and document type, where the indexed data will reside. If this does not already exist, it will be created automatically.
`search`	`elastic_query` or `elastic_aggs` object.

Value

A data.frame of search or aggregation results.

Examples

```
## Not run:
results <- elastic("http://localhost:9200", "iris", "data") %search% query('{"match_all": {}
head(results)
#   sepal_length sepal_width petal_length petal_width species
# 1          4.8         3.0          1.4         0.1  setosa
# 2          4.3         3.0          1.1         0.1  setosa
# 3          5.8         4.0          1.2         0.2  setosa
# 4          5.1         3.5          1.4         0.3  setosa
# 5          5.2         3.5          1.5         0.2  setosa
# 6          5.2         3.4          1.4         0.2  setosa

## End(Not run)
```