Package 'mongolite'

March 21, 2016

Type Package

Title Fast and Simple MongoDB Client for R

Description High-level, high-performance MongoDB client based on libmongoc and jsonlite. Includes support for aggregation, indexing, map-reduce, streaming, SSL encryption and SASL authentication. The vignette gives a brief overview of the available methods in the package.

Version 0.8.1

Maintainer Jeroen Ooms <jeroen.ooms@stat.ucla.edu>

Imports jsonlite (>= 0.9.16)

License Apache License 2.0

URL https://github.com/jeroenooms/mongolite
https://github.com/mongodb/mongo-c-driver

BugReports https://github.com/jeroenooms/mongolite/issues

SystemRequirements OpenSSL, Cyrus SASL (aka libsasl2)

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation yes

Author Jeroen Ooms [aut, cre],
MongoDB, Inc [cph] (Author of mongo-c-driver)

Repository CRAN

Date/Publication 2016-03-21 23:08:24

R topics documented:

mongo *MongoDB client*

Description

Connect to a MongoDB collection. Returns a mongo connection object with methods listed below.

Usage

```
mongo(collection = "test", db = "test", url = "mongodb://localhost",
  verbose = TRUE)
```

Arguments

collection	name of collection
db	name of database
url	address of the mongodb server in mongo connection string URI format.
verbose	emit some more output

Value

Upon success returns a pointer to a collection on the server. The collection can be interfaced using the methods described below.

Methods

`aggregate(pipeline = '{}', handler = NULL, pagesize = 1000)` Execute a pipeline using the Mongo aggregation framework.

`count(query = '{}')` Count the number of records matching a given `query`. Default counts all records in collection.

`distinct(key, query = '{}')` List unique values of a field given a particular query.

`drop()` Delete entire collection with all data and metadata.

`export(con = stdout(), bson = FALSE)` Streams all data from collection to a `connection` in jsonlines format (similar to mongoexport). Alternatively when `bson = TRUE` it outputs the binary bson format (similar to mongodump).

`find(query = '{}', fields = '{"_id" : 0}', sort = '{}', skip = 0, limit = 0, handler = NULL, pagesi` Retrieve `fields` from records matching `query`. Default `handler` will return all data as a single dataframe.

`import(con, bson = FALSE)` Stream import data in jsonlines format from a `connection`, similar to the mongoimport utility. Alternatively when `bson = TRUE` it assumes the binary bson format (similar to mongorestore).

`index(add = NULL, remove = NULL)` List, add, or remove indexes from the collection. The `add` and `remove` arguments can either be a field name or json object. Returns a dataframe with current indexes.

`info()` Returns collection statistics and server info (if available).

`insert(data, pagesize = 1000)` Insert a dataframe into the collection.

`iterate(query = '{}', fields = '{"_id":0}', sort = '{}', skip = 0, limit = 0)` Runs query and returns iterator to read single records one-by-one.

`mapreduce(map, reduce, query = '{}', sort = '{}', limit = 0, out = NULL, scope = NULL)` Performs a map reduce query. The `map` and `reduce` arguments are strings containing a JavaScript function. Set `out` to a string to store results in a collection instead of returning.

`remove(query = "{}", multiple = FALSE)` Remove record(s) matching `query` from the collection.

`rename(name, db = NULL)` Change the name or database of a collection. Changing name is cheap, changing database is expensive.

`update(query, update = '{"$set":{}}', upsert = FALSE, multiple = FALSE)` Replace or modify matching record(s) with value of the `update` argument.

References

Jeroen Ooms (2014). The `jsonlite` Package: A Practical and Consistent Mapping Between JSON Data and R Objects. *arXiv:1403.2805*. http://arxiv.org/abs/1403.2805

Examples

```
# Connect to mongolabs
con <- mongo("mtcars", url = "mongodb://readwrite:test@ds043942.mongolab.com:43942/jeroen_test")
if(con$count() > 0) con$drop()
con$insert(mtcars)
stopifnot(con$count() == nrow(mtcars))

# Query data
mydata <- con$find()
stopifnot(all.equal(mydata, mtcars))
con$drop()

# Automatically disconnect when connection is removed
rm(con)
gc()

## Not run:
# dplyr example
library(nycflights13)

# Insert some data
m <- mongo(collection = "nycflights")
m$drop()
m$insert(flights)

# Basic queries
m$count('{"month":1, "day":1}')
jan1 <- m$find('{"month":1, "day":1}')

# Sorting
```

```
jan1 <- m$find('{"month":1,"day":1}', sort='{"distance":-1}')
head(jan1)

# Sorting on large data requires index
m$index(add = "distance")
allflights <- m$find(sort='{"distance":-1}')

# Select columns
jan1 <- m$find('{"month":1,"day":1}', fields = '{"_id":0, "distance":1, "carrier":1}')

# List unique values
m$distinct("carrier")
m$distinct("carrier", '{"distance":{"$gt":3000}}')

# Tabulate
m$aggregate('[{"$group":{"_id":"$carrier", "count": {"$sum":1}, "average":{"$avg":"$distance"}}}]')

# Map-reduce (binning)
hist <- m$mapreduce(
  map = "function(){emit(Math.floor(this.distance/100)*100, 1)}",
  reduce = "function(id, counts){return Array.sum(counts)}"
)

# Stream jsonlines into a connection
tmp <- tempfile()
m$export(file(tmp))

# Remove the collection
m$drop()

# Import from jsonlines stream from connection
dmd <- mongo("diamonds")
dmd$import(url("http://jeroenooms.github.io/data/diamonds.json"))
dmd$count()

# Export
dmd$drop()

## End(Not run)
```

Index