

# Package ‘mongolite’

April 11, 2017

**Type** Package

**Title** Fast and Simple 'MongoDB' Client for R

**Description** High-performance 'MongoDB' client based on 'libmongoc' and 'jsonlite'. Includes support for aggregation, indexing, map-reduce, streaming, encryption, enterprise authentication. The online user manual provides an overview of the available methods in the package: <<https://jeroen.github.io/mongolite/>>.

**Version** 1.2

**Imports** jsonlite (>= 1.4), openssl (>= 0.9.4)

**License** Apache License 2.0

**URL** <https://github.com/jeroen/mongolite/> (devel)  
<https://jeroen.github.io/mongolite/> (user manual)  
<http://mongoc.org/> (upstream)

**BugReports** <https://github.com/jeroen/mongolite/issues>

**SystemRequirements** OpenSSL, Cyrus SASL (aka libsasl2)

**RoxygenNote** 6.0.1

**NeedsCompilation** yes

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

**Maintainer** Jeroen Ooms <jeroen@berkeley.edu>

**Repository** CRAN

**Date/Publication** 2017-04-11 10:35:33 UTC

## R topics documented:

mongo . . . . .	2
mongo_options . . . . .	4
ssl_options . . . . .	5
<b>Index</b>	<b>7</b>

---

 mongo

*MongoDB client*


---

## Description

Connect to a MongoDB collection. Returns a [mongo](#) connection object with methods listed below. The [mongolite user manual](#) is the best place to get started.

## Usage

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

## Arguments

collection	name of collection
db	name of database
url	address of the mongodb server in mongo connection string <a href="#">URI format</a>
verbose	emit some more output
options	additional connection options such as SSL keys/certs.

## 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, pagesize = 1000)` 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 rows into the collection. Argument 'data' must be a data-frame, named list (for single record) or character vector with json strings (one string for each row). For lists and data frames, arguments in ... get passed to `jsonlite::toJSON`

`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

### Mongolite User Manual

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$import(url("http://jeroen.github.io/data/diamonds.json"))
dmd$count()

# Export
dmd$drop()

## End(Not run)

```

**Description**

Get and set global client options. Calling with NULL parameters returns current values without modifying.

**Usage**

```
mongo_options(log_level = NULL, bigint_as_char = NULL,  
              date_as_char = NULL)
```

**Arguments**

log\_level integer between 0 and 6 or NULL to leave unchanged.  
bigint\_as\_char logical: parse int64 as strings instead of double.  
date\_as\_char logical: parse UTC datetime as strings instead of POSIXct.

**Details**

Setting **log\_level** to 0 suppresses critical warnings and messages, while 6 is most verbose and displays all debugging information. Possible values for level are:

- 0: *error*
- 1: *critical*
- 2: *warning*
- 3: *message*
- 4: *info (default)*
- 5: *debug*
- 6: *trace*

Note that setting it below 2 will suppress important warnings and setting below 1 will suppress critical errors (not recommended). The default is 4.

---

ssl\_options

*Connection SSL options*

---

**Description**

Set SSL options to connect to the MongoDB server.

**Usage**

```
ssl_options(cert = NULL, key = cert, ca = NULL, ca_dir = NULL,  
            crl_file = NULL, allow_invalid_hostname = NULL,  
            weak_cert_validation = NULL)
```

**Arguments**

cert	path to PEM file with client certificate, or a certificate as returned by <code>openssl::read_cert()</code>
key	path to PEM file with private key from the above certificate, or a key as returned by <code>openssl::read_key()</code> . This can be the same PEM file as cert.
ca	a certificate authority PEM file
ca_dir	directory with CA files
crl_file	file with revocations
allow_invalid_hostname	do not verify hostname on server certificate
weak_cert_validation	disable certificate verification

# Index

connection, [2](#)

jsonlite::toJSON, [3](#)

mongo, [2](#), [2](#)

mongo\_options, [4](#)

mongolite (mongo), [2](#)

openssl::read\_cert(), [6](#)

openssl::read\_key(), [6](#)

ssl\_options, [5](#)