Package 'GeoMongo'

September 9, 2017

Type Package

Title Geospatial Queries Using 'PyMongo'

Version 1.0.1

Date 2017-09-09

Author Lampros Mouselimis <mouselimislampros@gmail.com>

Maintainer Lampros Mouselimis <mouselimislampros@gmail.com>

BugReports https://github.com/mlampros/GeoMongo/issues

URL https://github.com/mlampros/GeoMongo

Description Utilizes methods of the 'PyMongo' 'Python' library to initialize, insert and query 'GeoJson' data (see <https://api.mongodb.com/python/current/#> for more information on 'PyMongo'). Furthermore, it allows the user to validate 'GeoJson' objects and to use the console for 'MongoDB' (bulk) commands. The 'reticulate' package provides the 'R' interface to 'Python' modules, classes and functions.

License Apache License 2.0

SystemRequirements MongoDB (>= 3.4.0), Python (>= 2.7). Installation instructions and links can be found in the README file.

Depends R(>= 3.2.3)

Imports reticulate, R6, geojsonR, data.table

Suggests testthat, covr, knitr, rmarkdown

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2017-09-09 15:03:13 UTC

R topics documented:

`geomongo` *mongodb geospatial methods (using PyMongo in R)*

Description

mongodb geospatial methods (using PyMongo in R)

Usage

```
# init <- geomongo$new(host = 'localhost', port = 27017,

       #                      tz_aware = FALSE, connect = TRUE, ...)
```

Arguments

`host`	(optional) hostname or IP address or Unix domain socket path of a single mongod or mongos instance to connect to, or a mongodb URI, or a list of hostnames / mongodb URIs. See the reference link for more information.
`port`	(optional) port number on which to connect
`document_class`	(optional) default class to use for documents returned from queries on this client
`tz_aware`	(optional) if TRUE, datetime instances returned as values in a document by this MongoClient will be timezone aware (otherwise they will be naive)
`connect`	(optional) if TRUE (the default), immediately begin connecting to MongoDB in the background. Otherwise connect on the first operation
`...`	See the reference link for more details on the *ellipsis* (...) concerning the additional parameters of the MongoClient()
`FILE`	a character string specifying a valid path to a file (applies to *read_mongo_bson* method)
`STR`	a character string (applies to *read_mongo_bson* method)
`DATA`	a valid path to a file/folder or a list (applies to *geoInsert* method)
`TYPE_DATA`	a character string. One of 'folder', 'file', 'dict_one' (takes as input a *list* or a *character string*) or 'dict_many' (takes as input a *list* or a *character string vector*) (applies to *geoInsert* method)
`COLLECTION`	a *pymongo.collection.Collection* object (applies to *geoInsert* and *geoQuery* methods)
`GEOMETRY_NAME`	a character string specifying the name of the geometry object, as it appears in the file/string (applies to *geoInsert* and *geoQuery* methods)

read_method	a character string specifying the method to use to read the data. Either using the *"geojsonR"* (package) or the *"mongo_bson"* utility function (applies to *geoInsert* method)
QUERY	a named list specifying the query to use in mongodb (applies to *geoQuery* method)
METHOD	a character string specifying the method to use to perform geospatial queries in mongodb. One of "find", "aggregate" OR "command" (applies to *geoQuery* method)
DATABASE	a *"pymongo.database.Database"* object (applies to *geoQuery* method)
TO_LIST	either TRUE or FALSE. If TRUE then the output of the *geoQuery* method will be a list, otherwise a data.table (matrix) object (applies to *geoQuery* method)

Format

An object of class `R6ClassGenerator` of length 24.

Details

the *geomongo$new* method initializes the MongoClient

the *getClient* method returns a *"pymongo.mongo_client.MongoClient"* object

the *read_mongo_bson* method allows the user to read a file/string using the *bson.json_util* module, which loads MongoDB Extended JSON data (SEE *https://stackoverflow.com/questions/42089045/bson-errors-invaliddocument-key-oid-must-not-start-with-trying-to-insert*)

the *geoInsert* method allows the user to import data to a mongo-db from a *folder*, *file* or *list*

the *geoQuery* method allows the user to perform geospatial queries using one of the *find*, *aggregate* or *command* methods

For spherical query operators to function properly, you must convert distances to radians, and convert from radians to the distances units used by your application.

To convert distance to radians: divide the distance by the radius of the sphere (e.g. the Earth) in the same units as the distance measurement. To convert radians to distance: multiply the radian measure by the radius of the sphere (e.g. the Earth) in the units system that you want to convert the distance to.

The equatorial radius of the Earth is approximately 3,963.2 miles or 6,378.1 kilometers.

If specifying latitude and longitude coordinates, list the longitude first and then latitude:

Valid longitude values are between -180 and 180, both inclusive. Valid latitude values are between -90 and 90 (both inclusive).

Methods

```
geomongo$new(host = 'localhost', port = 27017, tz_aware = FALSE, connect = TRUE, ...)

--------------

getClient()

--------------
```

```
read_mongo_bson(FILE = NULL, STR = NULL)
--------------
geoInsert(DATA = NULL, TYPE_DATA = NULL, COLLECTION = NULL, GEOMETRY_NAME = NULL, read_method = "ge

--------------
geoQuery(QUERY = NULL, METHOD = NULL, COLLECTION = NULL, DATABASE = NULL, GEOMETRY_NAME = NULL, TO_
```

References

https://api.mongodb.com/python/current/api/index.html, https://docs.mongodb.com/manual/tutorial/calculate-distances-using-spherical-geometry-with-2d-geospatial-indexes/

Examples

```
## Not run:
library(GeoMongo)

init = geomongo$new()

getter_client = init$getClient()

init_db = getter_client$get_database("example_db")

init_col = init_db$get_collection("example_collection")

#--------------------------
# geonear using 'aggregate'
#--------------------------

query_geonear = list('$geoNear' = list(near = list(type = "Point", coordinates = c(-122.5, 37.1)),

                     distanceField = "distance", maxDistance = 900 * 1609.34,

                     distanceMultiplier = 1 / 1609.34, spherical = TRUE))


init$geoQuery(QUERY = query_geonear, METHOD = "aggregate", COLLECTION = init_col,

              DATABASE = init_db, GEOMETRY_NAME = "location", TO_LIST = FALSE)

## End(Not run)
```

`json_schema_validator` *simple way to validate a json instance under a given schema*

Description

simple way to validate a json instance under a given schema

Usage

```
json_schema_validator(json_data = NULL, json_schema = NULL)
```

Arguments

json_data	a named list specifying the input data to validate against the json-schema
json_schema	a named list specifying the json-schema

Details

Define a json-schema that the input data should follow and then validate the input data against the schema. If the input data follows the schema then by running the function nothing will be returned, otherwise an error with Traceback will be printed in the R-session.

In case that *type* is at the same time also a property name in the json data, then do not include *"type" = "string"* in the json schema (https://github.com/epoberezkin/ajv/issues/137)

References

https://pypi.python.org/pypi/jsonschema, http://python-jsonschema.readthedocs.io/en/latest/

Examples

```
library(GeoMongo)

if (reticulate::py_available() && reticulate::py_module_available("jsonschema")) {

 schema_dict = list("type" = "object",

                     "properties" = list(

                       "name" = list("type" = "string"),

                          "location" = list("type" = "object",

                          "properties" = list(

                           "type" = list("enum" = c("Point", "Polygon")),

                           "coordinates" = list("type" = "array")
 ))))


 data_dict = list("name" = "example location",

                "location" = list("type" = "Point", "coordinates" = c(-120.24, 39.21)))
```

```
 json_schema_validator(json_data = data_dict, json_schema = schema_dict)

}
```

---

mongodb_console *MongoDB (bulk) commands*

Description

MongoDB (bulk) commands

Usage

```
mongodb_console(Argument = NULL, ...)
```

Arguments

Argument	a character string specifying the mongodb shell command to run from within an R-session
...	the *ellipsis* (...) parameter allows a unix-user (windows-user) to give additional parameters to the base-R *system() (shell())* function which is run in background.

Details

MongoDB shell commands are important for instance if someone has to import/export bulk data to a mongo database. This R function utilizes the *system* base function to run the mongodb shell command from within an R-session. See the reference links for more details. The *ellipsis* (...) parameter could be used for instance to disallow messages be printed in the console (on unix by using *ignore.stdout* and *ignore.stderr*).

References

https://docs.mongodb.com/manual/reference/program/mongoimport/, https://docs.mongodb.com/manual/reference/program/

Examples

```
## Not run:
library(GeoMongo)

ARGs = "mongoimport -d DB -c COLLECTION --type json --file /MY_DATA.json"

mongodb_console(Argument = ARGs)

## End(Not run)
```

Index