MongoDB: Web Scale!

MAJ Nicholas Uhorchak

02 April, 2020

Pre-tutorial requirements

This tutorial follows Labs 1 through 5 from the MongoDB teach session. Please review and complete lab 1 and 2 at a minimum, and enesure that you have an active and populated database to complete the following. Those tutorials are provided for your review in Teams.

Connecting to your cloud database

With a live, populated database in the cloud. . . what then?

Well, first we define the connection string that will allow us to connect to the database itself, and query. For that, we use the R package 'mongolite'

```
library(mongolite)

con <- "mongodb+srv://workshop:workshop@workshop-memwm.mongodb.net/test?retryWrites=true&w=majority"
```

Query the cloud database

Once the connection is established, 'con' above, we can connect to the DB and create the environment against which queries are run. This connection connects to the 'Workshop' database and the 'restaurant' collection stored within it.

For the connection to properly work, the IP address where you access the databse must be whitelisted in MongoDB Atlas cloud instance, (see labs 1-2).

```
restaurantDB <- mongolite::mongo(collection = "Restaurants", db = "Workshop", url = con)
```

'restaurantDB' above, is an environment, stored within the R studio environment. That environment contains a set of functions, accessed with the $ character, to perform necessary actions against the database connection.

Here we query, the restaurant collection from the database, executing the equivalent to 'SELECT * FROM *' from a relational database. The empty set '{}' ensures that we query the entire collection.

```
all_data <- restaurantDB$find("{}")
tibble::glimpse(all_data)
```

```
## Observations: 25,391
## Variables: 17
## $ address       <df[,4]> <data.frame[26 x 4]>
```

```
## $ borough       <chr> "Manhattan", "Manhattan", "Manhattan", "Staten Island...
## $ cuisine       <chr> "American", "Pizza", "American", "Delicatessen", "Del...
## $ grades        <list> [<data.frame[4 x 3]>, <data.frame[4 x 3]>, <data.fra...
## $ name          <chr> "P & S Deli Grocery", "Domino'S Pizza", "Spoon Bread ...
## $ restaurant_id <chr> "40362264", "40363945", "40364179", "40364286", "4036...
## $ mpg           <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ cyl           <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ disp          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ hp            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ drat          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ wt            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ qsec          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ vs            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ am            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ gear          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ carb          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
```

What happens however, when we do not want to query the entire set and want to send specific query parameters to mongoDB?

Specific queries

```
staten_island_chinese <- restaurantDB$find("{\"borough\": \"Staten Island\",
                  \"cuisine\":\"Chinese\"}")
tibble::glimpse(staten_island_chinese)
```

```
## Observations: 88
## Variables: 6
## $ address       <df[,4]> <data.frame[26 x 4]>
## $ borough       <chr> "Staten Island", "Staten Island", "Staten Island", "S...
## $ cuisine       <chr> "Chinese", "Chinese", "Chinese", "Chinese", "Chinese"...
## $ grades        <list> [<data.frame[3 x 3]>, <data.frame[5 x 3]>, <data.fra...
## $ name          <chr> "Chen'S Chinese", "Island Taste", "Master Wok", "Happ...
## $ restaurant_id <chr> "40372831", "40394383", "40394394", "40401094", "4053...
```

Given your understanding of the database, it is easy to pass a query string to the database, but you must ensure that double quotations encase both portion of the query argument, otherwise you will reeive an error. Single quotations are limited to the entire query being passed to the $find command.

##Example query analysis Given the newly queried data, we can now perform operations as needed. For this demonstration, we produce a quick mapping of these restaurants.

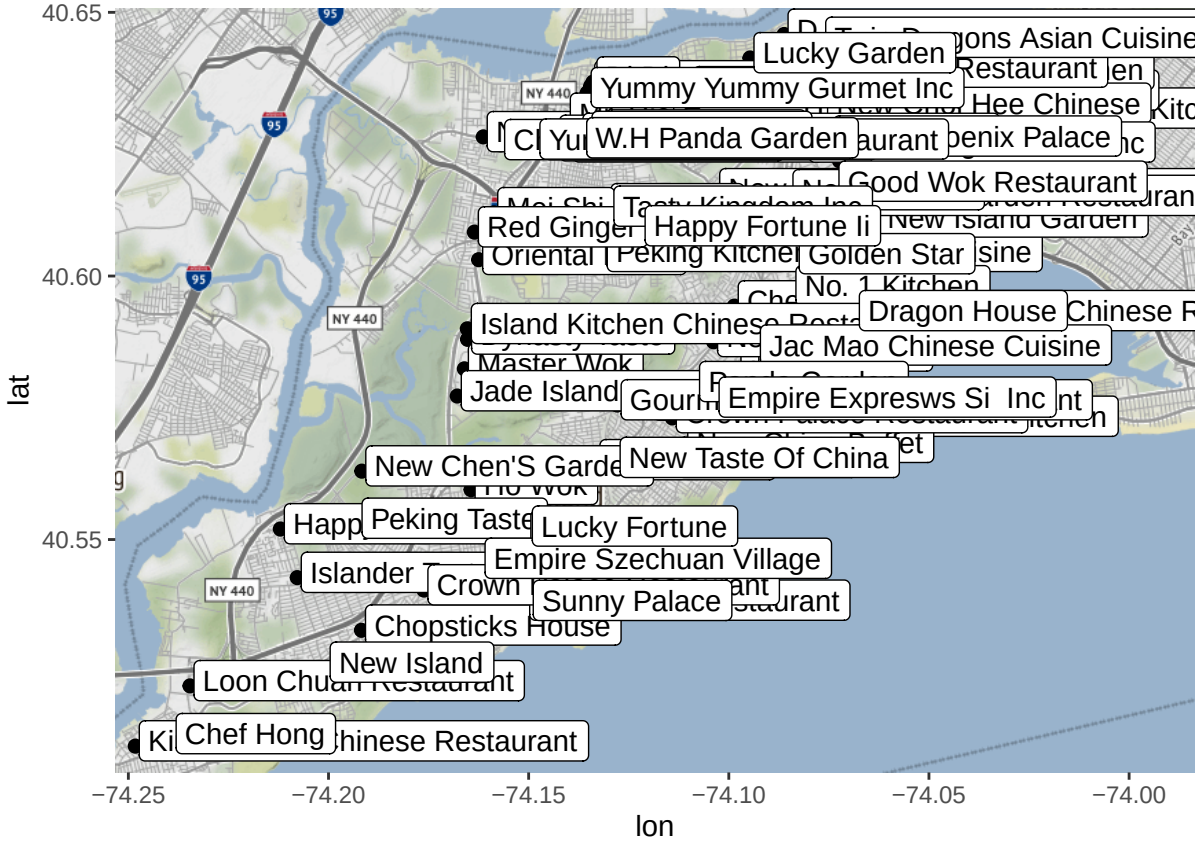
The initial data returned from the database is stored in a list of two values of two numeric inputs, latitude and longitude. It must be transformed prior to use.

```
lat_longs <- tibble::enframe(staten_island_chinese$address$coord, name = "name",
    value = "value") %>% tidyr::unnest() %>% dplyr::group_by(name) %>% dplyr::mutate(key = c("long",
    "lat")) %>% tidyr::pivot_wider(names_from = key, values_from = value)

staten_island_chinese <- staten_island_chinese %>% add_column(lat = lat_longs$lat)

staten_island_chinese <- staten_island_chinese %>% add_column(long = lat_longs$long)
```

Mapping is then an easy combination of the lat/long combination above, and any other information you want displayed.

Although it is possible to create a googe could account and connect to the google maps API and use ggmap (https://www.littlemissdata.com/blog/maps), this alternate method utilizes the open street maps API to get a map without a google account.

```
library(ggmap)

location = c((min(staten_island_chinese$long) - 0.005), (min(staten_island_chinese$lat) -
    0.005), (max(staten_island_chinese$long) + 0.08), (max(staten_island_chinese$lat) +
    0.005)) %>% as.numeric()
staten_island <- get_map(location = location, source = "osm")
ggmap(staten_island) + geom_point(data = staten_island_chinese, aes(x = long, y = lat),
    size = 2) + geom_label(data = staten_island_chinese, position = "nudge", aes(label = name,
    x = long + 0.001, y = lat + 0.001), hjust = 0)
```



Lessons Learned

- The mongoDB Atlas instance must be setup properly for this to work.
- Queries and quotations marks and excape quotes matter
- Because mongoDB is a JSON database, the return format DF/Tibble is not necessarily easy to work with in R
- This is a powerful tool if we ever get it on the network and use it *properly*

Enjoy.