National University of Singapore
School of Continuing & Lifelong Education (SCALE)

TBA2105 Web Mining
Tutorial/Lab 6

Learning Objectives

- Connect to an API using R
- Perform web scraping using RSelenium

OneMap API

In this question, we will explore how to use the OneMap Search API in R (https://docs.onemap.sg/#search). OneMap API allows you to search for address information of Singapore locations. You currently do not need to register an account in order to use this search API. However, if you want to use its Reverse Geocode API, you will need to register an account in order to get an access token.

Using OneMap Search API

For more information about the API, you can visit the OneMap documentation website for its search API (https://docs.onemap.sg/#search). Here's an example taken from the onemap documentation website

Example:
Suppose we are searching for location that contains the name "*revenue*", the URL should look something like:

```
https://developers.onemap.sg/commonapi/search?searchVal=revenue&returnGeom=Y&getAddrDetails=Y&pageNum=1
```

In here, we replace the `searchVal` parameter value with the search parameter we want to search for. There are a few more parameters which you can define (e.g. `returnGeom`, `getAddrDetails`, `pageNum`). The response of this request is as follows:


```
{
   "found":5,
   "totalNumPages":1,
   "pageNum":1,
   "results":[
      {
         "SEARCHVAL":"INLAND REVENUE AUTHORITY OF SINGAPORE (IRAS)",
         "BLK_NO":"55",
         "ROAD_NAME":"NEWTON ROAD",
         "BUILDING":"INLAND REVENUE AUTHORITY OF SINGAPORE (IRAS)",
         "ADDRESS":"55 NEWTON ROAD, SINGAPORE 307987",
         "POSTAL":"307987",
         "X":"28983.7537272647",
         "Y":"33554.4361084122",
         "LATITUDE":"1.31972890510723",
         "LONGITUDE":"103.842158118267",
         "LONGTITUDE":"103.842158118267"
      },
      {
```

```
          "SEARCHVAL":"REVENUE HOUSE",
          "BLK_NO":"55",
          "ROAD_NAME":"NEWTON ROAD",
          "BUILDING":"REVENUE HOUSE",
          "ADDRESS":"55 NEWTON ROAD, SINGAPORE 307987",
          "POSTAL":"307987",
          "X":"28977.8507137401",
          "Y":"33547.5712691676",
          "LATITUDE":"1.31966682211667",
          "LONGITUDE":"103.842105076401",
          "LONGTITUDE":"103.842105076401"
       }
   ]
}
```

To use this API in the R environment we will make use of the jsonlite package which will access an URL and convert the JSON into a R list object.

```
library(jsonlite)

query="vivo"

#form the URL
URL <- …

json <- fromJSON(url)

#get the address
address <- …

#get the lat and lng coordinates
lat <- …
lng <- …
```

If we want to display the location on a map view, it is also possible to do it in the R environment using the **leaflet** package.

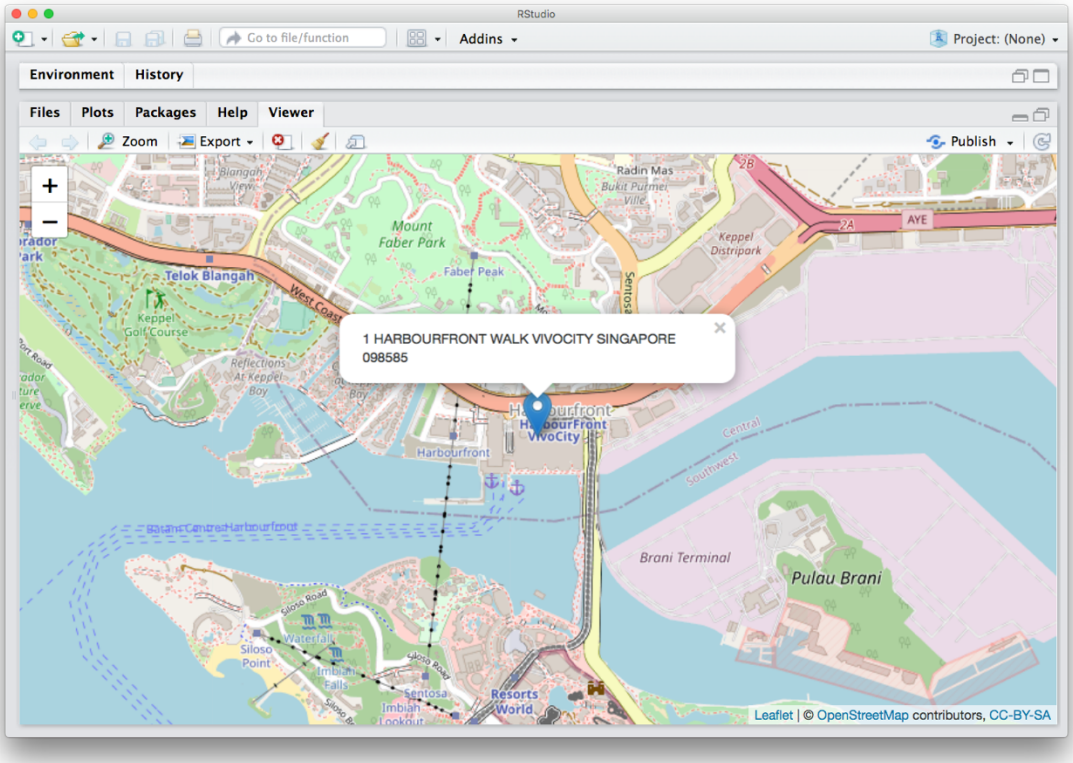
```
library(jsonlite)
library(leaflet)

…

#get the address
address <- …

#get the lat and lng coordinates
lat <- …
lng <- …

#show the location as a map in R
m <- leaflet() %>%
  addTiles() %>%  # Add default OpenStreetMap map tiles
  addMarkers(lng=lng, lat=lat, popup=address)
m  # Print the map
```



You can try searching for other location with the OneMap Search API. Note that it is possible to plot multiple markers on the leaflet map if the OneMap Search API returns multiple search results.

Burpple Web Scraper using RSelenium

In this section, we will be implementing a small proof of concept of how we can use Reselenium to do web scraping of dynamic loading websites. We will be using Burpple (https://www.burpple.com) as an example.

The skeleton of the web scraper is found below. Try to understand what each line is doing and complete the remaining portions.

```
library(RSelenium)

#depends on the chrome version installed on your machine
#when first time executing the rsDriver() command,
#it will download the different drivers
driver = rsDriver(browser=c('chrome'), chromever="94.0.4606.61")
remoteDriver = driver[["client"]]

remoteDriver$navigate("https://www.burpple.com/search/sg?q=Breakfast+%
26+Brunch")

#we will press the Load More button 3 times for 4 pages worth of results
#TODO: find the LOAD MORE button
loadButton <- …
```

```
loadButton$clickElement()


#sleep 5 seconds to let it load more
Sys.sleep(5)
loadButton$clickElement()

#sleep 5 seconds to let it load more
Sys.sleep(5)
loadButton$clickElement()

#wait for the data to load
Sys.sleep(5)

#TODO: find the elements containing the restaurant name
restaurantElements <- …

#TODO: retrieve the restaurant names from restaurantElements
#      and save it in a character vector
…

print(restaurants)

#close browser
remoteDriver$close()

#stop selenium server
#need to stop server if not you can't connect again
driver[["server"]]$stop()
```

The restaurants vector should look something like the following:

```
> print(restaurants)
 [1] "Oberstrasse"
 [2] "Elixir Boutique Roasters"
 [3] "Bread Yard"
 [4] "Cafe Milligram"
 [5] "Kream & Kensho"
 [6] "The Lobby Lounge (InterContinental Singapore)"
 [7] "Fuel Plus+"
 [8] "BURN"
 [9] "Sarnies (Telok Ayer)"
[10] "The Communal Place"
[11] "The Coffee Academics (Scotts Square)"
[12] "South Union Park"
[13] "The Forage Cafe"
[14] "Brine"
[15] "One Man Coffee (Upper Thomson)"
[16] "The Beast"
[17] "Wheeler'S Yard"
[18] "Platform 1094"
[19] "Sideways"
[20] "Carrara Cafe"
[21] "The Social Space (Chinatown)"
[22] "Food Barn (Fusionopolis)"
[23] "Tolido'S Espresso Nook"
[24] "The Coffee Academics (Raffles City)"
```

```
[25] "The Rebel Company Cafe & Bar"
[26] "King And The Pawn"
[27] "Five Oars Coffee Roasters"
[28] "Fresh Fruits Lab (FFL)"
[29] "Halcyon & Crane"
[30] "LÖWe'F Artisanal"
[31] "The Assembly Ground (The Cathay)"
[32] "Keong Saik Bakery"
[33] "Whisk & Paddle"
[34] "Two Cranes"
[35] "Rookery (Hong Leong Building)"
[36] "The M Plot"
[37] "Egg Stop (Paya Lebar Square)"
[38] "The Garden Slug"
[39] "Apollo Coffee Bar"
[40] "Crossings CafÉ"
[41] "Bacha Coffee (ION Orchard)"
[42] "Common Chefs"
[43] "Little Oasis"
[44] "Flavour Flings"
[45] "Kafe UTU"
[46] "Yardbird Southern Table & Bar"
[47] "The Muffinry By Bakery & Bar"
[48] "Bread & Hearth (Keong Saik)"
```