Chapter 3: Network Clustering and Optimization

Kevin Kam Fung Yuen
PhD, Senior Lecturer, School of Business, Singapore University of Social Sciences
kfyuen@suss.edu.sg, kevinkf.yuen@gmail.com

June 5, 2019

Contents

1 Background

The case of this chapter adopts the tutorial from (Yuen, 2018).

EAT DELIVERY, a company specializing in food order delivery services, would like to improve their delivery order assignment process. Therefore, they are planning to develop an automated system to optimize the delivery order assignment and route schedule for their riders. Our task is to help the company to design and develop the system. This chapter demonstrates the general guidelines as below.

2 Data Visualization

2.1 Map visualization

Firstly, we need to install `ggmap` package.

```
install.packages("ggmap")
```

Next, the *ggmap* package is referred to it.

```
library(ggmap)
```

We load the map data saved in a RData file.

```
load(file = "data/sgMap.RData")
```

> Although *ggmap* is free, use of *ggmap* is not completely free. The map data were achieved from `google map API service` with using *ggmap* package in R. As the `google map API service` will charge any query services and the user needs to register an account with credit card in order to get an `api_key`, this chapter will not cover tutorial how to use `ggmap` to deal with Google map data. Readers feeling interested should refer to *ggmap* documentation.

After the data is loaded, we use ggmap to display the map data.

```
ggmap(sgMap)
```

3 Challenges

In a lunch time, the delivery company received a lot of orders in *orders.csv*, and the locations of the riders available are given in *riders.csv*.

3.1 Orders

All the orders from customers in Singapore are presented in the data file, *orders.csv*. Let's take a look for this data set.

```
orders <- read.csv("data/orders.csv",row.names=1)
head(orders)
#The size of the dataset
dim(orders)
```

We found that there are 2965 food delivery orders from the customers , and their relevant locations are given.

3.2 Riders

The data file, *riders.csv*, includes location information of riders who are available to take up new deliveries. Let's take a look for this data set.

```
riders <- read.csv("data/riders.csv",row.names=1)
head(riders)
#The size of the dataset
dim(riders)
```

We found that 100 riders are ready for the food delivery, and their relevant locations are given.

3.3 Display by Simple plots

We could just use the simple *plot* and *points* functions to present the spatial data in two different data sets.

```
plot(orders[,2:3], col="red")
points(riders[,2:3], col="blue")
```

If we have limited resources to access external map services or reduce the development cost, this method for visualization is not a bad choice.

3.4 Display on Map

For the better looking, the distribution of the orders is shown in Red points on the map.

```
ggmap(sgMap)+
  geom_point(aes(x = Longitude, y = Latitude), data = orders, colour="red", size=1)
```

The distribution of the riders is visualized in blue on the map.

```
ggmap(sgMap)+
geom_point(aes(x = Longitude, y = Latitude), data = riders, colour="blue", size=2)
```

Now we combine two data sets to present on the same map.

```
# Type the code here
```

4 Goals and Deliverables

Our goal is to come up with an automated system providing EAT DELIVERY with the case assignments for each rider. The automated case assignment system will address the challenges as below:

1. How many orders will be assigned to each rider?
2. Which orders will be assigned for each rider?
3. What are the best routes for all riders to fulfill the orders?
4. How many percentages of customer orders are delivered on time, providing that the service promise of food order deliver is no more than one hour?

5 Assumptions

The case is provided by a local anonymous company and is further revised to fit for the purpose of this hackathon. To reduce the complexity of of the problem, we have made several assumptions.

5.1 Assumption 1

Each rider is at his/her current location, where the restaurants are located and have the unlimited food supply to fulfill all customer orders. Therefore, you do not have to consider the case of the riders having to visit the restaurants to pick up food once the orders are assigned. After the riders have delivered all orders assigned, they do not need to return to the original starting point.

5.2 Assumption 2

The actual travel paths along roads are not considered. The distance of the path between any two location points is only calculated by Havensine Distance Function presented as below.

Havensine Distance Function (in meter)

- long: Ending Longitude
- lat: Ending Latitude
- slong: Starting Longitude
- slat: Starting Latitude

```
Havensine <-function(slong,slat,long,lat){
  a=6378137*acos(
    cos((pi/180)*lat)*cos((pi/180)*long)*cos((pi/180)*slat)*cos((pi/180)*slong)
    + cos((pi/180)*lat)*sin((pi/180)*long)*cos((pi/180)*slat)*sin((pi/180)*slong)+
      sin((pi/180)*lat)*sin((pi/180)*slat));
  round(a,3)
}

#Test the function
Havensine(103.7911,1.2797,103.7712,1.3243)
```

Alternatively, the *geosphere::distHaversine()* function may be used to calculate the Havensine distance. Make sure that your package has been installed by executing *install.packages("geosphere")* before it is used.

```
library("geosphere")
distHaversine(c(103.7911,1.2797),c(103.7712,1.3243))
```

5.3 Assumption 3

Each order takes **1.5 minutes** to reach the customer when the rider arrives at the customer's building. The riders' travelling speed is **48km/hr** between two locations. Traffic lights and traffic congestion are not considered. The company's promise to the customers is to deliver the orders within one hour.

5.4 Others

You may make more assumptions for the system if it is needed.

6 A Basic Solution Guide

There are a number of methods to solve the problems. The following demonstration illustrates one of the methods to find the solution with the purpose for you to understand the challenges better. There should be the other better solutions for you to explore.

6.1 Divide the orders data into different subsets

We may develop or apply a method to divide the orders dataset into different subsets assigned to the dedicated riders.

```
# type the code to use kmeans to seperate the orders into appropiate portions.
```

Visualize the results by a simple plot.

```
# type the code here to visualize the results by using k-means
```

The Rider 5068 is taken as an example. The orders assigned to him/her may be like in this way.

```
# Type the code to display which customer orders assigned to rider 5068 by kmeans
```

We visualize the assignment on the map by the basic *plot* and *points* functions.

```
# type your code here for the plot for Rider 5068 and the related orders
```

Alternatively, the cluster is visualized on the map.

```
# load(file = "data/sgMap68.RData") # a variable sgMapCi will be loaded into Global Environment
# type your code here
```

Is it easy for us to recommend the tour for this single case? How can we design a solution to recommend the optimal tour for Rider 5068 to reach all the order points with the least effort?

6.2 Best routes

We may use *TSP* package to provide the solution as below by applying Nearest Neighbor method, `nn`. so firstly we install *TSP* package.

```
install.packages("TSP")
```

Load the *TSP* package, check how to use the *ESTP* function with *help* document and example(s).

```
library("TSP")
?ETSP
example(ETSP)
```

```
riderArea=rbind(ridersPoints[ci,],clusterMembers) #rider is the starting point
areaNames=rownames(riderArea)

etsp <- ETSP(riderArea,labels =areaNames) # Euclidean , use tsp to find the Havensine distance
tour <- solve_TSP(etsp,method="nn", start=1L)
stPath=as.integer(tour)
areaNames[stPath]

resultPath=riderArea[areaNames[stPath],]
```

The route can be visualized on the graph.

```
plot(resultPath, col="red")  # plot the orders
s <- seq(nrow(resultPath)-1)  # for s+1 later
x=resultPath[,1]
y=resultPath[,2]
arrows(x[s], y[s], x[s+1], y[s+1], col = "red",length = 0.1) # plot the path
points(ridersPoints["5068",], col="blue") #plot the rider
```

The route is visualized on the map in this way.

```
#ggmap(sgMapCi)
# type your code here
# geom_point
# geom_path
```

However, there should be the other better method to solve this.

- We may use TSP or find the other package and configure the settings to provide the better solution.
- We must have more credits if we develop and use our own new algorithms for the better use.

6.3 Evaluation

We perform the similar calculation for the rest riders. A table for the solution the Delivery Orders Assignment and Routing Information for all Riders is generated.

```
resultPath
totalDistance(resultPath)
```

6.4 All order

Similarly, we can use *for* loop to calculate the total distance and time of each rider to its all orders.

```
speed=800  #800m /min  =48km/hr
perOrdertime= 1.5 #mins
```

Once we have the results in the data table, we can generate two files to store the results for further use. Providing that you define **resultTable** as the **data.frame** variable for the result, we use the R functions below to save the results into the files.

```
save(resultTable,file="resultTable.Rda")
write.csv(resultTable,file="resultTable.csv",row.names = FALSE)
```

The details of the structure of the solution data format are presented as below.

```
dim(resultTable)
str(resultTable)
```

some riders' solutions are demonstrated as below. We use *kable* function to present the table in html with better look.

```
kable(resultTable[c(6,45,89,100),],row.names = FALSE)
```

6.5 Service quality

On the basis of the results in the table, we can calculate how many late delivery orders (i.e. *more than one hour*) to the customers. A example of the code below is used to find how many **riders** who could not complete the orders on time.

```
sum(resultTable$Total_time>60)
#more code...
```

According to the result,we can take closer looks into the order assignments and routes to make further improvement in order to minimize the number of late delivery orders and delivery time to the customers.

7 References

K.K.F. Yuen, Introduction and Tutorial Guide on SUSS-Microsoft Analytics Hackathon 2018, 12 pages report plus dataset and case writing, and codes. (Delivery system) https://github.com/kkfyuen/SMAH2018Guide

8 Exercise

In addition to the four criteria stated above, you may suggest additional measurement criteria to evaluate your approach and outputs. Extra credits will be given if you can provide better solution by changing or making some assumptions to make the solution more realistic and practical.

Please make sure that your programme can at least produce a csv file with the required format presented in *Table format* shown in this paper.

The evaluation criteria also include the quality of your code, report and Each should be clear and concise.

You are encouraged to submit the solution by solving the problem as much as you can. You need to submit a *report* and your codes. The *Rda* and *csv* files with the required solution format are needed to be submitted. You can define your own report format. The report should not be more than *10 pages*.