In-class Exercise 4: Preparing Spatial Interaction Modelling Variables

2023-02-13

## Overview

A healthy baby need healthy food. Likewise, a well calibrated Spatial Interaction Model need conceptually logical and well prepared propulsiveness and attractiveness variables. In this in-class exercise, you will gain hands-on experience on preparing propulsiveness and attractiveness variables require for calibrating spatial interaction models. By the end of this in-class exercise, you will be able to:

* perform geocoding by using SLA OneMap API,
* convert an aspatial data into a simple feature tibble data.frame,
* perform point-in-polygon count analysis, and
* append the propulsiveness and attractiveness variables onto a flow data.

## Getting Started

To get start, the following R packages will be loaded into R environment. They are:

* tidyverse, provide a family of modern R packages for data import, wrangling

pacman::p\_load(tidyverse, sf, httr,  
 tmap)

## Counting number of schools in each URA Planning Subzone

### Downloading General information of schools data from data.gov.sg

To get started, you are required to download *General information of schools* data set of School Directory and Information from [data.gov.sg](https://beta.data.gov.sg/).

|  |
| --- |
| Important |
| We assume that the downloaded School Directory and Information is placed in a sub-folder called data/aspatial/). |

### Geocoding using SLA API

Address geocoding, or simply geocoding, is the process of taking a aspatial description of a location, such as an address or postcode, and returning geographic coordinates, frequently latitude/longitude pair, to identify a location on the Earth’s surface.

Singapore Land Authority (SLA) supports an online geocoding service called [OneMap API](https://www.onemap.gov.sg/apidocs/). The [Search](https://www.onemap.gov.sg/apidocs/apidocs) API looks up the address data or 6-digit postal code for an entered value. It then returns both latitude, longitude and x,y coordinates of the searched location.

The code chunks below will perform geocoding using [SLA OneMap API](https://www.onemap.gov.sg/docs/#onemap-rest-apis). The input data will be in csv file format. It will be read into R Studio environment using *read\_csv* function of **readr** package. A collection of http call functions of **httr** package of R will then be used to pass the individual records to the geocoding server at OneMap.

Two tibble data.frames will be created if the geocoding process completed successfully. They are called found and not\_found. found contains all records that are geocoded correctly and not\_found contains postal that failed to be geocoded.

Lastly, the found data table will joined with the initial csv data table by using a unique identifier (i.e. POSTAL) common to both data tables. The output data table will then save as an csv file called found.

url<-"https://www.onemap.gov.sg/api/common/elastic/search"  
  
csv<-read\_csv("data/aspatial/Generalinformationofschools.csv")  
postcodes<-csv$`postal\_code`  
  
found<-data.frame()  
not\_found<-data.frame()  
  
for(postcode in postcodes){  
 query<-list('searchVal'=postcode,'returnGeom'='Y','getAddrDetails'='Y','pageNum'='1')  
 res<- GET(url,query=query)  
   
 if((content(res)$found)!=0){  
 found<-rbind(found,data.frame(content(res))[4:13])  
 } else{  
 not\_found = data.frame(postcode)  
 }  
}

Next, the code chunk below will be used to combine both *found* and *not\_found* data.frames into a single tibble data.frame called *merged*. At the same time, we will write *merged* and *not\_found* tibble data.frames into two separate csv files called *schools* and *not\_found* respectively.

merged = merge(csv, found, by.x = 'postal\_code', by.y = 'results.POSTAL', all = TRUE)  
write.csv(merged, file = "data/aspatial/schools.csv")  
write.csv(not\_found, file = "data/aspatial/not\_found.csv")

|  |
| --- |
| Do it yourself! |
| * With the help of Google Map, located the location information of the ungeocoded school by using it’s postcode. * Update the results.LATITUDE and results.LONGITUDE fields of the ungeocoded record in schoolss.csv manually. |

### Tidying schools data.frame

In this sub-section, you will import *schools.csv* into R environment and at the same time tidying the data by selecting only the necessary fields as well as rename some fields.

|  |
| --- |
| Do it yourself! |
| Using the steps you learned in Hands-on Exercise 1, perform the following tasks:   * import *schools.csv* in R environment as an tibble data.frame called *schools*, * rename *results.LATITUDE* and *results.LONGITUDE* to *latitude* and *longitude* respectively, * retain only *postal\_code*, *school\_name*, *latitude* and *longitude* in schools tibble data.frame. |

schools <- read\_csv("data/aspatial/schools.csv") %>%  
 rename(latitude = "results.LATITUDE",  
 longitude = "results.LONGITUDE")%>%  
 select(postal\_code, school\_name, latitude, longitude)

### Converting an aspatial data into sf tibble data.frame

Next, you will convert schools tibble data.frame data into a simple feature tibble data.frame called *schools\_sf* by using values in latitude and longitude fields.

Refer to [st\_as\_sf()](https://r-spatial.github.io/sf/reference/st_as_sf.html) of sf package.

schools\_sf <- st\_as\_sf(schools,   
 coords = c("longitude", "latitude"),  
 crs=4326) %>%  
 st\_transform(crs = 3414)

Let’s save *schools\_sf* into an rds file by usign the code chunk below.

write\_rds(schools\_sf, "data/rds/schools.rds")

### Plotting a point simple feature layer

To ensure that *schools* sf tibble data.frame has been projected and converted correctly, you can plot the schools point data for visual inspection.

First, let us import *MPSZ-2019* shapefile into R environment and save it as an sf tibble data.frame called *mpsz*.

|  |
| --- |
| Do it yourself! |
| Using the step your learned in previous hands-on exercise, import *MPSZ-2019* shapefile into R as sf tibble data.frame and name it *mpsz*. |

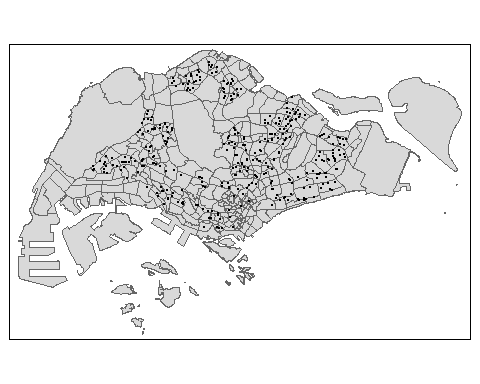
mpsz <- st\_read(dsn = "data/geospatial/",  
 layer = "MPSZ-2019") %>%  
 st\_transform(crs = 3414)

Reading layer `MPSZ-2019' from data source   
 `D:\tskam\ISSS624\In-class\_Ex\In-class\_Ex4\data\geospatial'   
 using driver `ESRI Shapefile'  
Simple feature collection with 332 features and 6 fields  
Geometry type: MULTIPOLYGON  
Dimension: XY  
Bounding box: xmin: 103.6057 ymin: 1.158699 xmax: 104.0885 ymax: 1.470775  
Geodetic CRS: WGS 84

#### Do it yourself!

Using the steps you learned in previous exercises, create a point symbol map showing the location of schools with OSM as the background map.

tmap\_options(check.and.fix = TRUE)  
tm\_shape(mpsz) +  
 tm\_polygons() +  
tm\_shape(schools\_sf) +  
 tm\_dots()



### Performing point-in-polygon count process

Next, we will count the number of schools located inside the planning subzones.

|  |
| --- |
| Do it yourself! |
| Using the steps you learned from previous hands-on exercises, count the number of schools within each planning subzone by using lengths() of **Base** and st\_intersects() of **sf** package. |

mpsz$`SCHOOL\_COUNT`<- lengths(  
 st\_intersects(  
 mpsz, schools\_sf))

It is always a good practice to examine the summary statistics of the derived variable.

|  |
| --- |
| Do it yourself! |
| Using the steps you learned in previous exercises, compute and display the summary statistics of *sch\_count* field. |

summary(mpsz$SCHOOL\_COUNT)

Min. 1st Qu. Median Mean 3rd Qu. Max.   
 0.000 0.000 0.000 1.054 2.000 12.000

|  |
| --- |
| Important |
| The summary statistics above reveals that there are excessive 0 values in SCHOOL\_COUNT field. If log() is going to use to transform this field, additional step is required to ensure that all 0 will be replaced with a value between 0 and 1 but not 0 neither 1. |

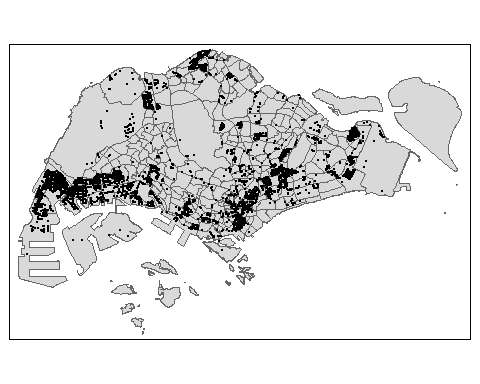
## Data Integration and Final Touch-up

|  |
| --- |
| Do it yourself! |
| Using the steps you learned in earlier sub-sections, count the number of Business points in each planning subzone. |

business\_sf <- st\_read(dsn = "data/geospatial",  
 layer = "Business")

Reading layer `Business' from data source   
 `D:\tskam\ISSS624\In-class\_Ex\In-class\_Ex4\data\geospatial'   
 using driver `ESRI Shapefile'  
Simple feature collection with 6550 features and 3 fields  
Geometry type: POINT  
Dimension: XY  
Bounding box: xmin: 3669.148 ymin: 25408.41 xmax: 47034.83 ymax: 50148.54  
Projected CRS: SVY21 / Singapore TM

tmap\_options(check.and.fix = TRUE)  
tm\_shape(mpsz) +  
 tm\_polygons() +  
tm\_shape(business\_sf) +  
 tm\_dots()



mpsz$`BUSINESS\_COUNT`<- lengths(  
 st\_intersects(  
 mpsz, business\_sf))

summary(mpsz$BUSINESS\_COUNT)

Min. 1st Qu. Median Mean 3rd Qu. Max.   
 0.00 0.00 2.00 19.73 13.00 307.00

Now, it is time for us to bring in the flow\_data.rds saved after Hands-on Exercise 3.

flow\_data <- read\_rds("data/rds/flow\_data.rds")  
flow\_data

Simple feature collection with 14734 features and 10 fields  
Geometry type: LINESTRING  
Dimension: XY  
Bounding box: xmin: 5105.594 ymin: 25813.33 xmax: 49483.22 ymax: 49552.79  
Projected CRS: SVY21 / Singapore TM  
First 10 features:  
 ORIGIN\_SZ DESTIN\_SZ MORNING\_PEAK dist ORIGIN\_AGE7\_12 ORIGIN\_AGE13\_24  
1 AMSZ01 AMSZ01 1998 50.0000 310 710  
2 AMSZ01 AMSZ02 8289 810.4491 310 710  
3 AMSZ01 AMSZ03 8971 1360.9294 310 710  
4 AMSZ01 AMSZ04 2252 840.4432 310 710  
5 AMSZ01 AMSZ05 6136 1076.7916 310 710  
6 AMSZ01 AMSZ06 2148 805.2979 310 710  
7 AMSZ01 AMSZ07 1620 1798.7526 310 710  
8 AMSZ01 AMSZ08 1925 2576.0199 310 710  
9 AMSZ01 AMSZ09 1773 1204.2846 310 710  
10 AMSZ01 AMSZ10 63 1417.8035 310 710  
 ORIGIN\_AGE25\_64 DESTIN\_AGE7\_12 DESTIN\_AGE13\_24 DESTIN\_AGE25\_64  
1 2780 310.00 710.00 2780.00  
2 2780 1140.00 2770.00 15700.00  
3 2780 1010.00 2650.00 14240.00  
4 2780 980.00 2000.00 11320.00  
5 2780 810.00 1920.00 9650.00  
6 2780 1050.00 2390.00 12460.00  
7 2780 420.00 1120.00 3620.00  
8 2780 390.00 1150.00 4350.00  
9 2780 1190.00 3260.00 13350.00  
10 2780 0.99 0.99 0.99  
 geometry  
1 LINESTRING (29501.77 39419....  
2 LINESTRING (29501.77 39419....  
3 LINESTRING (29501.77 39419....  
4 LINESTRING (29501.77 39419....  
5 LINESTRING (29501.77 39419....  
6 LINESTRING (29501.77 39419....  
7 LINESTRING (29501.77 39419....  
8 LINESTRING (29501.77 39419....  
9 LINESTRING (29501.77 39419....  
10 LINESTRING (29501.77 39419....

Notice that this is an sf tibble data.frame and the features are polylines linking the centroid of origins and destination planning subzone.

|  |
| --- |
| Do it yourself |
| Using the steps your learned in Hands-on Exercise 3, append *SCHOOL\_COUNT* and *BUSINESS\_COUNT* into flow\_data sf tibble data.frame. |

mpsz\_tidy <- mpsz %>%  
 st\_drop\_geometry() %>%  
 select(SUBZONE\_C, SCHOOL\_COUNT, BUSINESS\_COUNT)

Now, we will append SCHOOL\_COUNT and BUSINESS\_COUNT fields from mpsz\_tidy data.frame into flow\_data sf tibble data.frame by using the code chunk below.

flow\_data <- flow\_data %>%  
 left\_join(mpsz\_tidy,  
 by = c("DESTIN\_SZ" = "SUBZONE\_C")) %>%  
 rename(TRIPS = MORNING\_PEAK,  
 DIST = dist)

|  |
| --- |
| Important |
| Note that the unique join field used is DESTIN\_SZ of flow\_data and SUBZONE\_C of mpsz\_tidy. Do you know why? |

### Checking for variables with zero values

Since Poisson Regression is based of log and log 0 is undefined, it is important for us to ensure that no 0 values in the explanatory variables.

In the code chunk below, summary() of Base R is used to compute the summary statistics of all variables in *wd\_od* data frame.

summary(flow\_data)

ORIGIN\_SZ DESTIN\_SZ TRIPS DIST   
 Length:14734 Length:14734 Min. : 1 Min. : 50   
 Class :character Class :character 1st Qu.: 14 1st Qu.: 3346   
 Mode :character Mode :character Median : 76 Median : 6067   
 Mean : 1021 Mean : 6880   
 3rd Qu.: 426 3rd Qu.: 9729   
 Max. :232187 Max. :26136   
 ORIGIN\_AGE7\_12 ORIGIN\_AGE13\_24 ORIGIN\_AGE25\_64 DESTIN\_AGE7\_12   
 Min. : 0.99 Min. : 0.99 Min. : 0.99 Min. : 0.99   
 1st Qu.: 240.00 1st Qu.: 440.00 1st Qu.: 2200.00 1st Qu.: 240.00   
 Median : 700.00 Median : 1350.00 Median : 6810.00 Median : 720.00   
 Mean :1031.86 Mean : 2268.84 Mean :10487.62 Mean :1033.40   
 3rd Qu.:1480.00 3rd Qu.: 3260.00 3rd Qu.:15770.00 3rd Qu.:1500.00   
 Max. :6340.00 Max. :16380.00 Max. :74610.00 Max. :6340.00   
 DESTIN\_AGE13\_24 DESTIN\_AGE25\_64 SCHOOL\_COUNT BUSINESS\_COUNT   
 Min. : 0.99 Min. : 0.99 Min. : 0.000 Min. : 0.00   
 1st Qu.: 460.00 1st Qu.: 2200.00 1st Qu.: 0.000 1st Qu.: 0.00   
 Median : 1420.00 Median : 7030.00 Median : 1.000 Median : 3.00   
 Mean : 2290.35 Mean :10574.46 Mean : 1.583 Mean : 16.17   
 3rd Qu.: 3260.00 3rd Qu.:15830.00 3rd Qu.: 2.000 3rd Qu.: 12.00   
 Max. :16380.00 Max. :74610.00 Max. :12.000 Max. :307.00   
 geometry   
 LINESTRING :14734   
 epsg:3414 : 0   
 +proj=tmer...: 0

The print report above reveals that variables *ORIGIN\_AGE7\_12*, *ORIGIN\_AGE13\_24*, *ORIGIN\_AGE25\_64*, *DESTIN\_AGE7\_12*, *DESTIN\_AGE13\_24*, *DESTIN\_AGE25\_64* consist of 0 values.

In view of this, code chunk below will be used to replace zero values to 0.99.

flow\_data$SCHOOL\_COUNT <- ifelse(  
 flow\_data$SCHOOL\_COUNT == 0,  
 0.99, flow\_data$SCHOOL\_COUNT)  
flow\_data$BUSINESS\_COUNT <- ifelse(  
 flow\_data$BUSINESS\_COUNT == 0,  
 0.99, flow\_data$BUSINESS\_COUNT)

You can run the summary() again.

summary(flow\_data)

ORIGIN\_SZ DESTIN\_SZ TRIPS DIST   
 Length:14734 Length:14734 Min. : 1 Min. : 50   
 Class :character Class :character 1st Qu.: 14 1st Qu.: 3346   
 Mode :character Mode :character Median : 76 Median : 6067   
 Mean : 1021 Mean : 6880   
 3rd Qu.: 426 3rd Qu.: 9729   
 Max. :232187 Max. :26136   
 ORIGIN\_AGE7\_12 ORIGIN\_AGE13\_24 ORIGIN\_AGE25\_64 DESTIN\_AGE7\_12   
 Min. : 0.99 Min. : 0.99 Min. : 0.99 Min. : 0.99   
 1st Qu.: 240.00 1st Qu.: 440.00 1st Qu.: 2200.00 1st Qu.: 240.00   
 Median : 700.00 Median : 1350.00 Median : 6810.00 Median : 720.00   
 Mean :1031.86 Mean : 2268.84 Mean :10487.62 Mean :1033.40   
 3rd Qu.:1480.00 3rd Qu.: 3260.00 3rd Qu.:15770.00 3rd Qu.:1500.00   
 Max. :6340.00 Max. :16380.00 Max. :74610.00 Max. :6340.00   
 DESTIN\_AGE13\_24 DESTIN\_AGE25\_64 SCHOOL\_COUNT BUSINESS\_COUNT   
 Min. : 0.99 Min. : 0.99 Min. : 0.990 Min. : 0.99   
 1st Qu.: 460.00 1st Qu.: 2200.00 1st Qu.: 0.990 1st Qu.: 0.99   
 Median : 1420.00 Median : 7030.00 Median : 1.000 Median : 3.00   
 Mean : 2290.35 Mean :10574.46 Mean : 1.987 Mean : 16.47   
 3rd Qu.: 3260.00 3rd Qu.:15830.00 3rd Qu.: 2.000 3rd Qu.: 12.00   
 Max. :16380.00 Max. :74610.00 Max. :12.000 Max. :307.00   
 geometry   
 LINESTRING :14734   
 epsg:3414 : 0   
 +proj=tmer...: 0

Notice that all the 0 values have been replaced by 0.99.

Before we move on to calibrate the Spatial Interaction Models, let us save flow\_data sf tibble data.frame into an rds file. Call the file *flow\_data\_tidy*.

write\_rds(flow\_data,  
 "data/rds/flow\_data\_tidy.rds")