**Topic: Taxi Trajectory Analysis for Finding Pick-up Hotspots.**

**General Notice**

The purpose of the Hotspot Analysis is to provide taxi drivers with a service that can guide them to places where more passengers are waiting. The results of the analysis will have private and public value, allowing taxi drivers to benefit more and citizens to receive better taxi service.

**Exercise 1: Data preprocessing**

Data preprocessing means that data is processed appropriately for analysis prior to analysis. Here, R, a statistical analysis tool, Python, the most commonly used computer language and QGIS are used to preprocess data.

**Step 1. Using R**

First of all, the taxi data received from ‘NaviCall’ adds to a unique folder. If you use R to read the raw data, you can see the fields you need and the fields you don't need. To use only the fields you need, extract Date, x-coordinate, y-coordinate, and transfer car code(승빈차코드) from original taxi data. After that, the transfer vehicle code 1 which is Pickup and the transfer vehicle code 2 which is drop off are extracted. Then, a character string of the date is sliced to select only a time zone, select morning time of 06:00 to 09:00 and night time of 21:00 to 24:00. And save each as a csv file. Ultimately we extract boarding in morning and afternoon, alighting in morning and afternoon.

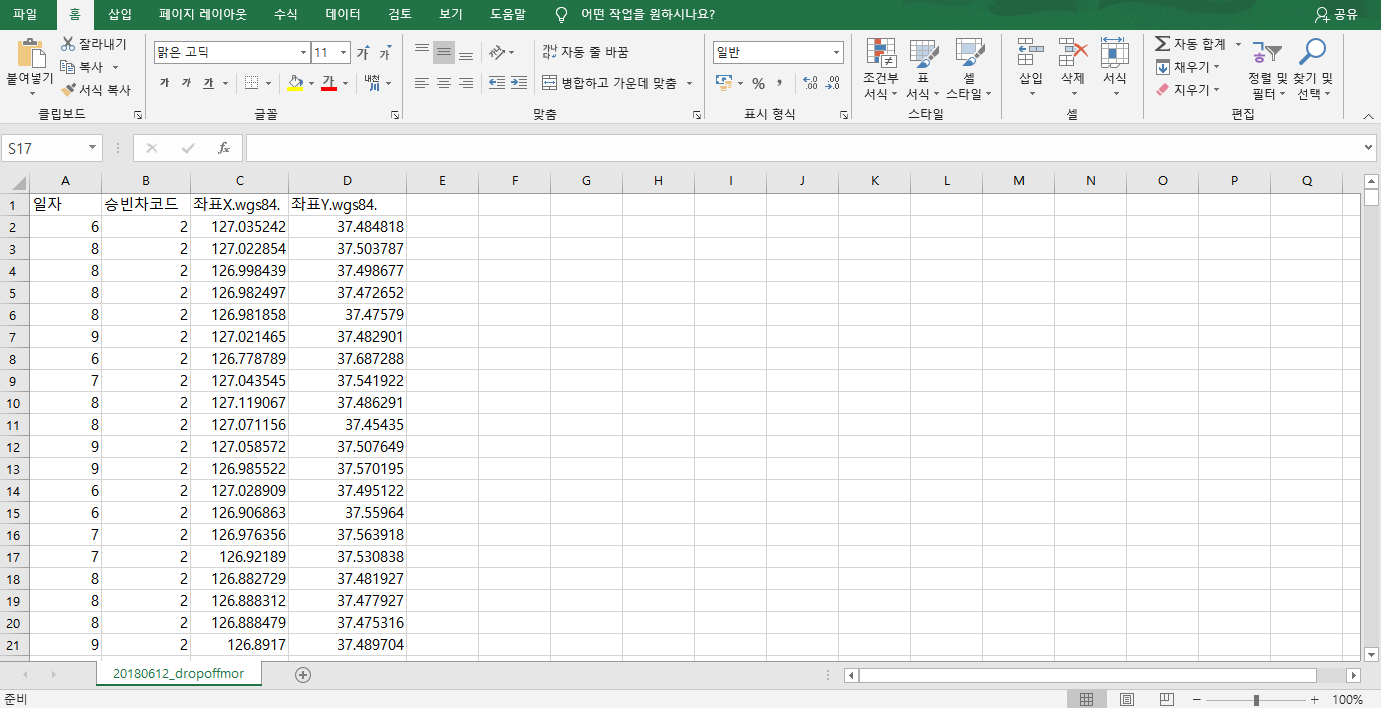


Figure 1

The Excel above is the result of data preprocessing. A total of four Excel files should be created to analyze boarding and alighting, morning and afternoon hours. And below is the example R code for preprocessing data.

**[R code]**

library(dplyr)

dat = read.csv(‘C://Temp//20180612\_taxi.csv’) # taxi csv 읽기

head(dat) #csv 앞부분 확인

dat = dat %>% select(일자, 승빈차코드, 좌표X.wgs84.,좌표Y.wgs84.) *#select fields*

dat = dat[-1,] #*delete first column(useless column)*

pick\_up = dat %>% filter(승빈차코드 == 1) *#extract transfer car code 1*

drop\_off = dat %>% filter(승빈차코드 ==2) *# extract transfer car code 2*

pick\_up$일자 = substr(pick\_up$일자, 9, 10) *#extract hh in yyyymmddhhmmss*

drop\_off$일자 = substr(drop\_off$일자, 9, 10)

pick\_up\_mor = pick\_up %>% filter(일자 %in% c('06','07','08','09')) *# extract morning time*

pick\_up\_nig = pick\_up %>% filter(일자 %in% c('21','22','23','24')) *# extract afternoon time*

drop\_off\_mor = drop\_off %>% filter(일자 %in% c('06','07','08','09'))

drop\_off\_nig = drop\_off %>% filter(일자 %in% c('21','22','23','24'))

write.csv(pick\_up\_mor, ‘C://Temp//20180612\_pickupmor.csv’) *#save as CSV file*

write.csv(pick\_up\_nig, ‘C://Temp//20180612\_pickupnig.csv’)

write.csv(drop\_off\_mor, ‘C://Temp//20180612\_dropoffmor.csv’)

write.csv(drop\_off\_nig, ‘C://Temp//20180612\_dropoffnig.csv’)

**[Python code]**

import pandas as pd

data = pd.read\_csv('C:/Users/scsi\_public/Desktop/intern/carloc\_hst-20180615.dat', sep = ",", engine = 'python', encoding = "cp949")

*# Delete useless columns*

data.drop(['#차량번호', '생성월', '지역코드', '좌표X(bessel)', '좌표Y(bessel)', '회사코드', '운전자ID'], axis = 1, inplace = True)

*# Extract Car code 1&2*

data1 = data[data.승빈차코드 == 1]

data2 = data[data.승빈차코드 == 2]

*# Convert the data type to float*

data1 = data1.astype(float)

data2 = data2.astype(float)

*# Extract morning time and afternoon time*

data1\_morning = data1[(data1['일자'] >= 20180615060000) & (data1['일자'] < 20180615090000)]

data2\_morning = data2[(data2['일자'] >= 20180615060000) & (data2['일자'] < 20180615090000)]

data1\_night = data1[(data1['일자'] >= 20180615210000) & (data1['일자'] < 20180615240000)]

data2\_night = data2[(data2['일자'] >= 20180615210000) & (data2['일자'] < 20180615240000)]

*# Save as csv file*

data1\_morning.to\_csv("C:\\Users\\scsi\_public\\Desktop\\intern\\take\_morning.csv", index = False, encoding = 'cp949')

data1\_night.to\_csv("C:\\Users\\scsi\_public\\Desktop\\intern\\take\_night.csv", index = False, encoding = 'cp949')

data2\_morning.to\_csv("C:\\Users\\scsi\_public\\Desktop\\intern\\getoff\_morning.csv", index = False, encoding = 'cp949')

data2\_night.to\_csv("C:\\Users\\scsi\_public\\Desktop\\intern\\getoff\_night.csv", index = False, encoding = 'cp949')

**Step 2. Using QGIS**

In order to do hotspots analysis, it is necessary to find out how many boarding and alighting passengers there are in each region(dong). After displaying the CSV file created by STEP 1 as point and the Seoul’s shapefile(Seoul WGS map.shp) on the QGIS screen(Figure 2), we will use an analysis tool to find out the number of points inside the polygon.

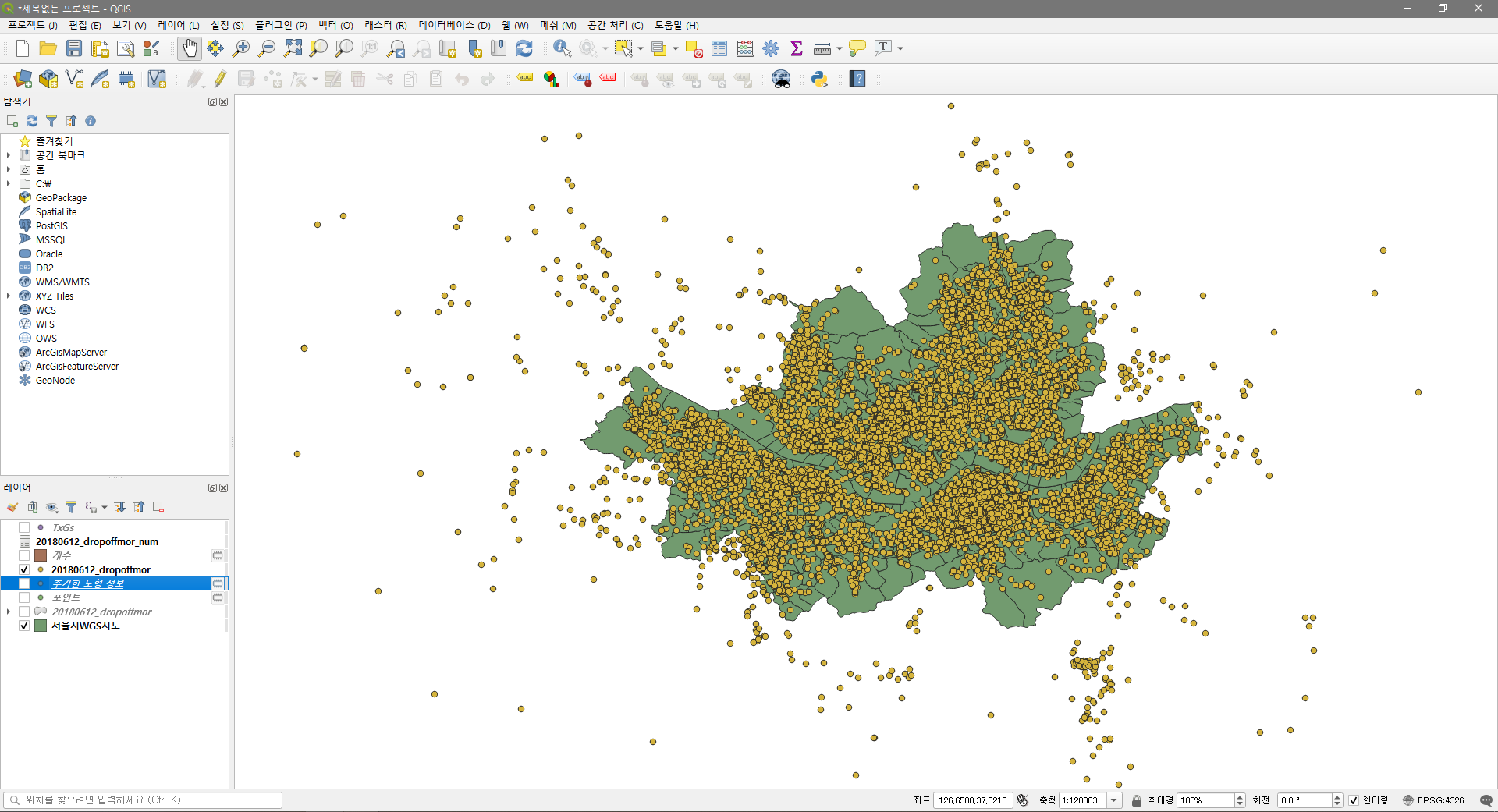


Figure 2

You can see how many points in each polygon. In other words, you can see how many times people got on and off the taxi in polygon(dong). Then ‘NUM’ layer having the number of points in the polygon is generated. (You can named the layer.)

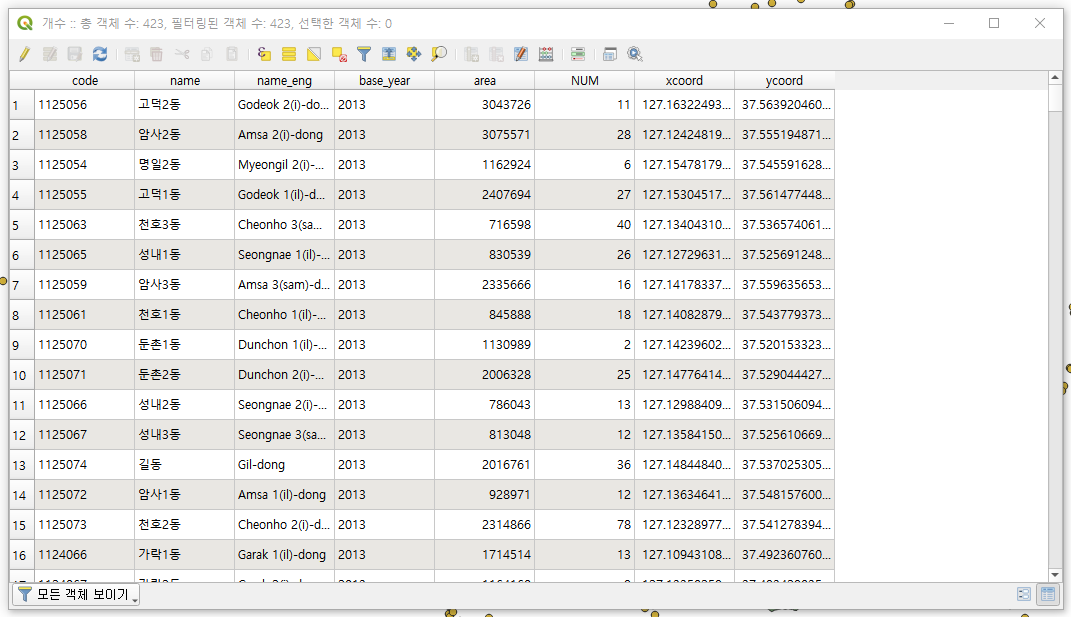


Figure 3

Next, since coordinates by region(dong) are not known, coordinates are obtained by using a toolbox. Centroid coordinates for each polygon can be obtained by [Add Shape Properties]. (Figure 4) After obtaining the coordinates, the ‘NUM’ field and ‘polygon X, Ycoordinate’ field can joinned by using the same dong name so that we can represent NUM as a shapefile. (Figure 5)

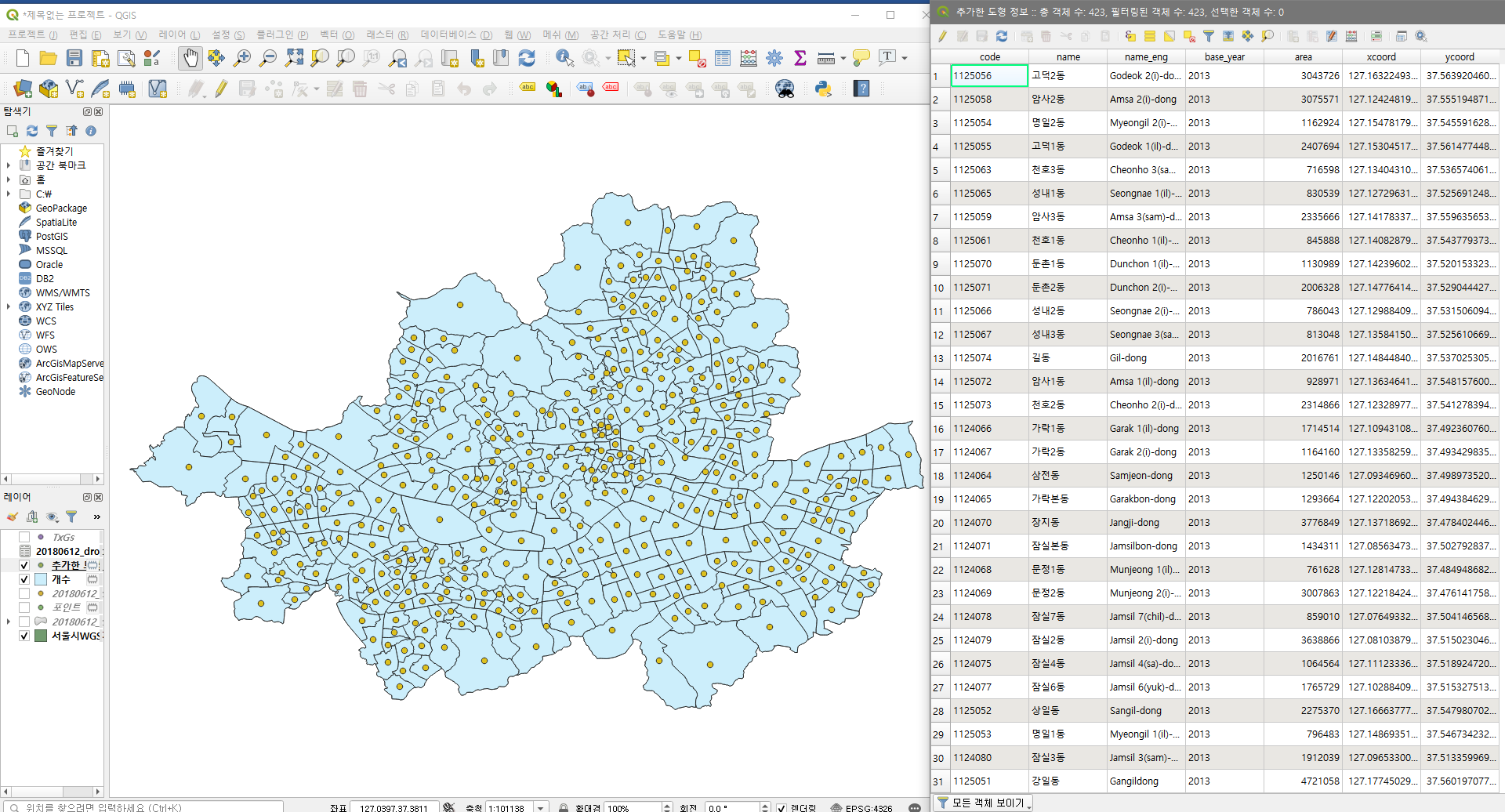


Figure 4

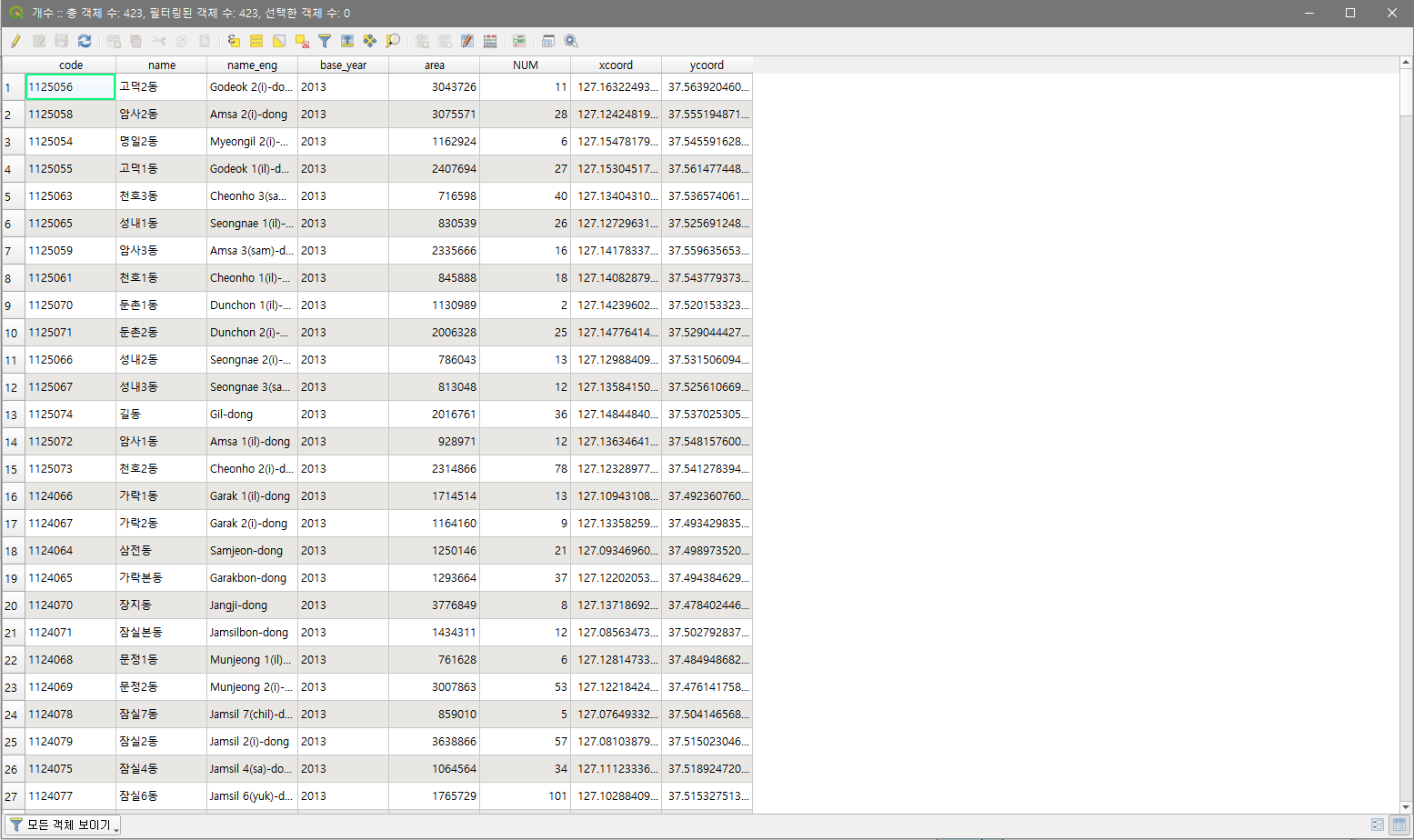


Figure 5

Finally export the joined shapefile to CSV file for analysis.

**Exercise 2: Hotspot Analysis**

Analyze the exported csv file using r. Below is the example R code for hotspot analysis.

**[R code]**

library(spdep)

library(readr)

library(foreign)

Tx = read.csv(‘C://Temp//20180612\_pickupmor.csv’) *#read CSV file*

xycoords = cbind(Tx$xcoords, Tx$ycoords)

nb3 = dnearneigh(xycoords, 0, 3, longlat = TRUE) *# Calculate adjacent areas of local points in Euclidean distance or km (longlat = TRUE)*

G3s = localG(Tx$NUM, nb2listw(include.self(nb3), style = ‘B’)) *#local spatial autocorrelation*

show(G3s)

Gs = cbind(Tx, G3) *# Combine hotspot analysis results with the original file*

zvalue = scale(dat$G3) *#Normalize Hotspot Analysis Results*

TxGs = cbind(Gs, zvalue)

Write.table(TxGs, ‘C://Temp//TxGs.txt’, row.names = FALSE, sep = ‘,’) #save as txt file

per = quantile(zvalue, probs = c(0.01, 0.05, 0.1, 0.9, 0.95, 0.99)) # *Visualization Reference Value (Figure 6)*



Figure 6

# Extract the shp file from qgis at the last step in step2 in Exercise 1. And use Geoda to get weight in shp file. And save in gwt format.

**[Python code]**

import pandas as pd

from pysal.esda.getisord import G\_Local

import numpy as np

from sklearn.preprocessing import StandardScaler

import pysal

Tx = pd.read\_csv('C:\\Users\\scsi\_public\\Desktop\\intern\\taxinew\\take\_morning.csv')

np.random.seed(12345)

y = Tx["NUMPOINTS"]

y = y.values

*# adjacent areas of local points*

w = pysal.open('C:/Users/scsi\_public/anaconda3/Lib/site-packages/pysal/examples/stl/seoul.gwt').read()

*# G\_Local process*

lg = G\_Local(y, w)

lg.n

len(lg.Gs)

x = lg.p\_sim

*# Scale process*

scaler = StandardScaler()

x = np.reshape(x, (-1,1))

lg\_scale = scaler.fit\_transform(x)

lg\_scale

*# Combine hotspot analysis results with the original file*

Tx['Local\_G'] = lg\_scale

*# Save as csv file*

Tx.to\_csv("C:/Users/scsi\_public/anaconda3/Lib/site-packages/pysal/examples/stl/Tx\_LocalG.csv", index = False, encoding='cp949')

# *Visualization Reference Value*

np.percentile(Tx['Local\_G'], [10, 20, 30, 70, 80, 90], interpolation ='linear')

**Exercise 3: Visualization of Hotspot by QGIS**

Use QGIS to visualize the results of hotspot analysis. First of all, display the CSV file containing the hotspot analysis results and the Seoul shapefile on QGIS. Because CSV files appear as points, join the two files together to form a shapefile using the same ‘dong name’ as you did before. Next, click the [symbol] on the [property] menu to classify the result value and visualize them. The classification uses the previously calculated visualization reference value, negative value means Coldspot and positive value means Hotspot. Here we assign Hotspot as a bright color and Coldspot as a dark color dark.

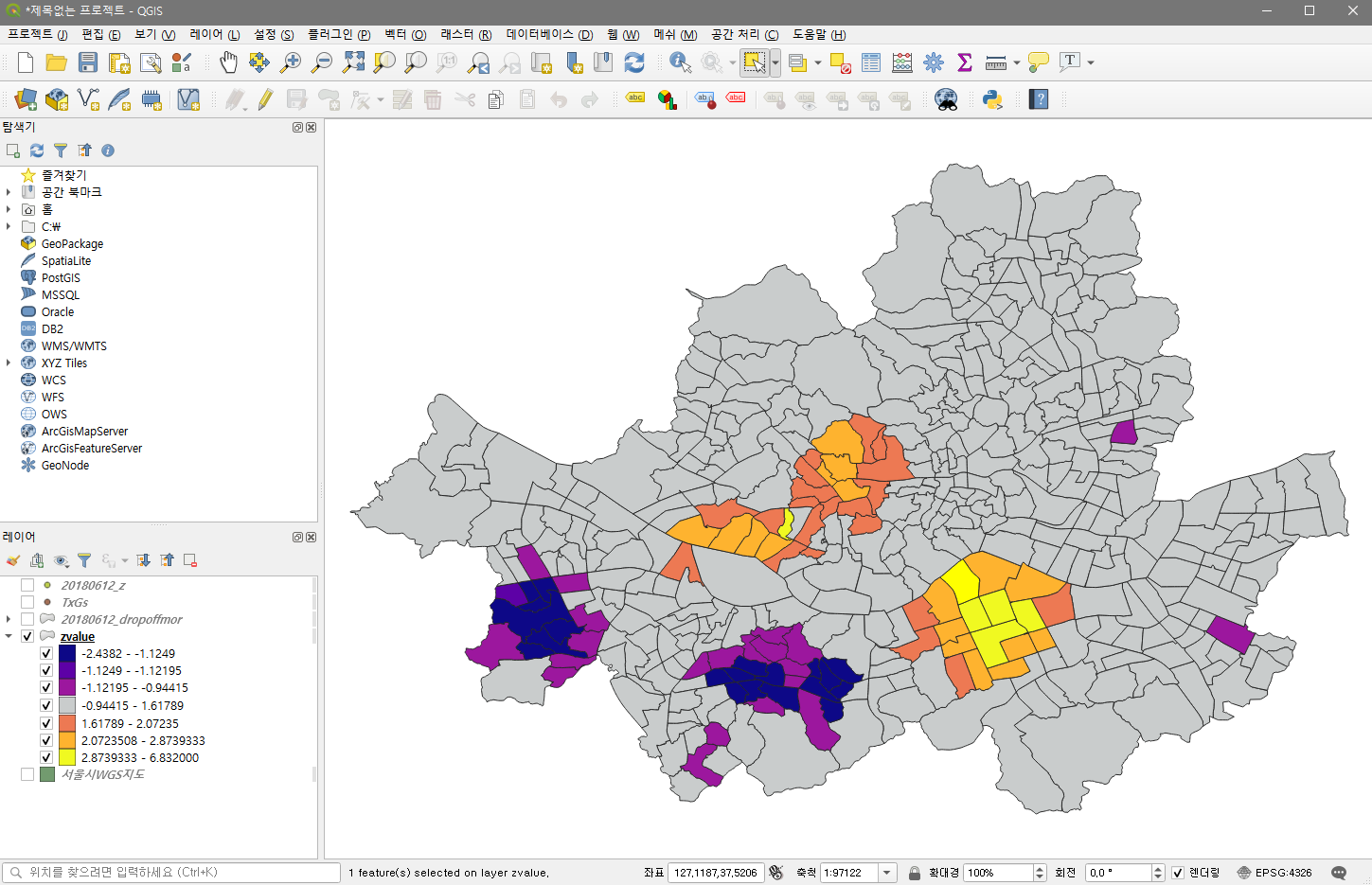


Figure 7

The above figure shows one of the four files created earlier, and Visualize the other three.