Tutorial Book: The Analysis of Distribution and Density Changes of Electric Vehicle Charging Points in London Boroughs

Zeqiang Fang

2021-01-15

Contents

Chapter 1

Introduction and Description of Data

Electric vehicle (EV) infrastructure is of importance for sustainable urban development. “UK e-charging market is recognised as one of the most advanced in Europe,” said Martin Lucas. He is a partner of Watson Farley & Williams LLP, an international law firm (2020). However, urban residents who purchase new energy vehicles have to face range anxiety, which has become a constraint on developing the EV market. EVs’ sales are lower than expected because of the potential users’ anxiety range (Bonges and Lusk, 2016).

In this report, the research question will be investigated and discussed — How do the distribution and density of EV charge change in London between 2019 and 2020? The aim is to apply theories from GIS, especially the spatial analysis method, to explore the distribution and density in these two years. Firstly, the big data of charge points, from the UK government official website, is preprocessed and cleaned. One of the analysis methods is to apply spatial pattern analysis. It is based on the number of samples in two years. It compares their distribution and density to obtain the corresponding objective value. Besides, a reproducible analysis process is established using open source spatial analysis software RStudio, which applies the advanced spatial analysis methods in clean energy and explores spatial value content to contribute to urban sustainability.

1.1 Electric Vehicle Charge Point Dataset

The NCR, a database of charge points for electric vehicles in the UK, is available for individuals and business data developers without charge (GOV.UK, 2020). Following the UK government website’s guidance, the National Chargepoint Registry (NCR) dataset was collected in CSV format.

You can access this dataset in this link. **OR**, you can also access this dataset in the github

1.2 London Boroughs Geometric Dataset

Spatial data is available on the London Datastore official website. The shape format file called Statistical GIS Boundary is the original geographic boundaries data, which is based on our spatial analysis (London Datastore, 2020). One of the variables called "GSS_CODE" can be identified via "sf," an R package, to present the London borough polygons in multiple types.

You should download this complete folder to read shape file! The link is here

Chapter 2

Environment Preparation

2.1 Environment Recommendation

- System: MacOS 10.5 / 11.0 Windows 10
- IDE: RStudio 1.3 (MacOS) R >= 3.6

2.2 Import packages for spatial analysis and map making

If there are errors take place, you can run `install.packages({missing package name})` to install packages.

```r
library(tidyverse)
library(data.table)
library(sp)
library(sf)
library(table1)
library(tm)
library(spatstat)
library(here)
library(sp)
library(rgeos)
# library(maptools)
library(tmap)
library(sf)
library(geojson)
library(geojsonio)
library(tmaptools)
library(RColorBrewer)
library(spdep)
library(lubridate)
```

Chapter 3

Data Pre-processing

3.1 Read data into R

The **National Chargepoint Register(NCR)** is a database of publicly-available chargepoints for electric vehicles in the UK established in 2011(,2021). you can access data in this link

Now, let's read original data in R

```r
UK_NCR= read.csv(here::here("dataset","national-charge-point-registry.csv"))
# This may take for a while, which depends on the speed of internet

# you can have a overview of this dataset
print("The number of rows is: ")
nrow(UK_NCR)
print("The number of columns is: ")
ncol(UK_NCR)
print("70 of all varriables are:")
head(names(UK_NCR),n = 70)
```

Tip: If you cannot successfully read this dataset, you can replace the above link with "https://raw.githubusercontent.com/Hereislittlemushroom/CASA0005_Final_Assessment/main/Dataset/national-charge-point-registry.csv"

3.2 Data Selection

Select the charge points of london area in this UK csv file You can utilise `filter` function from `dplyr` package to choose the charge point data in london boroughs

```
London_NCR = UK_NCR %>%
  dplyr::filter(  !is.na(county),
                  county == "London" |
                  county == "Greater London " |
                  county == "London Borough of Camden" |
                  county == "London Borough of Ealing" |
                  county == "London Borough of Greenwich" |
                  county == "London Borough of Hackney" |
                  county == "London Borough of Hammersmith and Fulham" |
                  county == "London Borough of Hounslow" |
                  county == "London Borough of Islington" |
                  county == "London Borough of Lambeth" |
                  county == "London Borough of Richmond upon Thames" |
                  county == "London Borough of Southwark" |
                  county == "London Borough Of Southwark" |
                  county == "London Borough of Waltham Forest" |
                  county == "London Borough of Wandsworth")
```

Check if all values in `county` are attributed to "London"

```
isLondon = London_NCR$county %>%
  unique()
isLondon
```

In the next step, you can select the valuable attributes e.g. latitude,longitude.

```
# Tip: the index of data frame starts from 1
# Select the variables by their index
London_NCR = London_NCR %>%
  select(1,4,5,13,14,15,32,35,36,38,54)

# Check the variables we have chosen and the number of rows & cols
London_NCR %>%
  names()
London_NCR %>%
  nrow()
London_NCR %>%
  ncol()
```

3.3 Data Cleaning

Map and visualisation play important roles in spatial analysis. To make a heat map for further research, you need to merge geographic information for each row in charge point dataset in the first place.

To begin with, import “PostcodesioR”. This R package offer methods to match

```
# install.packages("PostcodesioR")
library(PostcodesioR)
```

Before applying “for-loop” method to fill values in `GSS_CODE` by identifying `postcode`, you can add a new columns called `GSS_CODE` in London_NCR dataset.

```
# Attentions: you can skip this chunk because the for-loop process can take for a quite long time
# It is not necessary to stick on it, just skip!

London_NCR_GSS_Added = London_NCR %>%
  rowwise() %>%
  mutate(GSS_CODE = postcode) %>%
  # Tip: it is essential to transform numerical data into one in character
  mutate(GSS_CODE = as.character(GSS_CODE))

# Pay attention to the for loop in dataframe, it starts from 1

i = 1
for (val in London_NCR_GSS_Added$postcode) {
  try({ temp1 = PostcodesioR::postcode_lookup(val)
        if(!is.null(temp)){
          temp2 = temp1$admin_district_code[1]
          London_NCR_GSS_Added$GSS_CODE[i] = temp2
        }else{
          London_NCR_GSS_Added$GSS_CODE[i] = ""
        }
        i = i+1 }
      ,silent = TRUE)
}

# remove the rows whose value of `GSS_CODE` is empty
# There are limitations in this process because the rows missing `GSS_CODE` cannot be included in

London_NCR_GSS_Added$GSS_CODE[London_NCR_GSS_Added$GSS_CODE==""] = NA
London_NCR_GSS_Added = London_NCR_GSS_Added %>%
  filter(!is.na(GSS_CODE))
```

Finally, it is of importance to export our prepossessed data into csv file! Now we get the London_NCR_GSS_Added.csv in our “/Dataset” path.

```
# export London_NCR_GSS_Added data frame into .CSV format
library(here)
write.csv(London_NCR_GSS_Added, here::here("Dataset","London_NCR_GSS_Added.csv"), row.r
# `col.names = TRUE` is important to be writen down
```

Also, you can access this prepocessed dataset in github link: https://raw.githubusercontent.com/Hereislittlemushroom/CASA0005_Final_Assessment/main/Dataset/London_NCR_GSS_Added.csv

Chapter 4

Basic Settings

In this section, you will set the work path, import R packages, download the shape file & its folder and read datasets in RStudio.

4.1 Set the path of your project.

Before you do the research, you should set the default path. The path below is mine, you should set your **own work path**

```
setwd("/Users/fangzeqiang/Github/tutorial_bookdown/")
```

4.2 Import the shape file

What you should keep in mind is that this shape file should be run in the complete ESRI dir because there are some dependent files that the shape file might use.

```
# you can download these files from github to your local work path that you set above
# github link: https://github.com/Hereislittlemushroom/CASA0005_Final_Assessment/tree/main/Datase

London_Borough = st_read("dataset/statistical-gis-boundaries-london/ESRI/London_Borough_Excluding
```

```
## Reading layer `London_Borough_Excluding_MHW' from data source `/Users/fangzeqiang/Github/tutor
## Simple feature collection with 33 features and 7 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: 503568.2 ymin: 155850.8 xmax: 561957.5 ymax: 200933.9
## CRS:            27700
```

```
# plot the map
plot(st_geometry(London_Borough))
```

4.3 Import the processed London national chargepoint register (NCR) dataset

The reason why choosing `fread` to read is because the read method is faster than the traditional ones. You can read .CSV file by `df = fread("Dataset/London_NCR_GSS_Added.csv")`. However, in the following codes, I recommend that you read the pre-processed dataset from my github link.

```
# df = fread("Dataset/London_NCR_GSS_Added.csv")
# df = fread(here::here("dataset","London_NCR_GSS_Added.csv"))
# df = fread("http://zeqiang.fun/_book/dataset/London_NCR_GSS_Added.csv")
 df = fread("http://raw.githubusercontent.com/Hereislittlemushroom/CASA0005_Final_Asses
```

Chapter 5

The Distribution analysis of samples

You can generate a formal table to overview the distribution of samples. In order to compare samples between 2019 and 2020, you can obtain year from "dataCreated" which is timestamp format. Then you can create two columns in table (2019 & 2020).

```
#3.1 select the years
df$year = year(df$dateCreated)
table(df$year)
```

```
##
## 2017 2018 2019 2020 2021
##    1    4  696  362    3
```

```
#select 2019 & 2020
df = df[year==2019|year==2020,]
table(df$year)
```

```
##
## 2019 2020
##  696  362
```

```
#3.2 show the table of the distribution of two years samples
table1(~county|factor(year),data=df)
```

```
## [1] "<table class=\"Rtable1\">\n<thead>\n<tr>\n<th class='rowlabel firstrow lastrow'></th>\n<t
```

```
# head(df)
```

Chapter 6

Visualisation and comparison of the density of London EV charge point between two years

6.1 Data cleaning for mapping

You should split the NCR dataset to two dataset of two years, and merged these two processed dataset with geometric data respectively. Then we calculate the density for two years. Finally we can visualise and map the density

```r
#5.1 process the data to divide them by years(2019 & 2020)
df1<-subset(df,year==2019) #2019 year
df2<-subset(df,year==2020) #2020 year

# EV charge points created in 2019
# sdf1<-merge(London_Borough,df1,by="GSS_CODE")
sdf1<-merge(London_Borough,df1,by="GSS_CODE",all = TRUE)
sdf1<-sdf1[,c("GSS_CODE","geometry","longitude","latitude")]

# EV charge points created in 2020
# sdf2<-merge(London_Borough,df2,by="GSS_CODE")
sdf2<-merge(London_Borough,df2,by="GSS_CODE",all = TRUE)
sdf2<-sdf2[,c("GSS_CODE","geometry","longitude","latitude")]
```

6.2 The density of data in 2019

First, you should calculate the density and transform the unit of area from m^2 o km^2. Then, select the necessary columns and add frequency of samples grouped by `GSS_CODE`. In other words, you get the numeric and spatial data of EV charge point for each borough of London.

```
# Data preparation
nsdf1 = sdf1%>%
  add_count(GSS_CODE)%>%
  mutate(area=st_area(.))%>%
  # Use dplyr::mutate to calculate the density of the charge point for each borough
  mutate(density=n*1000*1000/area)
  # because the st_area default unit is square metre

# select the following variables---"density","GSS_CODE","n"(the count of GSS_CODE)
nsdf1 = dplyr::select(nsdf1,density,GSS_CODE, n)

nsdf1 = nsdf1%>%
  group_by(GSS_CODE)%>%
  summarise(density =first(density),GSS_CODE=first(GSS_CODE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

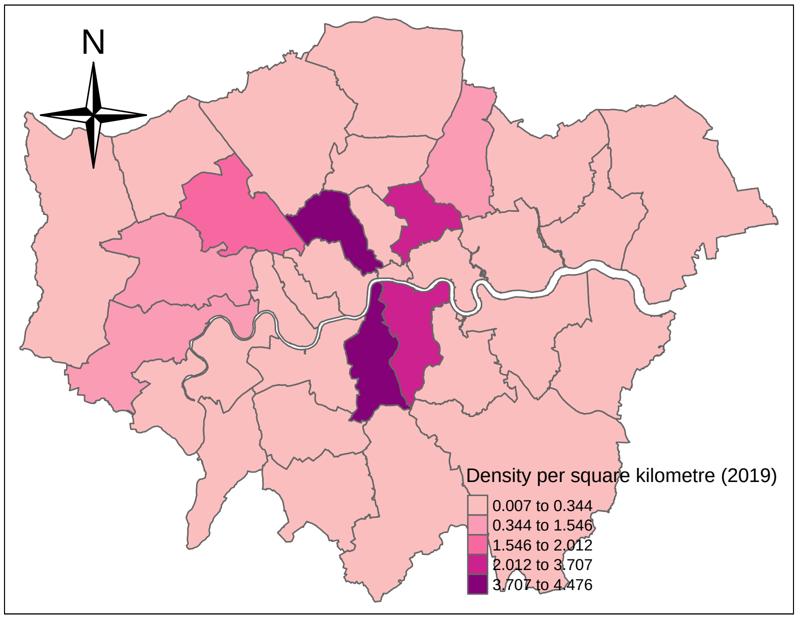
Now you can generate map after setting some variables in `tmap_mode`, `tm_compass` and `tm_polygons`.

```
tmap_mode("plot")
```

```
## tmap mode set to plotting
```

```
# plot the figure: The distribution of the density of the London charge points in 2019
tm_shape( nsdf1) +
  tm_compass( north = 0,
              type = "4star",
              text.size = 0.8,
              size = 2.5,
              show.labels = 1,
              cardinal.directions = c("N", "E", "S", "W"),
              lwd = 1,
              position = c("left","top"),
              bg.color = NA,
              bg.alpha = NA,
              just = NA,
               fontsize = 1.5) +
  tm_polygons("density",
              style="jenks",
              palette="RdPu",
              midpoint=NA,
              popup.vars=c("GSS_CODE", "density"),
              title="Density per square kilometre (2019)"
              )
```

```
## Warning: The argument fontsize of tm_compass is deprecated. It has been renamed
## to text.size
```



6.3 The density of data in 2020

Then you should conduct the similar process as the 4.2 section to draw the density of EV charge point data in 2020.

```
nsdf2 = sdf2%>%
  add_count(GSS_CODE)%>%
  mutate(area=st_area(.))%>%
  mutate(units::set_units(area,km^2))%>%
  mutate(density=n*1000*1000/area)
```

```
nsdf2 = dplyr::select(nsdf2,density,GSS_CODE, n)

nsdf2 = nsdf2%>%
  group_by(GSS_CODE)%>%
  summarise(density =first(density),GSS_CODE=first(GSS_CODE))
```

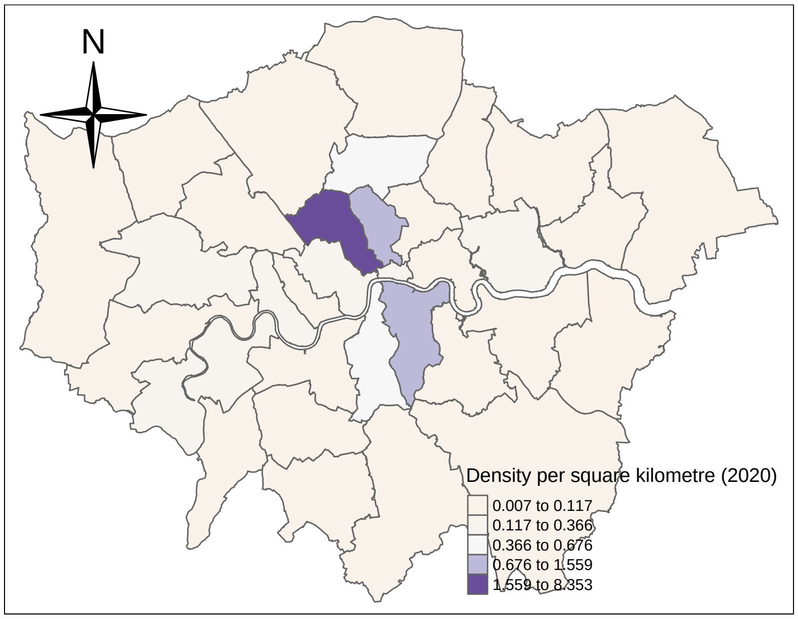
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
tmap_mode("plot")
```

```
## tmap mode set to plotting
```

```
tm_shape( nsdf2) +
  tm_compass( north = 0,
              type = "4star",
              text.size = 0.8,
              size = 2.5,
              show.labels = 1,
              cardinal.directions = c("N", "E", "S", "W"),
              lwd = 1,
              position = c("left","top"),
              bg.color = NA,
              bg.alpha = NA,
              just = NA,
              fontsize = 1.5) +
  tm_polygons("density",
              style="jenks",
              palette="PuOr",
              midpoint=NA,
              popup.vars=c("GSS_CODE", "density"),
              title="Density per square kilometre (2020)")
```

```
## Warning: The argument fontsize of tm_compass is deprecated. It has been renamed
## to text.size
```


N
Density per square kilometre (2020)
0.007 to 0.117
0.117 to 0.366
0.366 to 0.676
0.676 to 1.559
1.559 to 8.353

Chapter 7

Analysing Spatial Autocorrelation with Moran's I

Since the sample in 2019 & 2020 is too small to run a good result, you can analysis the autocorrelation based on the samples which contain these two years.

7.1 Generate the data for analysis

This process is similar to the 4.2 and 4.3 data preparation.

```r
sdf = merge(London_Borough,df,by="GSS_CODE",all = TRUE)
sdf = sdf[,c("GSS_CODE","geometry","longitude","latitude")]
nsdf = sdf%>%
  add_count(GSS_CODE)%>%
  mutate(area=st_area(.))%>%
  mutate(density=n*1000*1000/area)

nsdf = dplyr::select(nsdf,density,GSS_CODE, n)

nsdf = nsdf%>%
  group_by(GSS_CODE)%>%
  summarise(density = first(density), GSS_CODE = first(GSS_CODE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

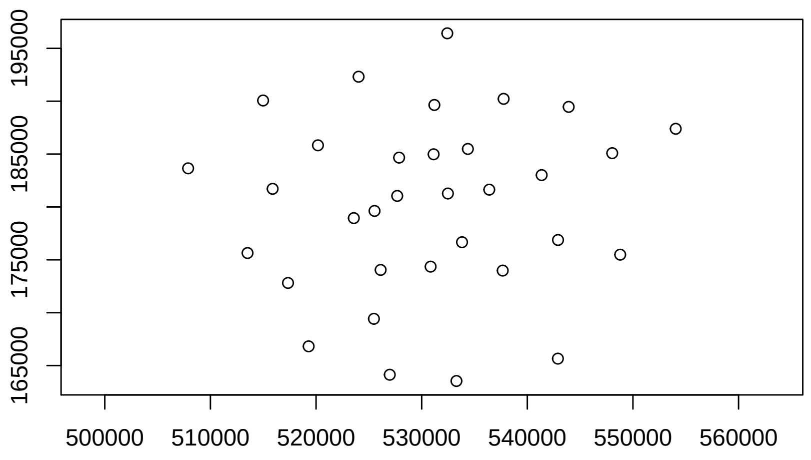
7.2 Centroids and neighbour list

Plot the centroids of all boroughs in London

```
coordsW = nsdf%>%
   st_centroid()%>%
   st_geometry()
```

```
## Warning in st_centroid.sf(.): st_centroid assumes attributes are constant over
## geometries of x
```

```
plot(coordsW,axes=TRUE)
```

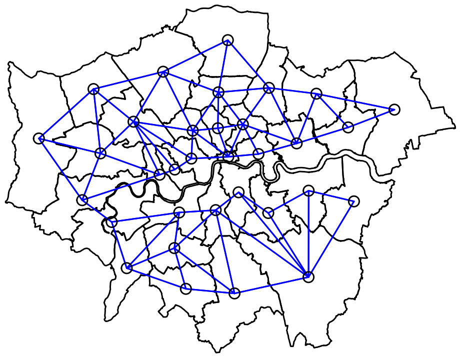


Create a neighbours list

```
LWard_nb = nsdf %>%
 poly2nb(.,queen=T)
```

Plot the neighbours list we create

```
plot(nsdf$geometry)
plot(LWard_nb, st_geometry(coordsW), col="blue", add = T)
```



Create a spatial weights object from these weights, which can contribute to the further analysis autocorrelation analysis (Moran's I test)

```
Lward.lw = nb2listw(LWard_nb, style="C")
head(Lward.lw$neighbours)
```

```
## [[1]]
## [1]  7 12 19 30 33
##
## [[2]]
## [1] 16 25 26
##
## [[3]]
## [1]  5  7 10 14 15
##
## [[4]]
## [1]  6 11
##
## [[5]]
## [1]  3  7  9 13 15 20 33
##
## [[6]]
## [1]  4  8 11 22 23 28
```

7.3 Calculate the Global Moran'I Index

Conduct the global Moran's I test to get the value.

```
 I_LWard_Global_Density = nsdf %>%
   pull(density) %>%
   as.vector()%>%
   moran.test(.,Lward.lw)
names(I_LWard_Global_Density)
```

```
## [1] "statistic"   "p.value"     "estimate"    "alternative" "method"
## [6] "data.name"
```

```
head(I_LWard_Global_Density)
```

```
## $statistic
## Moran I statistic standard deviate
##                         0.03513093
##
## $p.value
## [1] 0.4859877
##
## $estimate
## Moran I statistic       Expectation          Variance
##      -0.028283348      -0.031250000       0.007131055
##
## $alternative
## [1] "greater"
##
## $method
## [1] "Moran I test under randomisation"
##
## $data.name
## [1] ".  \nweights: Lward.lw    \n"
```

Conduct the Local Moran's I test in these two years

```
 I_LWard_Local_Density = nsdf %>%
   pull(density) %>%
   as.vector()%>%
   localmoran(., Lward.lw)%>%
   as_tibble()
```

Merge the moran test result with the geometric dataset. The I value is stored in “density_Iz” and the Z value is stored in “density_Iz”.

```
nsdf<-nsdf%>%
   mutate(density_I = as.numeric(I_LWard_Local_Density$Ii))%>%
   mutate(density_Iz =as.numeric(I_LWard_Local_Density$Z.Ii))
summary(nsdf$density_I)
summary(nsdf$density_Iz)
```

7.4 Interactive visulisation of the distribution of the local Moran results

For drawing an interactive map, you should set “view” variables in `tmap_mode` function. You can set break box by the minimum and maximum value of Moran.

```
tmap_mode("view")
```

```
## tmap mode set to interactive viewing
```

```
#set the group and colour
summary(nsdf$density_Iz)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.813183 -0.007425  0.133178  0.046786  0.449411  0.975067
```

```
# breaks1 = seq(-3,1,0.5)
breaks1 = c(-2,-1,-0.1,-0.01,0,0.01,0.1,0.45,1,1.5 )
# breaks2 = c(-1000,-2.58,-1.96,-1.65,1.65,1.96,2.58,1000)
# Depends on the max and min value in Moran's I
MoranColours = rev(brewer.pal(8, "RdGy"))
# Plot the map
tm_shape(nsdf) +
  tm_polygons("density_Iz",
              style="fixed",
              breaks=breaks1,
              palette=MoranColours,
              midpoint=NA,
              title="Local Moran's I,EV charge points in London")
```

7.5 GI score

You can repeat the similar process as you can in 5.4 section to do the data preparation.

```
Gi_LWard_Local_Density = nsdf %>%
  pull(density) %>%
  as.vector()%>%
  localG(., Lward.lw)

# To get the Min and Max value
summary(Gi_LWard_Local_Density)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.8861 -0.4445 -0.1112  0.1998  0.4569  2.6561
```

Calculate the Gi score and transform the data type into numeric one.

```
Gi_nsdf = nsdf %>%
  mutate(density_G = as.numeric(Gi_LWard_Local_Density))
```

Based on the summary result of GI score to set the scale of breaks. Finally, the interactive map of Gi score distribution can be created.

```
GIColours = rev(brewer.pal(8, "RdBu"))

# This breaks box bases on the summary result
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# -0.8861 -0.4445 -0.1112  0.1998  0.4569  2.6561
breaks_GI = c(-2,-1.5,-1,-0.4,-0.2,0,0.2,0.5,1,2,2.5,3)

# Now plot on an interactive map
tmap_mode("view")
```

```
## tmap mode set to interactive viewing
```

```
tm_shape(Gi_nsdf) +
    tm_polygons("density_G",
        style="fixed",
        breaks=breaks_GI,
        palette=GIColours,
        midpoint=NA,
        title="Gi*, EV charge points in London")
```