In-class Exercise 2: Emerging Hot Spot Analysis: sfdep methods

2023-02-13

# Overview

Emerging Hot Spot Analysis (EHSA) is a spatio-temporal analysis method for revealing and describing how hot spot and cold spot areas evolve over time. The analysis consist of four main steps:

* Building a space-time cube,
* Calculating Getis-Ord local Gi\* statistic for each bin by using an FDR correction,
* Evaluating these hot and cold spot trends by using Mann-Kendall trend test,
* Categorising each study area location by referring to the resultant trend z-score and p-value for each location with data, and with the hot spot z-score and p-value for each bin.

# Getting started

## Installing and Loading the R Packages

As usual, p\_load() of **pacman** package will be used to check if the necessary packages have been installed in R, if yes, load the packages on R environment.

Five R packages are need for this in-class exercise, they are: sf, sfdep, tmap, plotly and tidyverse.

|  |
| --- |
| Do It Yourself! |
| Using the steps you learned in previous lesson, install and load **sf**, **tmap**, **sfdep** and **tidyverse** packages into R environment. |

pacman::p\_load(sf, sfdep, tmap, plotly, tidyverse)

# The Data

For the purpose of this in-class exercise, the Hunan data sets will be used. There are two data sets in this use case, they are:

* Hunan, a geospatial data set in ESRI shapefile format, and
* Hunan\_GDPPC, an attribute data set in csv format.

Before getting started, reveal the content of *Hunan\_GDPPC.csv* by using Notepad and MS Excel.

## Importing geospatial data

In the code chunk below, st\_read() of **sf** package is used to import *Hunan* shapefile into R.

hunan <- st\_read(dsn = "data/geospatial",   
 layer = "Hunan")

Reading layer `Hunan' from data source   
 `D:\tskam\ISSS624\In-class\_Ex\In-class\_Ex2\data\geospatial'   
 using driver `ESRI Shapefile'  
Simple feature collection with 88 features and 7 fields  
Geometry type: POLYGON  
Dimension: XY  
Bounding box: xmin: 108.7831 ymin: 24.6342 xmax: 114.2544 ymax: 30.12812  
Geodetic CRS: WGS 84

### Do it Yourself

Using the steps you learned in previous lesson, examine the content *hunan* sf data.frame

## Importing attribute table

In the code chunk below, read\_csv() of **readr** is used to import *Hunan\_GDPPC.csv* into R.

GDPPC <- read\_csv("data/aspatial/Hunan\_GDPPC.csv")

### Do it Yourself

Using the steps you learned in previous lesson, examine the content the *GDPPC* tibble data.frame.

# Creating a Time Series Cube

Before getting started, students must read this [article](https://sfdep.josiahparry.com/articles/spacetime-s3.html) to learn the basic concept of spatio-temporal cube and its implementation in sfdep package.

In the code chunk below, [spacetime()](https://sfdep.josiahparry.com/reference/spacetime.html) of sfdep is used to create an spacetime cube.

GDPPC\_st <- spacetime(GDPPC, hunan,  
 .loc\_col = "County",  
 .time\_col = "Year")

Next, is\_spacetime\_cube() of sfdep package will be used to varify if GDPPC\_st is indeed an space-time cube object.

is\_spacetime\_cube(GDPPC\_st)

[1] TRUE

The **TRUE** return confirms that *GDPPC\_st* object is indeed an time-space cube.

## Computing Gi\*

Next, we will compute the local Gi\* statistics.

### Deriving the spatial weights

The code chunk below will be used to identify neighbors and to derive an inverse distance weights.

GDPPC\_nb <- GDPPC\_st %>%  
 activate("geometry") %>%  
 mutate(nb = include\_self(st\_contiguity(geometry)),  
 wt = st\_inverse\_distance(nb, geometry,  
 scale = 1,  
 alpha = 1),  
 .before = 1) %>%  
 set\_nbs("nb") %>%  
 set\_wts("wt")

|  |
| --- |
| Things to learn from the code chunk above |
| * activate() of dplyr package is used to activate the geometry context * mutate() of dplyr package is used to create two new columns *nb* and *wt*. * Then we will activate the data context again and copy over the nb and wt columns to each time-slice using set\_nbs() and set\_wts()   + row order is very important so do not rearrange the observations after using set\_nbs() or set\_wts(). |

Note that this dataset now has neighbors and weights for each time-slice.

head(GDPPC\_nb)

# A tibble: 6 × 5  
 Year County GDPPC nb wt   
 <dbl> <chr> <dbl> <list> <list>   
1 2005 Anxiang 8184 <int [6]> <dbl [6]>  
2 2005 Hanshou 6560 <int [6]> <dbl [6]>  
3 2005 Jinshi 9956 <int [5]> <dbl [5]>  
4 2005 Li 8394 <int [5]> <dbl [5]>  
5 2005 Linli 8850 <int [5]> <dbl [5]>  
6 2005 Shimen 9244 <int [6]> <dbl [6]>

## Computing Gi\*

We can use these new columns to manually calculate the local Gi\* for each location. We can do this by grouping by *Year* and using local\_gstar\_perm() of sfdep package. After which, we use unnest() to unnest *gi\_star* column of the newly created *gi\_starts* data.frame.

gi\_stars <- GDPPC\_nb %>%   
 group\_by(Year) %>%   
 mutate(gi\_star = local\_gstar\_perm(  
 GDPPC, nb, wt)) %>%   
 tidyr::unnest(gi\_star)

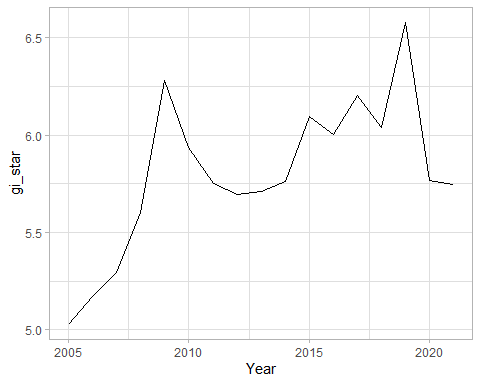
# Mann-Kendall Test

With these Gi\* measures we can then evaluate each location for a trend using the Mann-Kendall test. The code chunk below uses Changsha county.

cbg <- gi\_stars %>%   
 ungroup() %>%   
 filter(County == "Changsha") |>   
 select(County, Year, gi\_star)

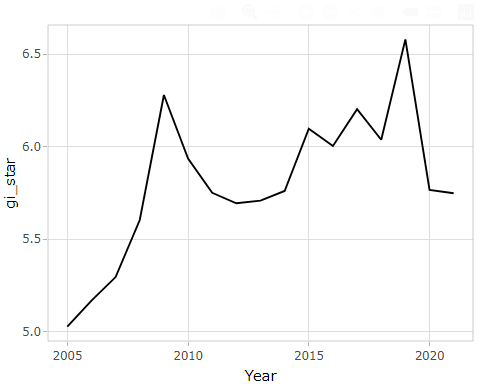
Next, we plot the result by using ggplot2 functions.

ggplot(data = cbg,   
 aes(x = Year,   
 y = gi\_star)) +  
 geom\_line() +  
 theme\_light()



We can also create an interactive plot by using ggplotly() of **plotly** package.

p <- ggplot(data = cbg,   
 aes(x = Year,   
 y = gi\_star)) +  
 geom\_line() +  
 theme\_light()  
  
ggplotly(p)



cbg %>%  
 summarise(mk = list(  
 unclass(  
 Kendall::MannKendall(gi\_star)))) %>%   
 tidyr::unnest\_wider(mk)

# A tibble: 1 × 5  
 tau sl S D varS  
 <dbl> <dbl> <dbl> <dbl> <dbl>  
1 0.485 0.00742 66 136. 589.

In the above result, sl is the p-value. This result tells us that there is a slight upward but insignificant trend.

We can replicate this for each location by using group\_by() of dplyr package.

ehsa <- gi\_stars %>%  
 group\_by(County) %>%  
 summarise(mk = list(  
 unclass(  
 Kendall::MannKendall(gi\_star)))) %>%  
 tidyr::unnest\_wider(mk)

## Arrange to show significant emerging hot/cold spots

emerging <- ehsa %>%   
 arrange(sl, abs(tau)) %>%   
 slice(1:5)

## Performing Emerging Hotspot Analysis

Lastly, we will perform EHSA analysis by using [emerging\_hotspot\_analysis()](https://sfdep.josiahparry.com/reference/emerging_hotspot_analysis.html) of sfdep package. It takes a spacetime object x (i.e. GDPPC\_st), and the quoted name of the variable of interest (i.e. GDPPC) for .var argument. The k argument is used to specify the number of time lags which is set to 1 by default. Lastly, nsim map numbers of simulation to be performed.

ehsa <- emerging\_hotspot\_analysis(  
 x = GDPPC\_st,   
 .var = "GDPPC",   
 k = 1,   
 nsim = 99  
)

### Visualising the distribution of EHSA classes

In the code chunk below, ggplot2 functions ised used to reveal the distribution of EHSA classes as a bar chart.

ggplot(data = ehsa,  
 aes(x = classification)) +  
 geom\_bar()

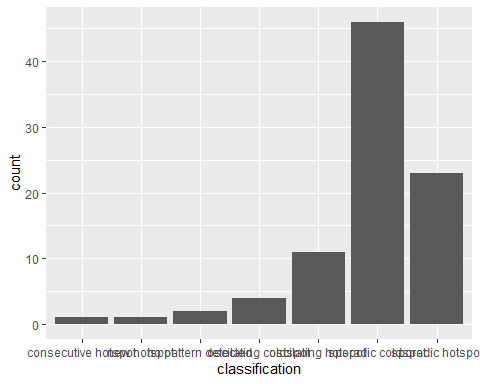


Figure above shows that sporadic cold spots class has the high numbers of county.

### Visualising EHSA

In this section, you will learn how to visualise the geographic distribution EHSA classes. However, before we can do so, we need to join both *hunan* and *ehsa* together by using the code chunk below.

hunan\_ehsa <- hunan %>%  
 left\_join(ehsa,  
 by = join\_by(County == location))

Next, tmap functions will be used to plot a categorical choropleth map by using the code chunk below.

ehsa\_sig <- hunan\_ehsa %>%  
 filter(p\_value < 0.05)  
tmap\_mode("plot")  
tm\_shape(hunan\_ehsa) +  
 tm\_polygons() +  
 tm\_borders(alpha = 0.5) +  
tm\_shape(ehsa\_sig) +  
 tm\_fill("classification") +   
 tm\_borders(alpha = 0.4)

