In-class Exercise 2: Spatial Weights - sfdep methods

2023-02-10

## Overview

This in-class introduces an alternative R package to spdep package you used in Hands-on Exercise 6. The package is called [**sfdep**](https://sfdep.josiahparry.com/index.html). According to Josiah Parry, the developer of the package, “sfdep builds on the great shoulders of **spdep** package for spatial dependence. sfdep creates an sf and tidyverse friendly interface to the package as well as introduces new functionality that is not present in spdep. sfdep utilizes list columns extensively to make this interface possible.”

## Getting started

### Installing and Loading the R Packages

Four R packages will be used for this in-class exercise, they are: sf, sfdep, tmap, tidyverse.

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

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

## 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\_2012, an attribute data set in csv format.

### Importing geospatial data

|  |
| --- |
| Do It Yourself! |
| Using the steps you learned in previous lesson, import *Hunan* shapefile into R environment as an sf data frame. |

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

### Importing attribute table

|  |
| --- |
| Do It Yourself! |
| Using the steps you learned in previous lesson, import *Hunan\_2012.csv* into R environment as an tibble data frame. |

hunan2012 <- read\_csv("data/aspatial/Hunan\_2012.csv")

### Combining both data frame by using left join

|  |
| --- |
| Do It Yourself! |
| Using the steps you learned in previous lesson, combine the Hunan sf data frame and Hunan\_2012 data frame. Ensure that the output is an sf data frame. |

hunan\_GDPPC <- left\_join(hunan, hunan2012) %>%  
 select(1:4, 7, 15)

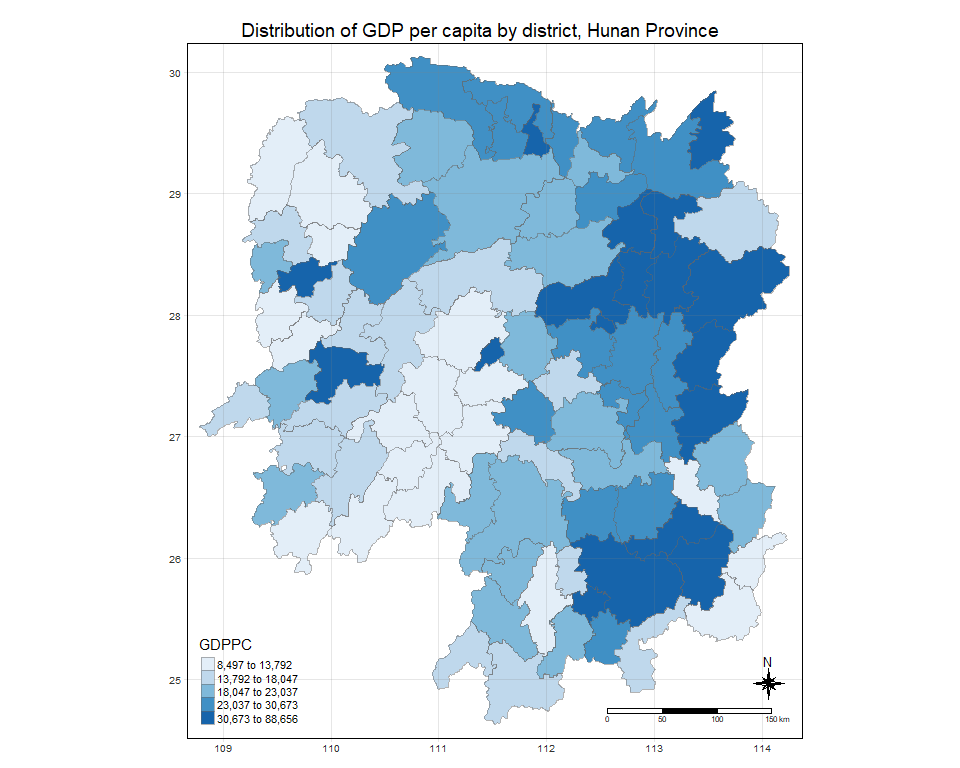
|  |
| --- |
| Important |
| In order to retain the geospatial properties, the left data frame must the sf data.frame (i.e. hunan) |

### Plotting a choropleth map

|  |
| --- |
| Do It Yourself! |
| Using the steps you learned in previous lesson, plot a choropleth map showing the distribution of GDPPC of Hunan Province. |

The choropleth should look similar to ther figure below.

tmap\_mode("plot")  
tm\_shape(hunan\_GDPPC) +  
 tm\_fill("GDPPC",   
 style = "quantile",   
 palette = "Blues",  
 title = "GDPPC") +  
 tm\_borders(alpha = 0.5) +  
 tm\_layout(main.title = "Distribution of GDP per capita by district, Hunan Province",  
 main.title.position = "center",  
 main.title.size = 1.2,  
 legend.height = 0.45,   
 legend.width = 0.35,  
 frame = TRUE) +  
 tm\_compass(type="8star", size = 2) +  
 tm\_scale\_bar() +  
 tm\_grid(alpha =0.2)



## Deriving Contiguity Spatial Weights

By and large, there are two types of spatial weights, they are contiguity wights and distance-based weights. In this section, you will learn how to derive contiguity spatial weights by using sfdep.

Two steps are required to derive a contiguity spatial weights, they are:

1. identifying contiguity neighbour list by [st\_contiguity()](https://sfdep.josiahparry.com/reference/st_contiguity.html) of **sfdep** package, and
2. deriving the contiguity spatial weights by using [st\_weights()](https://sfdep.josiahparry.com/reference/st_weights.html) of **sfdep** package

In this section, we will learn how to derive the contiguity neighbour list and contiguity spatial weights separately. Then, we will learn how to combine both steps into a single process.

### Identifying contiguity neighbours: Queen’s method

In the code chunk below [st\_contiguity()](https://sfdep.josiahparry.com/reference/st_contiguity.html) is used to derive a contiguity neighbour list by using Queen’s method.

nb\_queen <- hunan\_GDPPC %>%   
 mutate(nb = st\_contiguity(geometry),  
 .before = 1)

|  |
| --- |
| Important |
| By default, queen argument is **TRUE**. If you do not specify *queen = FALSE*, this function will return a list of first order neighbours by using the Queen criteria. Rooks method will be used to identify the first order neighbour if queen = FALSE is used. |

The code chunk below is used to print the summary of the first lag neighbour list (i.e. nb) .

summary(nb\_queen$nb)

Neighbour list object:  
Number of regions: 88   
Number of nonzero links: 448   
Percentage nonzero weights: 5.785124   
Average number of links: 5.090909   
Link number distribution:  
  
 1 2 3 4 5 6 7 8 9 11   
 2 2 12 16 24 14 11 4 2 1   
2 least connected regions:  
30 65 with 1 link  
1 most connected region:  
85 with 11 links

The summary report above shows that there are 88 area units in Hunan province. The most connected area unit has 11 neighbours. There are two are units with only one neighbour.

To view the content of the data table, you can either display the output data frame on RStudio data viewer or by printing out the first ten records by using the code chunk below.

nb\_queen

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  
First 10 features:  
 nb NAME\_2 ID\_3 NAME\_3 ENGTYPE\_3  
1 2, 3, 4, 57, 85 Changde 21098 Anxiang County  
2 1, 57, 58, 78, 85 Changde 21100 Hanshou County  
3 1, 4, 5, 85 Changde 21101 Jinshi County City  
4 1, 3, 5, 6 Changde 21102 Li County  
5 3, 4, 6, 85 Changde 21103 Linli County  
6 4, 5, 69, 75, 85 Changde 21104 Shimen County  
7 67, 71, 74, 84 Changsha 21109 Liuyang County City  
8 9, 46, 47, 56, 78, 80, 86 Changsha 21110 Ningxiang County  
9 8, 66, 68, 78, 84, 86 Changsha 21111 Wangcheng County  
10 16, 17, 19, 20, 22, 70, 72, 73 Chenzhou 21112 Anren County  
 County GDPPC geometry  
1 Anxiang 23667 POLYGON ((112.0625 29.75523...  
2 Hanshou 20981 POLYGON ((112.2288 29.11684...  
3 Jinshi 34592 POLYGON ((111.8927 29.6013,...  
4 Li 24473 POLYGON ((111.3731 29.94649...  
5 Linli 25554 POLYGON ((111.6324 29.76288...  
6 Shimen 27137 POLYGON ((110.8825 30.11675...  
7 Liuyang 63118 POLYGON ((113.9905 28.5682,...  
8 Ningxiang 62202 POLYGON ((112.7181 28.38299...  
9 Wangcheng 70666 POLYGON ((112.7914 28.52688...  
10 Anren 12761 POLYGON ((113.1757 26.82734...

The print shows that polygon 1 has five neighbours. They are polygons number 2, 3, 4, 57,and 85.

One of the advantage of **sfdep** over **spdep** is that the output is an sf tibble data frame.

|  |
| --- |
| Do It Yourself! |
| Using the steps you learned in previous lesson, display nb\_queen sf tibble data frame in a table display. |

kable(head(nb\_queen,  
 n=10))

| nb | NAME\_2 | ID\_3 | NAME\_3 | ENGTYPE\_3 | County | GDPPC | geometry |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2, 3, 4, 57, 85 | Changde | 21098 | Anxiang | County | Anxiang | 23667 | POLYGON ((112.0625 29.75523… |
| 1, 57, 58, 78, 85 | Changde | 21100 | Hanshou | County | Hanshou | 20981 | POLYGON ((112.2288 29.11684… |
| 1, 4, 5, 85 | Changde | 21101 | Jinshi | County City | Jinshi | 34592 | POLYGON ((111.8927 29.6013,… |
| 1, 3, 5, 6 | Changde | 21102 | Li | County | Li | 24473 | POLYGON ((111.3731 29.94649… |
| 3, 4, 6, 85 | Changde | 21103 | Linli | County | Linli | 25554 | POLYGON ((111.6324 29.76288… |
| 4, 5, 69, 75, 85 | Changde | 21104 | Shimen | County | Shimen | 27137 | POLYGON ((110.8825 30.11675… |
| 67, 71, 74, 84 | Changsha | 21109 | Liuyang | County City | Liuyang | 63118 | POLYGON ((113.9905 28.5682,… |
| 9, 46, 47, 56, 78, 80, 86 | Changsha | 21110 | Ningxiang | County | Ningxiang | 62202 | POLYGON ((112.7181 28.38299… |
| 8, 66, 68, 78, 84, 86 | Changsha | 21111 | Wangcheng | County | Wangcheng | 70666 | POLYGON ((112.7914 28.52688… |
| 16, 17, 19, 20, 22, 70, 72, 73 | Chenzhou | 21112 | Anren | County | Anren | 12761 | POLYGON ((113.1757 26.82734… |

### Identify contiguity neighbours: Rooks’ method

|  |
| --- |
| Do It Yourself! |
| Using the steps you just learned, derive a contiguity neighbour list using Rooks’ method. |

nb\_rook <- hunan\_GDPPC %>%   
 mutate(nb = st\_contiguity(geometry,  
 queen = FALSE),  
 .before = 1)

### Identifying higher order neighbors

There are times that we need to identify high order contiguity neighbours. To accomplish the task, [st\_nb\_lag\_cumul()](https://sfdep.josiahparry.com/reference/st_nb_lag_cumul.html) should be used as shown in the code chunk below.

|  |
| --- |
| Do It Yourself! |
| Using the steps you just learned, derive a contiguity neighbour list using lag 2 Queen’s method. |

nb2\_queen <- hunan\_GDPPC %>%   
 mutate(nb = st\_contiguity(geometry),  
 nb2 = st\_nb\_lag\_cumul(nb, 2),  
 .before = 1)

Note that if the order is 2, the result contains both 1st and 2nd order neighbors as shown on the print below.

nb2\_queen

Simple feature collection with 88 features and 8 fields  
Geometry type: POLYGON  
Dimension: XY  
Bounding box: xmin: 108.7831 ymin: 24.6342 xmax: 114.2544 ymax: 30.12812  
Geodetic CRS: WGS 84  
First 10 features:  
 nb  
1 2, 3, 4, 57, 85  
2 1, 57, 58, 78, 85  
3 1, 4, 5, 85  
4 1, 3, 5, 6  
5 3, 4, 6, 85  
6 4, 5, 69, 75, 85  
7 67, 71, 74, 84  
8 9, 46, 47, 56, 78, 80, 86  
9 8, 66, 68, 78, 84, 86  
10 16, 17, 19, 20, 22, 70, 72, 73  
 nb2  
1 2, 3, 4, 5, 6, 32, 56, 57, 58, 64, 69, 75, 76, 78, 85  
2 1, 3, 4, 5, 6, 8, 9, 32, 56, 57, 58, 64, 68, 69, 75, 76, 78, 85  
3 1, 2, 4, 5, 6, 32, 56, 57, 69, 75, 78, 85  
4 1, 2, 3, 5, 6, 57, 69, 75, 85  
5 1, 2, 3, 4, 6, 32, 56, 57, 69, 75, 78, 85  
6 1, 2, 3, 4, 5, 32, 53, 55, 56, 57, 69, 75, 78, 85  
7 9, 19, 66, 67, 71, 73, 74, 76, 84, 86  
8 2, 9, 19, 21, 31, 32, 34, 35, 36, 41, 45, 46, 47, 56, 58, 66, 68, 74, 78, 80, 84, 85, 86  
9 2, 7, 8, 19, 21, 35, 46, 47, 56, 58, 66, 67, 68, 74, 76, 78, 80, 84, 85, 86  
10 11, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 70, 71, 72, 73, 74, 82, 83, 86  
 NAME\_2 ID\_3 NAME\_3 ENGTYPE\_3 County GDPPC  
1 Changde 21098 Anxiang County Anxiang 23667  
2 Changde 21100 Hanshou County Hanshou 20981  
3 Changde 21101 Jinshi County City Jinshi 34592  
4 Changde 21102 Li County Li 24473  
5 Changde 21103 Linli County Linli 25554  
6 Changde 21104 Shimen County Shimen 27137  
7 Changsha 21109 Liuyang County City Liuyang 63118  
8 Changsha 21110 Ningxiang County Ningxiang 62202  
9 Changsha 21111 Wangcheng County Wangcheng 70666  
10 Chenzhou 21112 Anren County Anren 12761  
 geometry  
1 POLYGON ((112.0625 29.75523...  
2 POLYGON ((112.2288 29.11684...  
3 POLYGON ((111.8927 29.6013,...  
4 POLYGON ((111.3731 29.94649...  
5 POLYGON ((111.6324 29.76288...  
6 POLYGON ((110.8825 30.11675...  
7 POLYGON ((113.9905 28.5682,...  
8 POLYGON ((112.7181 28.38299...  
9 POLYGON ((112.7914 28.52688...  
10 POLYGON ((113.1757 26.82734...

## Deriving contiguity weights: Queen’s method

Now, you are ready to compute the contiguity weights by using [st\_weights()](https://sfdep.josiahparry.com/reference/st_weights.html) of **sfdep** package.

### Deriving contiguity weights: Queen’s method

In the code chunk below, queen method is used to derive the contiguity weights.

wm\_q <- hunan\_GDPPC %>%  
 mutate(nb = st\_contiguity(geometry),  
 wt = st\_weights(nb,  
 style = "W"),  
 .before = 1)

Notice that st\_weights() provides tree arguments, they are:

* *nb*: A neighbor list object as created by st\_neighbors().
* *style*: Default “W” for row standardized weights. This value can also be “B”, “C”, “U”, “minmax”, and “S”. B is the basic binary coding, W is row standardised (sums over all links to n), C is globally standardised (sums over all links to n), U is equal to C divided by the number of neighbours (sums over all links to unity), while S is the variance-stabilizing coding scheme proposed by Tiefelsdorf et al. 1999, p. 167-168 (sums over all links to n).
* *allow\_zero*: If TRUE, assigns zero as lagged value to zone without neighbors.

wm\_q

Simple feature collection with 88 features and 8 fields  
Geometry type: POLYGON  
Dimension: XY  
Bounding box: xmin: 108.7831 ymin: 24.6342 xmax: 114.2544 ymax: 30.12812  
Geodetic CRS: WGS 84  
First 10 features:  
 nb  
1 2, 3, 4, 57, 85  
2 1, 57, 58, 78, 85  
3 1, 4, 5, 85  
4 1, 3, 5, 6  
5 3, 4, 6, 85  
6 4, 5, 69, 75, 85  
7 67, 71, 74, 84  
8 9, 46, 47, 56, 78, 80, 86  
9 8, 66, 68, 78, 84, 86  
10 16, 17, 19, 20, 22, 70, 72, 73  
 wt  
1 0.2, 0.2, 0.2, 0.2, 0.2  
2 0.2, 0.2, 0.2, 0.2, 0.2  
3 0.25, 0.25, 0.25, 0.25  
4 0.25, 0.25, 0.25, 0.25  
5 0.25, 0.25, 0.25, 0.25  
6 0.2, 0.2, 0.2, 0.2, 0.2  
7 0.25, 0.25, 0.25, 0.25  
8 0.1428571, 0.1428571, 0.1428571, 0.1428571, 0.1428571, 0.1428571, 0.1428571  
9 0.1666667, 0.1666667, 0.1666667, 0.1666667, 0.1666667, 0.1666667  
10 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125  
 NAME\_2 ID\_3 NAME\_3 ENGTYPE\_3 County GDPPC  
1 Changde 21098 Anxiang County Anxiang 23667  
2 Changde 21100 Hanshou County Hanshou 20981  
3 Changde 21101 Jinshi County City Jinshi 34592  
4 Changde 21102 Li County Li 24473  
5 Changde 21103 Linli County Linli 25554  
6 Changde 21104 Shimen County Shimen 27137  
7 Changsha 21109 Liuyang County City Liuyang 63118  
8 Changsha 21110 Ningxiang County Ningxiang 62202  
9 Changsha 21111 Wangcheng County Wangcheng 70666  
10 Chenzhou 21112 Anren County Anren 12761  
 geometry  
1 POLYGON ((112.0625 29.75523...  
2 POLYGON ((112.2288 29.11684...  
3 POLYGON ((111.8927 29.6013,...  
4 POLYGON ((111.3731 29.94649...  
5 POLYGON ((111.6324 29.76288...  
6 POLYGON ((110.8825 30.11675...  
7 POLYGON ((113.9905 28.5682,...  
8 POLYGON ((112.7181 28.38299...  
9 POLYGON ((112.7914 28.52688...  
10 POLYGON ((113.1757 26.82734...

## Distance-based Weights

There are three popularly used distance-based spatial weights, they are:

* fixed distance weights,
* adaptive distance weights, and
* inverse distance weights (IDW).

### Deriving fixed distance weights

Before we can derive the fixed distance weights, we need to determine the upper limit for distance band by using the steps below:

geo <- sf::st\_geometry(hunan\_GDPPC)  
nb <- st\_knn(geo, longlat = TRUE)  
dists <- unlist(st\_nb\_dists(geo, nb))

|  |
| --- |
| Things to learn from the code chunk above |
| * [st\_nb\_dists()](https://sfdep.josiahparry.com/reference/st_nb_dists.html) of sfdep is used to calculate the nearest neighbour distance. The output is a list of distances for each observation’s neighbors list. * [unlist()](https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/unlist) of Base R is then used to return the output as a vector so that the summary statistics of the nearest neighbour distances can be derived. |

Now, we will go ahead to derive summary statistics of the nearest neighbour distances vector (i.e. dists) by usign the coced chunk below.

summary(dists)

Min. 1st Qu. Median Mean 3rd Qu. Max.   
 21.56 29.11 36.89 37.34 43.21 65.80

The summary statistics report above shows that the maximum nearest neighbour distance is 65.80km. By using a threshold value of 66km will ensure that each area will have at least one neighbour.

Now we will go ahead to compute the fixed distance weights by using the code chunk below.

wm\_fd <- hunan\_GDPPC %>%  
 mutate(nb = st\_dist\_band(geometry,  
 upper = 66),  
 wt = st\_weights(nb),  
 .before = 1)

|  |
| --- |
| Things to learn from the code chunk above |
| * [st\_dists\_band()](https://sfdep.josiahparry.com/reference/st_dist_band.html) of sfdep is used to identify neighbors based on a distance band (i.e. 66km). The output is a list of neighbours (i.e. nb). * [st\_weights()](https://sfdep.josiahparry.com/reference/st_weights.html) is then used to calculate polygon spatial weights of the nb list. Note that:   + the default style argument is set to “W” for row standardized weights, and   + the default allow\_zero is set to TRUE, assigns zero as lagged value to zone without neighbors. |

|  |
| --- |
| Do It Yourself |
| Using the steps you learned in previous section, examine the data frame of the fixed distance weights. |

### Deriving adaptive distance weights

In this section, you will derive an adaptive spatial weights by using the code chunk below.

wm\_ad <- hunan\_GDPPC %>%   
 mutate(nb = st\_knn(geometry,  
 k=8),  
 wt = st\_weights(nb),  
 .before = 1)

|  |
| --- |
| Things to learn from the code chunk above |
| * [st\_knn()](https://sfdep.josiahparry.com/reference/st_knn.html) of sfdep is used to identify neighbors based on k (i.e. k = 8 indicates the nearest eight neighbours). The output is a list of neighbours (i.e. nb). * [st\_weights()](https://sfdep.josiahparry.com/reference/st_weights.html) is then used to calculate polygon spatial weights of the nb list. Note that:   + the default style argument is set to “W” for row standardized weights, and   + the default allow\_zero is set to TRUE, assigns zero as lagged value to zone without neighbors. |

### Calculate inverse distance weights

In this section, you will derive an inverse distance weights by using the code chunk below.

wm\_idw <- hunan\_GDPPC %>%  
 mutate(nb = st\_contiguity(geometry),  
 wts = st\_inverse\_distance(nb, geometry,  
 scale = 1,  
 alpha = 1),  
 .before = 1)

|  |
| --- |
| Things to learn from the code chunk above |
| * [st\_contiguity()](https://sfdep.josiahparry.com/reference/st_contiguity.html) of sfdep is used to identify the neighbours by using contiguity criteria. The output is a list of neighbours (i.e. nb). * [st\_inverse\_distance()](https://sfdep.josiahparry.com/reference/st_inverse_distance.html) is then used to calculate inverse distance weights of neighbours on the nb list. |