Spatial Utilities

Rodrigo Rodrigues-SIlveira

November 11th, 2016

This code, spatial utilities, gathers some functions that are usefull for perfoming some basic spatial analysis tasks and are not entirely coded in R. Although the basic algorithms are included, many tasks are not automated. This code tries to solve this limitation.

It is composed by five functions: Points in Polygon, Voronoi diagrams, Spatial descriptives, Spatial lagged variable, Location Quotient.

Points in Polygon

Points in polygon (pointpoly) determines the number of events (points) that falls within a particular polygon. The function enable the use of a grouping variable to count events belonging to different categories.

Parameters:

sppoint - SpatialPointsDataFrame containing the events.

sppol - SpatialPolygonsDataFrame to be used as base for the aggregation of cases.

idvar - The name of the variable in sppol identifying each polygon.

idvar - The name of the variable in sppol identifying each polygon.

The first step is to load all the required packages and prepare the data to be used in the examples

```
# Loads the basic packages to be employed in the examples
library(sp)
library(maps)
library(maptools)
```

```
## Checking rgeos availability: TRUE
```

```
library(plotrix)

# Loads all the functions
source("Spatial_Utilities.R")


# Loads all the data for the examples and create the variables to be employed
e <- readShapeSpatial("Estados.shp")
a <- readShapeSpatial("Aerodromos.shp")
b <- readShapeSpatial("mapa_base.shp")
load("BR_Pres_vote.RData")

a$pavement <- "other"
a$pavement[a$PAVIMENTO=="terra"] <- "dirt runway"
a$pavement[a$PAVIMENTO%in%c("asfalto ou concreto Asfál","concreto")] <- "paved"

bl <- readShapeSpatial("Brazilian_Localities.shp")
bl <- bl[,c("GEOCODIG_M", "LONG","LAT")]
```

```r
uf <- c("SP","MG","RJ","BA","RS","PR","PE","CE","PA","MA","SC","GO","AM","PB",
        "ES","RN","AL","MT","PI","DF","MS","SE","RO","TO","AC","AP","RR")
pop <- c(44.8,21,16.6,15.3,11.3,11.2,9.4,9,8.3,7,6.9,6.7,4,4,4,3.5,3.4,3.3,3.2,
         3.2,2.7,2.3,1.8,1.5,0.8,0.8,0.5)

u <- data.frame(UF=uf, Population=pop)

e <- merge(e,u, by="UF")
```

Once the data is loaded and the function is in the memory, we can test the function, retrieving the number of aerodromes (including airports) (a) in each Brazilian State:

```r
# Retrieves the number of points in each polygon
# using polygons row.names for identification.
pointpoly(a,e)
```

```
## Loading required package: rgdal
```

```
## rgdal: version: 1.1-10, (SVN revision 622)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.0.1, released 2015/09/15
##  Path to GDAL shared files: C:/Users/rodrodr/Documents/R/win-library/3.3/rgdal/gdal
##  Loaded PROJ.4 runtime: Rel. 4.9.2, 08 September 2015, [PJ_VERSION: 492]
##  Path to PROJ.4 shared files: C:/Users/rodrodr/Documents/R/win-library/3.3/rgdal/proj
##  Linking to sp version: 1.2-3
```

```
## Loading required package: rgeos
```

```
## rgeos version: 0.3-19, (SVN revision 524)
##  GEOS runtime version: 3.5.0-CAPI-1.9.0 r4084
##  Linking to sp version: 1.2-3
##  Polygon checking: TRUE
```

```
##    id tot_points
## 1   0         29
## 2   1          7
## 3   2         55
## 4   3         82
## 5   4        179
## 6   5          9
## 7   6         47
## 8   7         61
## 9   8         19
## 10  9         19
## 11 10         13
## 12 11         15
## 13 12         24
## 14 13         15
## 15 14          5
## 16 15        141
## 17 16        206
## 18 17         11
## 19 18         28
## 20 19        295
## 21 20         99
## 22 21         30
```

```
## 23 22        108
## 24 23        455
## 25 24        531
## 26 25        152
## 27 26          3
```

In this first example, the IDs are the row.numbers of the SpatialPolygonsDataFrame. If we want to identify the States, we must define an ID variable (idvar=):

```
pointpoly(a,e, idvar="UF")
```

```
##    id tot_points
## 1  RO         29
## 2  AC          7
## 3  AM         55
## 4  RR         82
## 5  PA        179
## 6  AP          9
## 7  TO         47
## 8  MA         61
## 9  PI         19
## 10 CE         19
## 11 RN         13
## 12 PB         15
## 13 PE         24
## 14 AL         15
## 15 SE          5
## 16 BA        141
## 17 MG        206
## 18 ES         11
## 19 RJ         28
## 20 SP        295
## 21 PR         99
## 22 SC         30
## 23 RS        108
## 24 MS        455
## 25 MT        531
## 26 GO        152
## 27 DF          3
```

The last parameter allows to define different categories to aggregate points. In this example, we will use the type of pavement (dirt runway, paved, other):

```
pointpoly(a,e, idvar="UF", by = "pavement")
```

```
##    UF dirt run other paved
## 1  AC        0     2     5
## 2  AL        2     9     4
## 3  AM        5    13    37
## 4  AP        1     4     4
## 5  BA       27    49    65
## 6  CE        0     7    12
## 7  DF        0     2     1
## 8  ES        2     1     8
## 9  GO       49    65    38
## 10 MA        3    44    14
```

```
## 11 MG       52    67    87
## 12 MS      109   306    40
## 13 MT      125   364    42
## 14 PA       25   118    36
## 15 PB        1     3    11
## 16 PE        1     5    18
## 17 PI        0    12     7
## 18 PR       12    44    43
## 19 RJ        2     9    17
## 20 RN        0     7     6
## 21 RO        2    20     7
## 22 RR       47    32     3
## 23 RS        8    73    27
## 24 SC        3    12    15
## 25 SE        0     3     2
## 26 SP      116    83    96
## 27 TO        7    31     9
```

Voronoi Diagram

Voronoi diagrams define que area where the limits are closer to a particular event than any other. It divides a certain territory into subareas defined by the distance among points or events. This procedure makes possible to determine the area of influence in a given set of events. The potential applications are various and can include the area of coverage of any public infrastructure (Schools, hospitals, police stations, metro stations, bus stops), or the spread of a given disease.

One classical application was the classical cholera map made by John Snow in England in the 19th century. The hypothesis adopted by Snow considered that cholera was transmitted by water instead of air. In order to test this assumption, he plotted all water pumps in the Soho neighborhood in London and applied a series of tests and methods (including a dot density map) to verify the spatial clustering of deaths and their relation to the pumps. One of these methods was the computation of the Voronoi diagrams for each pump. Once he has done it, he aggregated the number of deaths falling within the area of influence of each pump, finding those more associated with the disease and discarding the ones that had no or few cases.

There are many algorithms for Voronoi diagrams in R, what is new about this one? While the algorithms create the diagram as a squared polygon, this function converts it into a new SpatialPolygonDataFrame and intersects them with a frame (country, city or other boundaries.). This helps to establish frontiers and use the new polygons in overlay operations (such as points in polygons, for instance). Therefore, it is an automation of a set of pre-existing functions, such as Points in Polygons. It helps to avoid recoding every step in order to obtain a framed result for a Voronoi diagram.

The parameters for the function are:

sppoint - SpatialPoints(DataFrame) object used as base for the calculation of the voronoi diagram.

sppoly - SpatialPolygons(DataFrame) object used as frame to define the limits of the voronoi.

ID - character vector for the identification of polygons.

plot - plot the results. The default is TRUE.

oper - one of geographical operations to be applied to the voronoi: “DIFF”, “INT”, “UNION”, or “XOR”, representing difference, intersection, union, and exclusive-or, respectively.

Let’s subset the data for obtaining the boundaries of the Sao Paulo state in Brazil and the airports belonging to this state.

```
es <- e[e$UF=="SP",]

as <- a[which(a$UF=="SP"),]

par(mar=rep(0,4))
plot(es)
plot(as, pch=19, add=T)
```

Once it is done, we can run the spvoro function to obtain the voronoi. It returns a SpatialPolygonsDataFrame object with the new boundaries computed from the points.

```
par(mar=rep(0,4))
vo <- spvoro(as,es,ID = "NOME")
```

```
## Loading required package: PBSmapping
```

```
## Warning: package 'PBSmapping' was built under R version 3.3.2
```

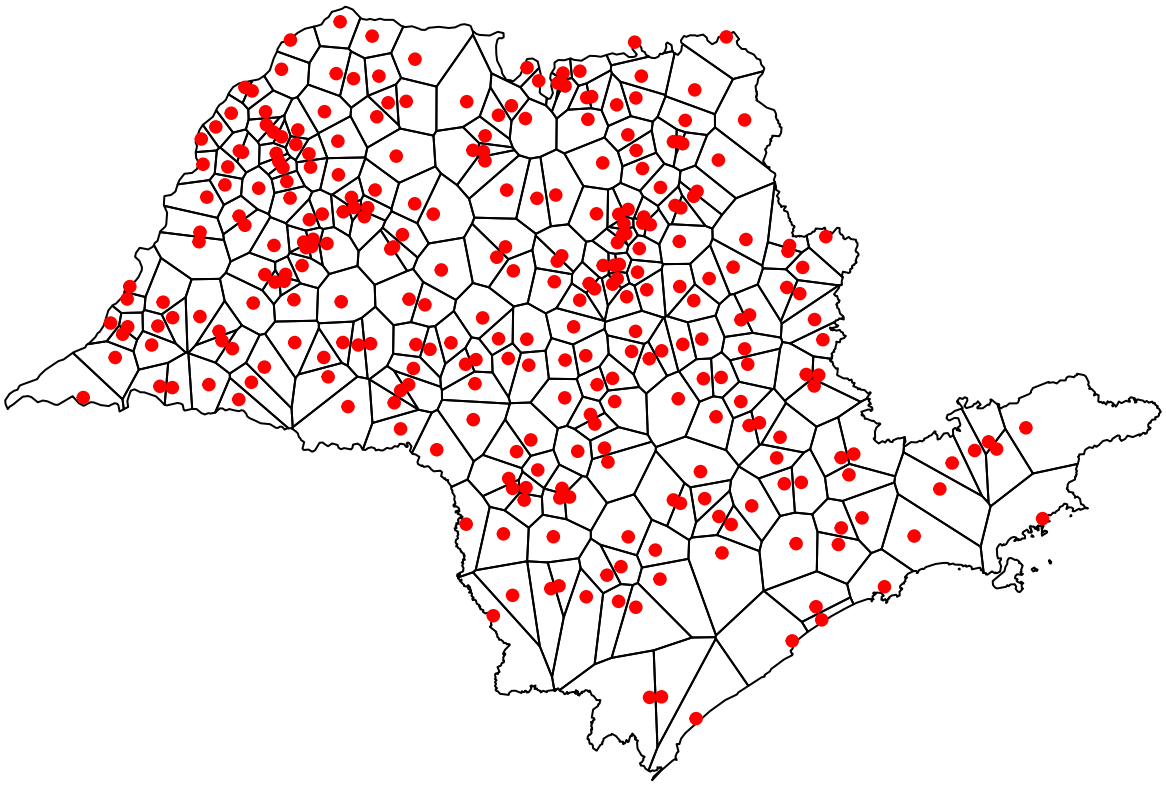
```
##
## ------------------------------------------------------------
## PBS Mapping 2.69.76 -- Copyright (C) 2003-2016 Fisheries and Oceans Canada
##
## PBS Mapping comes with ABSOLUTELY NO WARRANTY;
## for details see the file COPYING.
## This is free software, and you are welcome to redistribute
## it under certain conditions, as outlined in the above file.
##
## A complete user guide 'PBSmapping-UG.pdf' is located at
```

```
## C:/Users/rodrodr/Documents/R/win-library/3.3/PBSmapping/doc/PBSmapping-UG.pdf
##
## Packaged on 2015-04-23
## Pacific Biological Station, Nanaimo
##
## All available PBS packages can be found at
## http://code.google.com/p/pbs-software/
##
## To see demos, type '.PBSfigs()'.
## ------------------------------------------------------------

## Loading required package: deldir

## deldir 0.1-12

##
##      PLEASE NOTE:  The components "delsgs" and "summary" of the
##  object returned by deldir() are now DATA FRAMES rather than
##  matrices (as they were prior to release 0.0-18).
##  See help("deldir").
##
##      PLEASE NOTE: The process that deldir() uses for determining
##  duplicated points has changed from that used in version
##  0.0-9 of this package (and previously). See help("deldir").
```



As you can see, the function plotted the new map with the voronoi polygons with the boundaries of the Sao Paulo state and created a new spatial object called vo. Now we can just plot vo again to see what happens.

```
par(mar=rep(0,4))
plot(vo)
```

Now we are able to perform new analyses such as the number of people served by each aerodrome or the demographic density of each of them (since we can determine the area of these polygons). This logic can be applied from coffee shops to hospitals or political party committees.

Spatial Descriptive measures: mean center and standard distance

Another useful application for spatial analysis is to assess some basic descriptive values for the distribution of events. Where is the mean center of a given social, political or economic phenomenon? what is the level of dispersion or concentration of a given event? Before using more sophisticated techniques, such as kernel density estimators or kriging, it is useful to have some basic description of the location of the center and the size of variation to be dealt with. This can be particularly handy when we compare the same phenomenon in different points in time or according to different groups or strata.

In order to illustrate the functions, we will load the data and compute the mean center and the standard distance for all Brazilian Municipalities. Afterwards, we will plot the mean center and a circle representing the standard distance on the Brazilian States map.

The parameters of the spdesc function are the following:

** spobj -** the SpatialPointsDataFrame to be employed to compute the spatial descriptive statistics.

** w -** the name of the variable on the SpatialPointsDataFrame used to weight cases.

** by -** a vector containing the grouping information employed to separate cases.

```
## Merges electoral data (br) to the geographical locations of Brazilian municipalities (bl)
br <- br[order(br$year),]
br <- merge(br, bl, by="GEOCODIG_M", all.x=T)
br <- br[! is.na(br$LONG),]

## Converts the new object into a SpatialPointsDataFrame
coordinates(br) <- ~LONG+LAT

### Simple example: Brazilian localities

# Spatial descriptives for the Brazilian localities
bx <- spdesc(bl)

# See the results
bx
```

```
##              coordinates   sp_sd
## 1 (-46.22703, -16.44608) 10.4712
```

```
# plot the results
par(mar=rep(0,4))
plot(e)
plot(e[e$UF=="MG",], col="grey90", add=T)
plot(bx, pch=19, col="orange", add=T)

# Draws a circle with a radius corresponding to the standard distance (package plotrix)
draw.circle(coordinates(bx)[,1],coordinates(bx)[,2], bx$sp_sd)
```

This map represents the geographical center of Brazil and the circle the standard distance among municipalities. In itself it tells us little more about what we already know. These statistical procedures make the most when applied to particular phenomena, such electoral politics or demographic changes.

In the next example, we will employ the vote in presidential elections in Brazil from 1994 to 2010 to illustrate the usefulness of these techniques. We will select the candidates for the two major parties PT (Workers Party) and PSDB (Brazilian Social Democratic Party) and Marina Silva, a former PT militant that became the third contender in the last two presidential elections. Since cities vary enormously in size, it will be wise to weight the results by the number of votes received by each candidate. This would capture more precisely the effect of big cities in the voting pattern of certain candidates.

On the other hand, we do not want to know just the differences between parties, but we also are curious about how the geographical pattern of each party changed over time. Therefore, we define the year of election as a grouping variable. This will make the function return a mean center and a standard distance for each election. The code and the results are below:

```
# Separates the data for each major party: PT, PSDB and PSB
pt <- br[br$party==13,]
ps <- br[br$party==45,]
po <- br[br$party==43 & br$year==2010,]
po <- rbind(po, br[br$party==40 & br$year==2014,])


# Compute the spatial descriptives for each party
xt <- spdesc(pt, w = "qt_votes",by = pt$year)
xs <- spdesc(ps, w = "qt_votes",by = ps$year)
xo <- spdesc(po, w = "qt_votes",by = po$year)
```

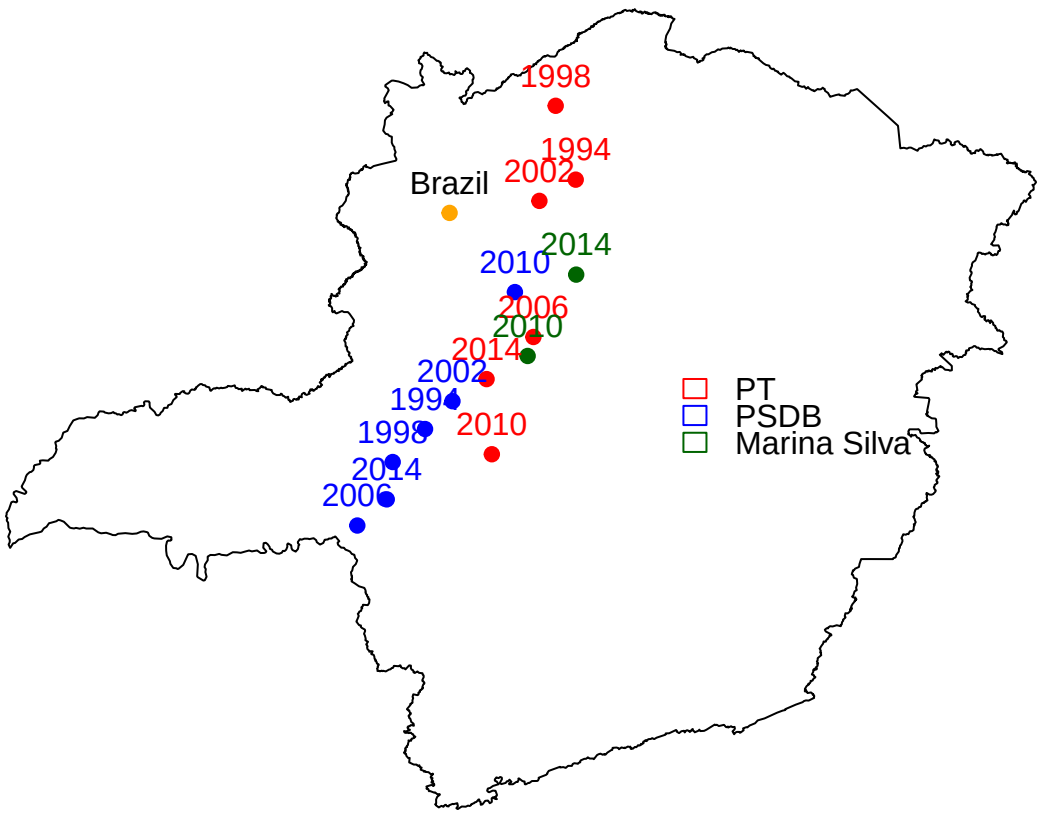
```r
# Plot the State of Minas Gerais (where all cases are located)
par(mar=rep(0,4))
plot(e[e$UF=="MG",])

# Adds the centers for each party and year in the map
plot(xt, pch=19, col="red", add=T)
plot(xs, pch=19, col="blue", add=T)
plot(xo, pch=19, col="darkgreen", add=T)

# Adds the years to identify each election
text(coordinates(xt), labels = xt$by, col = "red", pos=3)
text(coordinates(xs), labels = xs$by, col = "blue", pos=3)
text(coordinates(xo), labels = xo$by, col = "darkgreen", pos=3)

# Adds Brazilian mean center
plot(bx, pch=19, col="orange", add=T)
text(coordinates(bx), labels = "Brazil", pos = 3)

# Adds a legend to the map
legend(-44,-18, legend = c("PT","PSDB","Marina Silva"), fill=rep("white",3),
       border = c("red", "blue","darkgreen"), bty="n", y.intersp = 0.7)
```



The resulting map tells us a few things. Firstly, parties differ in terms of their geographical distribution of votes. They do so among themselves and in different elections. The PT vote is more concentrated in the North and Northeastern regions of the countries while the PSDB vote is centered on the Center-South region. Marina Silva performs better in the Southeastern metropolises and capture some of the vote of the PT in the

Northeast region. This happens especially in 2014, when she runs for the PSB, a party with deep roots in the Northeastern state of Pernambuco.

For both the PT and the PSDB, we can observe different trends of going north and south. This can be explained by different causes. One of them is the level of national appeal of presidential candidates. For instance, José Serra (PSDB, 2010) was a much more "national" candidate than his colleagues Geraldo Alckimin (PSDB, 2006) or Aécio Neves (PSDB, 2014), much more supported in the Center-South, but with little penetration in the North-Northeast.

The PT vote presents a trend to move south from 1998 on. This is a sign of the nationalization of the vote in the party and the increasing margins of victory obtained by the party until 2010. In 2014, the election was more competitive, with the south massively supporting Aécio Neves. This can be seen in the displacement of the PT center to the north.

This straightforward example shows us the usefulness of these summary statistics. They do not replace deeper and careful analyses, but they allow researchers and users to grasp general trends and changes in patterns that could lead to new hypotheses and explanations.

The data and the code is available at Github: https://github.com/rodrodr/Spatial_Analysis/