OS 4118: Spatial Exercise Fall AY 2020

**Question:**  Are parking enforcement actions in San Francisco spatially random?

In San Francisco you can dial “311” on your telephone to connect to a Customer Service Center and report an “event” like graffiti, dumping, broken traffic lights, and so on. From the website <https://sf.connect.socrata.com/#!/view-data>, click on the View link in the “Service Requests, 311 Cases” row. Scroll down to the “Table Preview” and click the blue “View Data” button. “Filter” the data so that the “Opened” field – the time at which the event was started – is between 3/8/2019 12:00:00 AM and 3/10/2019 12:00:00 AM. (When I did this there were around 1,300 rows). Export the data to your computer. I recommend CSV for Excel, but any format will be fine.

Take these steps:

(a) Read the data into R. When you read into R, set quote="\"" and comment="" so that you don’t misinterpret words like “O’Farrell Street” or “block #2.” (note: newer versions of the 311 data seem to have eliminated these issues). I also recommend stringsAsFactors=FALSE so you don’t create factors until you’re ready. Remove rows for which Latitude = 0.

(b) First let us address the hypothesis that says that open 311 “parking enforcement” reports are like a completely spatially random process. So, restrict your dataset to parking enforcement actions – that is, those with “Parking Enforcement” in Category. Now use the ppp() function from the spatstat library to create a “point pattern” from the parking actions. You will need to pass in x and y (the longitudes and latitudes of your events), plus a window that describes the possible range of x and y (since San Francisco is not a rectangle). You can construct this window with the owin() function using poly=; coordinates for the boundaries of the city are given in cells D4:E14 of the sf.polygon spreadsheet. (Notes: (ii) the left part of the spreadsheet won’t be needed. (ii) A small number of points may lie outside the city. Ignore them.)

Once your ppp object has been constructed, plot it. Then use envelope() to compute and plot the estimated *F* function for this data. Make the picture, especially the title, pretty. Do 311 “parking enforcement” reports appear to come from a completely spatially random process?

(c) Now let’s draw a prettier picture. Acquire my write.kml function from Sakai, under Resources | R Scripts, by using source() on the write.kml.txt file. Run it to create a KML output file.

Now let us start **Google Earth**. If you have this on your desktop, use File | Import or Open to read in the KML file. If not, go to <https://earth.google.com> using the Chrome browser, then “Launch Earth in Chrome.” From the main menu (three horizontal lines, top left), pick “My Places” and Import KML file. Either way, show me an attractive picture. You can construct this picture by screen capture. From the desktop, File | E-mail | Email Image will let you send yourself a JPG, but from the browser I don’t know how to get a picture. What does the picture tell you?

If your set-up is such that you cannot run either the desktop app or Chrome, let me know.

I’m hoping for the ppp plot (a bunch of dots on a city-shaped area), the picture of the *F* function, and a Google Earth-type photo-style picture of San Francisco, with a bunch of icons on it.

**Notes on write.kml():** This function takes these arguments:

df data frame to be written

long.name **name** of column with longitude in it – not the column itself, but a character string

lat.name **name** of column with latitude in it – not the column itself, but a character string

outfile file name for output. Default: "h:/output.kml"

icon.num Selects color of icon. 1 = red, 2 = blue, 3 = green, 4 = yellow, 5 = white. Default 1.

print.open Logical: print the “top of file” stuff? Default TRUE.

print.close Logical: print the “bottom of file” stuff? Default TRUE.

append Logical: open for append? Default: !print.open

The plan here is, if you’re running this only once, you want to create a file with the “top” and “bottom” KML stuff in it. That’s the default, and that’s what we do here. So, for example, if your data frame was called “parking,” you might use the command

write.kml (parking, "long", "lat", outfile = "c:/myfile.kml", icon.num = 1)

Notice that the columns are identified by the **name** of the column within the data set.