02.221 – Lab 9: Spatial Analysis in R

The data needed for this lab can be found on the course website or downloaded from Dropbox: https://www.dropbox.com/s/zybl27zn6mqbske/02221-Lab9.zip?dl=0. Extract the data from the zip file to an appropriate location within your documents.

Goals

The primary goal of this lab to learn how various spatial analyses can be performed within R. We will do so through an exploration of the density and various attributes of hawker centres in Singapore.

Data Manipulation

Imagine you are interested in the spatial distribution and various characteristics of the many hawker centres around Singapore. Unfortunately, there is very few readily available data available on this topic. You did find a list of all successful tenders since early 2012 on the website of the National Environment Agency (http://www.nea.gov.sg/public-health/hawker-centres/tender-notice). However, it is not immediately usable as it's published as PDF and not as csv or xls file. This is a surprisingly common occurrence. To save you some time, you have tasked your intern to extract the data from the pdf into table format using the Tabula tool (http://tabula.technology/).[1]

Start a new project in RStudio and install the following packages that you will need in this lab: data.table, sp, rgdal, spatstat, ggmap, maptools, ggplot2. Subsequently, make sure to load them all using library(). In this lab, we are going to use RMarkdown to document our progress. You can read more about RMarkdown at rmarkdown.rstudio.com. I recommend reading the Quick Tour and keeping the cheatsheet handy for future reference. In principle, it is a convenient way to combine R code, plots and explanatory text all in one single document. You can open a new Markdown document by going to File | New File | R Markdown.

We start by reading our raw csv data into R.

```
tenders <- fread("tabula-tender-bids-from-mar-
2012-to-jan-2017.csv", header = F, fill = T)
colnames(tenders) <- c("centre", "stall", "area",
"trade", "bid", "month")
```

As you inspect the table, you will notice a number of messy aspects of the data that will not allow you to do any analysis for now. We need to clean this up a bit more. First, we get rid of empty rows. Our tenders object is a data.table, this is a kind of superset of the data.frame class that allows for faster more efficient manipulation. For

[1] You are welcome to try this for yourself but you will see an extracted csv file is included in this lab's data.

example, in this case we don't need to specify tenders$centre but can just use only the column name:

```
tenders <- tenders[centre != "",]
```

There are two types of stall in the data. Those for cooked food and 'lock-up' stalls. At row 1135 is the split between them. First we need to get rid of that particular row and subsequently set anything before it to type=cooked and anything after it to type=lockup. We use the data.table assignment operator ':=' here to assign these values to a subset of the rows in our table:

```
tenders <- tenders[!1135,]
tenders[1:1134,type:="cooked",]
tenders[1135:nrow(tenders),type:="lockup",]
```

Finally, it makes sense to do some formatting to get the columns in the right class for subsequent analysis and visualization. We need to remove all occurences of '$' and ',' from the bid price before casting it as a number; the date is currently a string that we can interpret as a real date instead; and, since we both now price and area, we can calculate the price per square meter as well.

```
tenders[,bidNum:=as.numeric(gsub(bid,
pattern="\\$|,", replacement = "")),]
tenders[,date:=as.Date(paste0("01-",month), "%d-
%b-%Y"),]
tenders[,priceM2:=bidNum/as.numeric(area),]
```

We can include all the previous steps in our Markdown document so it becomes clear to anybody else what data cleaning steps were executed:

```
```{r}
library(data.table)
tenders <- fread("tabula-list-of-successful-tenderers-from-march-2012.csv", header = F)
colnames(tenders) <- c("centre", "stall", "area", "trade", "bid", "month") #table doesn't have headers
tenders <- tenders[centre != "",] #remove empty rows
tenders <- tenders[!822,] #remove 'lockup' row
tenders[1:821,type:="cooked",] # set type to cooked for anything pre-lockup
tenders[822:nrow(tenders),type:="lockup",] # and vice-versa
tenders[,bidNum:=as.numeric(gsub(bid, pattern="\\$|,", replacement = "")),] #convert formatted price to numeric
tenders[,date:=as.Date(paste0("01-",month), "%d-%b-%Y"),] #same for date
tenders[,priceM2:=bidNum/as.numeric(area),] # price per m2
head(tenders) # everything ok?
```
```

You can subsequently 'Knit HTML' to render the results. What this will do is start a new R session, run all the commands and render both the output of your code as well as any text you have typed.

Now that all the data is in the right format, it would be good to explore some initial questions through data visualization. For example:
- Is there a relation between area and price?
- What is the average price per m2 per type and trade?

- What is the average price per m2 per hawker centre?
- How many bids are there for each hawker centre?
- Does the number of bids change over time?
- Does the average price per m2 change over time?

Here we can use data.table's quick manipulation techniques again to slice and dice the data in different ways. For example:

```
tenders[,list(price=mean(priceM2)),by=centre] #avg price per centre
tenders[type==”cooked”,list(price=mean(priceM2)),by=centre] # avg price per centre for only cooked food
tenders[,list(count=length(stalls)),by=centre] # number of bids per centre
```

Choose 2-3 questions from the list above or your own additions and include visualizations that answer those questions in your Markdown document. The plots will render within the html document automatically!

```
```{r}
library(ggplot2)
ggplot(tenders[,list(count=length(stall)),by=date], aes(date, count)) + geom_line()
```
```

Spatial Data

Of course, each hawker centre has a specific location but this is not explicitly included in the dataset. To plot things spatially, we need to find a spatial dataset that we can join to. Luckily, you can find a KML file of all hawker centres on data.gov.sg. Download the file, extract it to your project directory and read it in:

```
h <- readOGR("hawker-centres.kml", "HAWKERCENTRE")
plot(h)
```

Perfect! Now all we need to do is extract the lat/lon for each centre and join those to our tenders data.frame. In R, joining is done through the merge() function, like so:

```
h.t <- data.frame(toupper(h$Name), h@coords[,1:2])
colnames(h.t) <- c("name", "lon", "lat")
tenders.sp <- merge(tenders, h.t, by.x="centre", by.y="name", all.x = T)
```

Do you understand what each of these steps does? Review the results of your joining process. You will quickly notice many of the hawker centres weren't joined properly. This is due to varying ways of spelling as well as alternate names for the same centre.

All in all, the names in the KML file do not correspond all that well with the names in the tender data. Another solution would be to manually geocode each hawker centre. As there are only 100-something centres in Singapore, this shouldn't be too troublesome. First we create a list of unique centre names, then we add "Singapore" to the name of each centre to aid the geocoding service, and finally we use the geocode() function from the ggmap package to geocode using Google's service.

```
centres <- tenders[,list(count=.N),by=centre]
centres[,location:=paste0(centre, ", Singapore"),]
g <- geocode(centres[,location,], output =
"latlon", source = "google", sensor = F)
centres <- cbind(centres, g)
tenders.sp <- merge(tenders, centres,
by.x="centre", by.y="centre", all.x = T)
```

Try to understand each step and view the results. We can double-check our results by doing a quick plot with ggplot:

```
ggplot(tenders.sp, aes(x=lon, y=lat, size=priceM2,
color=type)) + geom_point(alpha=0.3) +
coord_fixed()
```

Note that we have lots of overplotting since, of course, many stalls are in the same hawker centre. We could try and do a density plot instead:

```
ggplot(tenders.sp, aes(x=lon, y=lat)) +
geom_point() + geom_density2d() + coord_fixed()
ggplot(tenders.sp, aes(x=lon, y=lat)) +
geom_point() + geom_hex() + coord_fixed()
```

Do you think there is a specific spatial distribution with regard to the number of bids per 'trade' or 'type'? Or different patterns based on the year? Create a (series of) faceted plot(s) to help you answer that question and include them (and the code leading up to it) in your markdown document.

Spatial Point Patterns

Ggplot allows us to visualize spatial patterns but it does not allow us to analyze those patterns. For example, we cannot tell whether the clustering we see in some spatial data might be the result of pure chance. To do that, we can use the spatstat package, which allows for the *analysis* of spatial point patterns.

First we need to get the data in the right format. We create a table with one row for each centre, including the location but also the average price and the number of bids. We then tell R which columns in our data table are spatial coordinates and finally convert our spatial dataset to a format that spatstat can understand ('ppp').

```
centres.sp <- tenders.sp[lat > 0,list(lon=lon[1],
lat=lat[1], price=mean(priceM2),
count=.N),by=centre]
centres.sp[is.na(price),price:=0,] #NA to 0
coordinates(centres.sp) <- c('lon', 'lat')
centres.ppp <- unmark(as.ppp(centres.sp))
```

Try plotting the resulting object to see if the procedure worked. You will see that it uses the bounding box of all points as the plotting extent. This is good for visualizations but for analysis we want to set the extent to our total study area instead. After all, if in some corner of Singapore there aren't any hawker centres that could be relevant but by using the bounding box only, that part of the country would be 'chopped' off. We read in a shapefile that contains the outline of the country and define it as the 'window' for our spatial point pattern.

```
sg <- readOGR(".", "sg-all")
sg.window <- as.owin(sg)
centres.ppp <- centres.ppp[sg.window]
```

Plot the centres.ppp object again and notice the difference. Now that we have a proper window defined, let's look at the clustering of hawker centres[2].

```
plot(Kest(centres.ppp))
```

We can also create density plots or contour maps using the same data. Try to experiment with the smoothing/sigma parameter to understand its effect.

```
plot(density(centres.ppp, 0.02))
contour(density(centres.ppp, 0.02))
```

Of course, we find that hawker centres are clustered together in space. We would expect that already based on our experience living in Singapore. You might wonder if the clustering of hawker centres is just a function of the underlying population in that area. We can now easily test that assumption. First we load a raster file with population in Singapore[3], and then estimate the intensity of the hawker center point pattern as a function of the population (covariate).

```
pop <- as.im(readGDAL("sg-pop.tif"))
plot(rhohat(centres.ppp, pop))
```

We can also see what the effect is, if we weigh each centre by its average tender price:

[2] N.B. Wait until the April 3 class for an explanation of this graph
[3] Derived from the gridded population dataset we worked with in Lab 6

```
plot(rhohat(centres.ppp, pop,
weights=centres.sp$price))
```

You notice there are outliers in areas with a high population. We would have expected a much higher intensity of hawker centers there based on the rest of the pattern. Try to find out where these outliers are located and reflect on what makes these centres different and why we could expect a 'dip' towards the upper end of the population density.

```
plot(pop)
plot(centres.ppp, add=T)
```

Include all your plots and your reflection for this part of the assignment in your markdown document. When you are finished, 'knit' the markdown to html and publish the html to rpubs.

Assignment

On the class website, you will find the assignment for this lab. It just consists of your published rpubs document so you can submit the url of your document as your assignment. Please make sure you submit the assignment by **April 5.**