**OBJECTIVE.** Learn how to make flow maps in the R-environment. Total: 6pts. Due Friday October 27th 2017

Exercise 8: Flow mapping in R

Part 1: Simple Mapping with Lines (2pts)

In some cases it could be useful to connect the dots on your map if the order of the points have any relevance. With online location services such as Twitter or Facebook, Foursquare, etc… growing in popularity, location traces are not that rare. An easy way to draw lines is, well, with the **lines()** function. To demonstrate, look at fake locations traveled during a fake, seven days and nights trip.

Step 1:  
Start with loading the data (as usual) and drawing a base world map.

faketrace < read.csv("http://book.flowingdata.com/ch08/points/faketrace.txt", sep="\t")

map(database="world", col="#cccccc")

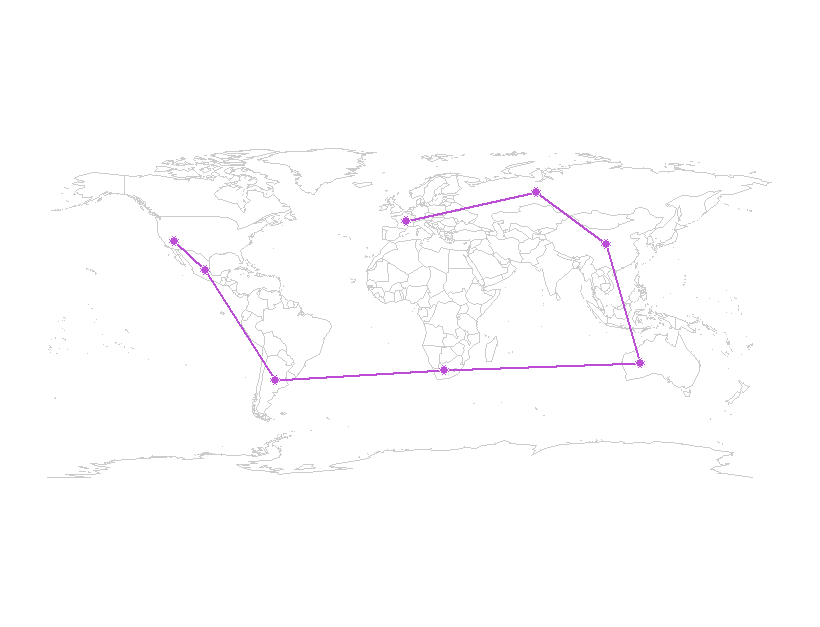
Step 2: Take a look at the data frame by entering *faketrace* in your R console. You see that it is just two columns for latitude and longitude and eight data points. You can assume that the points are already in the order that I traveled during those long seven nights

|  |  |  |  |
| --- | --- | --- | --- |
|  | latitude | longitude | |
| 1 | 46.31658 | 3.515625 | |
| 2 | 61.27023 | 69.60938 | |
| 3 | 34.30714 | 105.4688 | |
| 4 | -26.116 | 122.6953 | |
| 5 | -30.1451 | 22.85156 | |
| 6 | -35.1738 | -63.6328 | |
| 7 | 21.28937 | -99.4922 | |
| 8 | 36.17336 | 115.181 |
|  |  |  | |

Step3: Draw the lines by simply plugging in the two columns into lines(). Also specify color (col) and line width (lwd).   
lines(faketrace$longitude, faketrace$latitude, col="#bb4cd4", lwd=2)

Step 4: Now also add dots

symbols(faketrace$longitude, faketrace$latitude, lwd=1, bg="#bb4cd4", fg="#ffffff", circles=rep(1, length(faketrace$longitude)), inches=0.05, add=TRUE)



Step 5: You can also make connections in all the countries visited. It could be interesting to draw lines from the last location to all the others.

map(database="world", col="#cccccc")

for (i in 2:length(faketrace$longitude)-1) {

lngs <-

c(faketrace$longitude[8], faketrace$longitude[i])

lats <-

c(faketrace$latitude[8], faketrace$latitude[i])

lines(lngs, lats, col="#bb4cd4", lwd=2)}



After you create the base map, you can loop through each point and draw a line from the last point in the data frame to every other location. This isn’t incredibly informative, but maybe you can find a good use for it. The point here is that you can draw a map and then use R’s other graphics functions to draw whatever you want using latitude and longitude coordinates.

Turn in: Include the figure that shows the path of the made-up locations (use another color, say green or red).

Part 2: Simple mapping with Line with your own data (2pts).

Repeat Part I of this exercise, this time with your own dataset (could be for the GPS dataset for instance). If the data that you use is located in Charlotte, you may need to change the second line of code to:

map(database="Charlotte", col="#cccccc")

Turn in: Include the figure that shows the path of the locations of your own dataset.

Part 3: Mapping commuting to work in 2011 in England[[1]](#footnote-1) (2pts).

You will use data from the Office for National Statistics flow data. The file we will use is called wu03ew\_v1.csv (>109 mb uncompressed). You also will use middle layer super output areas (MSOA) codes and their co-ordinates. The code is available here. Make sure you save your code in the directory where the data is located, and make sure you have patience. Lots of data to map.

Step 1: Install and load these libraries:  
library(plyr)  
library(ggplot2)  
library(maptools)

Step 2: Load the flow data required – origin and destination points are needed.   
input<-read.table("~/wu03ew\_v1.csv", sep=",", header=T)

Step 3: We only need the first 3 columns of the above (origin, destination, and the flow)  
input<- input[,1:3]  
names(input)<- c("origin", "destination","total")

Step 4: Joining coordinates to the census information  
centroids<- read.csv("~/msoa\_popweightedcentroids.csv")

#Joining to get the xy coordinates joined to the origin and then the destination points.

or.xy<- merge(input, centroids, by.x="origin", by.y="Code")

names(or.xy)<- c("origin", "destination", "trips", "o\_name", "oX", "oY")

dest.xy<- merge(or.xy, centroids, by.x="destination", by.y="Code")

names(dest.xy)<- c("origin", "destination", "trips", "o\_name", "oX", "oY","d\_name", "dX", "dY")

Step 5: Plotting with ggplot2.  
#This first step removes the axes in the resulting plot.

xquiet<- scale\_x\_continuous("", breaks=NULL)

yquiet<-scale\_y\_continuous("", breaks=NULL)

quiet<-list(xquiet, yquiet)

Step 6: Now for the plot. First specify the dataframe we need, with a filter excluding flows of <10

ggplot(dest.xy[which(dest.xy$trips>10),], aes(oX, oY))+

#The next line tells ggplot that we wish to plot line segments. The "alpha=" is line transparency and used below

geom\_segment(aes(x=oX, y=oY,xend=dX, yend=dY, alpha=trips), col="white")+

#Here is the magic bit that sets line transparency - essential to make the plot readable

scale\_alpha\_continuous(range = c(0.03, 0.3))+

#Set black background, ditch axes and fix aspect ratio

theme(panel.background = element\_rect(fill='black',colour='black'))+quiet+coord\_equal()

Step 7: Save your map (or take a screenshot). Try changing the lines to black and the background to white. Discuss in 5 lines max the pattern that you see



1. http://spatial.ly/2015/03/mapping-flows/ [↑](#footnote-ref-1)