GIS Assignment 1:

When approaching the task of creating a GIS map, it is fundamental to understand the type of data you are showing, and thus which programme would be most beneficial to use.

One of the main benefits of GUI programmes such as ArcGIS, is the graphic interface. This makes it possible to visualise ones’ data, and see instantly how each layer relates to the other. Certain errors, such as layers holding different geocode systems, are relatively simple to find, however small numerical discrepancies may go unnoticed. Furthermore, tools such as *select by attribute* and *select by location* enable easy extraction of data points, which once overlaid, may show unexpected trends. GUI programmes usually use colours to show variations in value or variables, however they are generally limited to 2 or 3 variables, and it may be difficult to find colour schemes that are ‘colour-blind friendly’. The final exported maps are also lower in quality than those produced by their counter part – command line programmes.

Command line programs, in this case R, are very intuitive due to their coding based system. Once knowledge is acquired around how spatial data files are stored, the different formats datasets are found in, and how to convert such files into compatible objects, data may be manipulated very freely. There are currently large libraries written and shared for command line programmes, allowing simple access to complex projections. Unlike GUI, all errors are reported instantly, thus saving time trying to figure out the issue. The error notification may also be expanded with a full description of the issue. However, if a project consists of multiple data sets, and one changes their directory location, issues may arise due to the mismatch in paths. Command line programmes are also able to process complex geographies in raster formats.

The following two maps are compiled mainly from governmental datasets, hence, one would expect a certain level of accuracy. Nonetheless, discrepancies are abundant. Certain collections of data are not updated frequently (for example the UK census is only produced once per decade), nor are they stored under the same parameters, units or written formats. Converting each dataset to fit the given format, can be a tedious task; one that may also highlight incorrect relationships.

The level of cartographic good practice shown by the GUI bivariate choropleth map, is of a decent standard. The use of a dual colour scheme is generally a good way to show how two variables relate to one another, yet the intensities of colours depicting the separate classes may be enhanced further for clarity. A bivariate tool may also be used in this instance, to create a fully integrated dual colour scheme. Moreover, the key has relatively vague values. The two layers shown to the bottom right of the map augment the individual spatial characteristics of the two variables, adding value to the relationships seen in the bivariate map.

The location and intensity map produced in R, (*intensity to be depicted through the size/area of the points – to be completed…),* is not so cartographically sound. Firstly, there lacks a heading and key of what is being shown. The *mapview()* tool has also coloured each point differently leading to confusion if there is a class difference between the points. However, the scale is shown, and the points are proportionally spaced within the map.

**Workflows for GUI and command line:**

GUI:

-London Borough shapefile downloaded – for basemap

-Traffic flow data .xls downloaded- sourced from London data store

*Data cleaned for appropriate values*

-Imported both data sets and joined through the local authority codes

*Checked for same geocode (BNG) – as this could cause an issue with*

*uncertainties*

-Exported the new joined dataset onto a new layer

-Run bivariate tool through the properties function to create a choropleth map

*Values/colour schemes/classes adjusted to show best representation of*

*Data*

Command Line:

*#importing libraries*

library(geojsonio)

library(tmap)

library(tmaptools)

library(sf)

library(gdal)

library(sp)

library(sf)

library(mapview)

*#importing dataset*

library(readxl)

PM\_longlat <- read\_excel("~/Downloads/PM\_longlat.xlsx")

View(PM\_longlat)

class(PM\_longlat)

*#converting to sf object*

*#indicating long and lat to be used as coords*

*#confirming crs for WGS84...*

PM\_longlat\_sf <- st\_as\_sf(PM\_longlat, coords = c("longitude", "latitude"), crs=4326)

mapview(PM\_longlat\_sf)

plot(PM\_longlat\_sf)

*#saving data ....*

write.csv(PM\_levels\_sf, "~/Downloads/aqipointsnyc.csv")

