Practical 1: Geoprocessing for Population Data Analysis

Edith Darin, Alina Game, Michael Harper

19 December, 2019

Getting Started

This document provides an example of how we can use geoprocessing techniques to analyse population datasets. The aims of this exercise include:

1. Demonstrate how we can load datasets within R
2. Provide some quick visualisation examples to gain an understanding of the dataset
3. Demonstrate commonly used geospatial techniques for analysing population datasets.

These methods are designed to provide an introduction to the second workshop exercise, which will build on these techniques and demonstrate how we could do a larger analysis around this work.

Setup

There are two main packages used for spatial data analysis:

1. **sp**: this is the more established R spatial data analysis package.
2. **sf**: this is the more recent implementation, and generally has easier to use syntax and clearer functions, along with performance improvements.

Although **sf** is generally seen as the more modern way of doing things, it is still partially in development and therefore there can be some operations which should be done is **sp**, particularly when dealing with raster datasets.

```
# Load core spatial packages
library(sp)
library(raster)
library(rgdal)
library(sf)
library(tmap)       # Easy way of producing maps
library(spatialEco)
```

Load data

3 datasets are loaded into the analysis, representing typical spatial datasets. These datasets have been accessed through the GRID3 data portal.

- **Population Data**: A raster with population data at 100m resolution for Nigeria
- **Roads**: Primary roads covering the whole of Nigeria
- **Roads**: a cropped dataset of road locations

- **Lagos State Admin 1 Boundaries**

Calculating Population Summaries

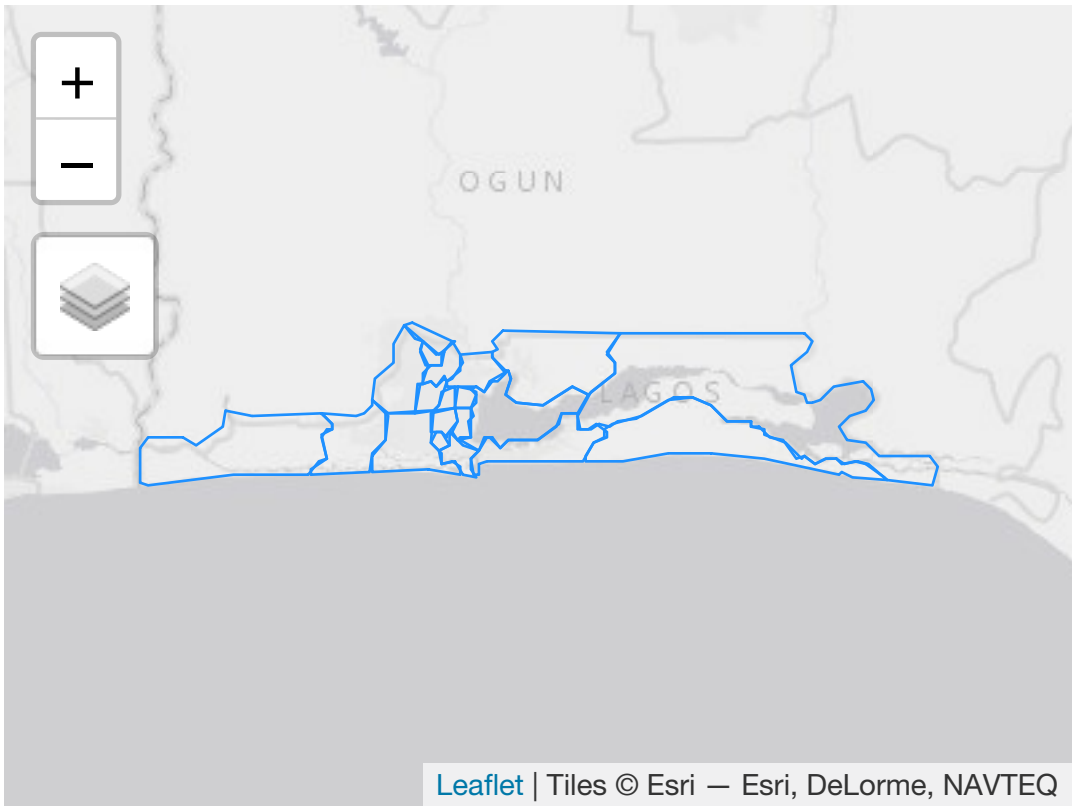
Two summaries are calculated:

1. One for existing boundaries
2. One for a buffer of a spatial line

Existing Polygon

We will use the existing Lagos admin 1 boundaries to calculate population estimates. These are displayed below:

```
tmap_mode("view") # Create a static map

tm_shape(lagos_sp) +
  tm_polygons(alpha = 0, border.col = "dodgerblue")
```



We can use the 'zonal.stats' function to calculate a summary of the population for each region in the shapefile. These calculations are merged to the original dataset:

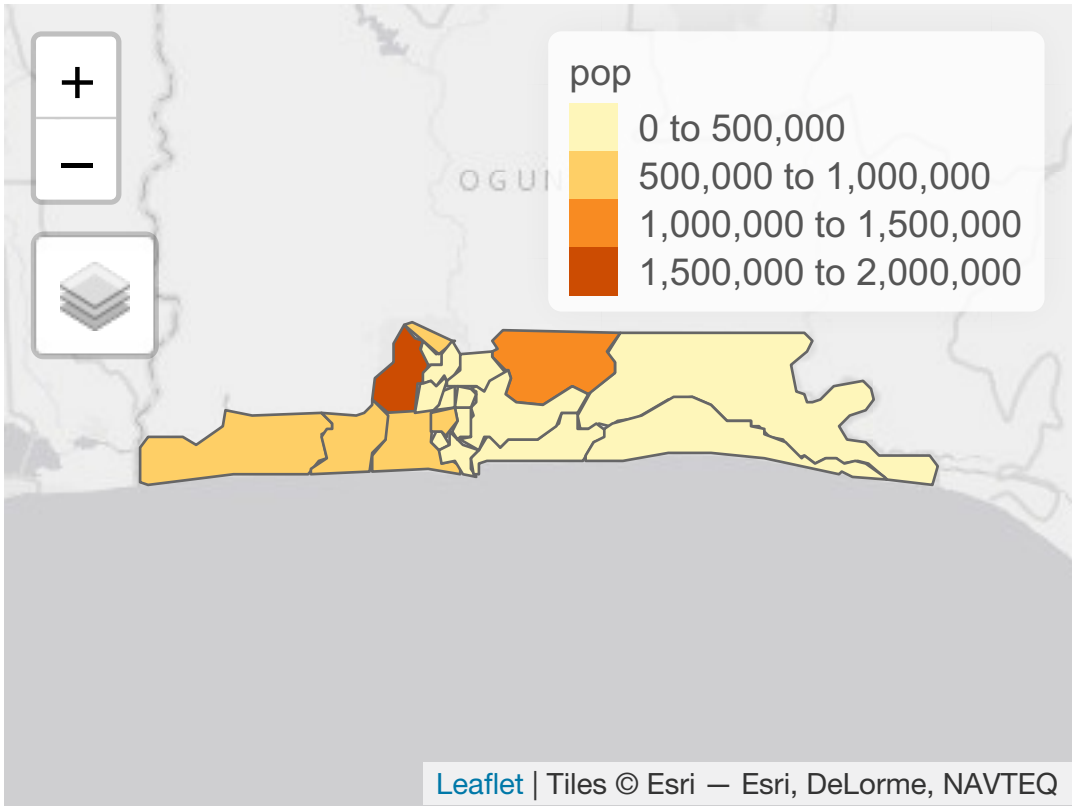
```
lagos_sp$pop <- zonal.stats(x = lagos_sp, population, stat = sum, plot = FALSE)
```

```
## processing 1 of 20
## processing 2 of 20
## processing 3 of 20
```

```
## processing 4 of 20
## processing 5 of 20
## processing 6 of 20
## processing 7 of 20
## processing 8 of 20
## processing 9 of 20
## processing 10 of 20
## processing 11 of 20
## processing 12 of 20
## processing 13 of 20
## processing 14 of 20
## processing 15 of 20
## processing 16 of 20
## processing 17 of 20
## processing 18 of 20
## processing 19 of 20
## processing 20 of 20
```

We can visualise the final population raster below:

```
tm_shape(lagos_sp) +
  tm_polygons("pop")
```



Lines or polygons

If we have a spatial feature which does not have an associated area with it, we will normally wish to calculate an area around this feature to work out the total number of people living wihthin the area

surrounding the object.

We can apply buffers around objects, and can be applied to **points**, **lines** and **polygons**. This is useful to know which areas fall within a certain distance from an attribute. Note, we must convert the geographic coordinate system into a planar coordinate system to enable us to calculate distance. The topic of coordinate systems is not covered fully in this training. We calculate a buffer below using the `st_buffer` function, calculating a buffer of 5000 metres. 'st_union' is used to merge the overlapping shapefiles into a single shapefile.

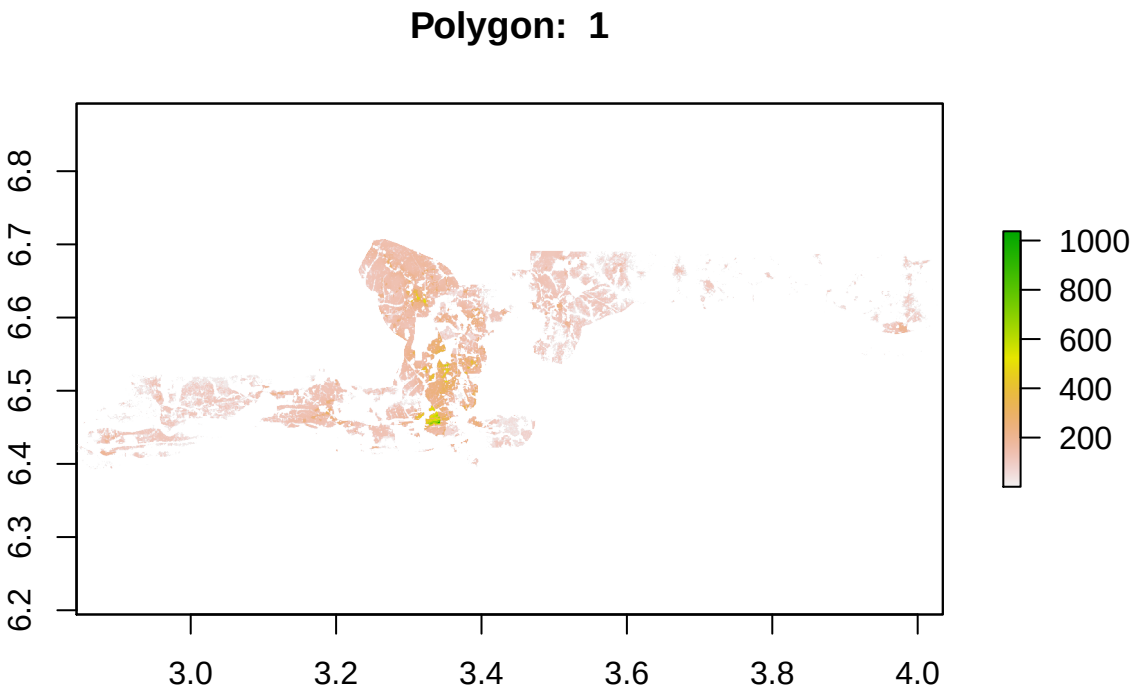
```
# Convert the geographic coordinate system to planar so that we can do distance based calculations
roadsLagos_planar <- st_transform(roadsLagos, crs = 26393)

# Calculate buffer
roads_buffer <- sf::st_buffer(roadsLagos_planar, 5000) %>%
  st_union()

# Convert back to geographic coordinate system
roads_buffer <- st_transform(roads_buffer, crs = 4326)
```

```
# Plot Data
map_roads <-
  tm_shape(roadsLagos) +
  tm_lines() +
  tm_layout(title = "Road")

map_buffer <-
  tm_shape(roads_buffer) +
  tm_polygons() +
  tm_shape(roadsLagos) +
  tm_lines() +
  tm_layout(title = "Buffer")

# Plot results
tmap_arrange(map_roads, map_buffer, ncol = 2, sync = TRUE)
```

As before, we can use the `zonal.stats` function to calculate the sum of the population within a region.

```
roads_buffer_sp <- as(roads_buffer, "Spatial")
roads_buffer_sp <- SpatialPolygonsDataFrame(Sr = roads_buffer_sp, data = data.frame(ID = 1), match.I

spatialEco::zonal.stats(roads_buffer_sp, population, stat = sum)
```

```
## processing 1 of 1
```




```
## [1] 8044475
```

Appendix: Geoprocessing Techniques

In the previous exercise, we presented some cleaned data. These scripts here include some of the extra steps taken to get the data in the correct format. Such data cleaning and preparation is often an important part of the geospatial data analysis workflow.

Clip vector data

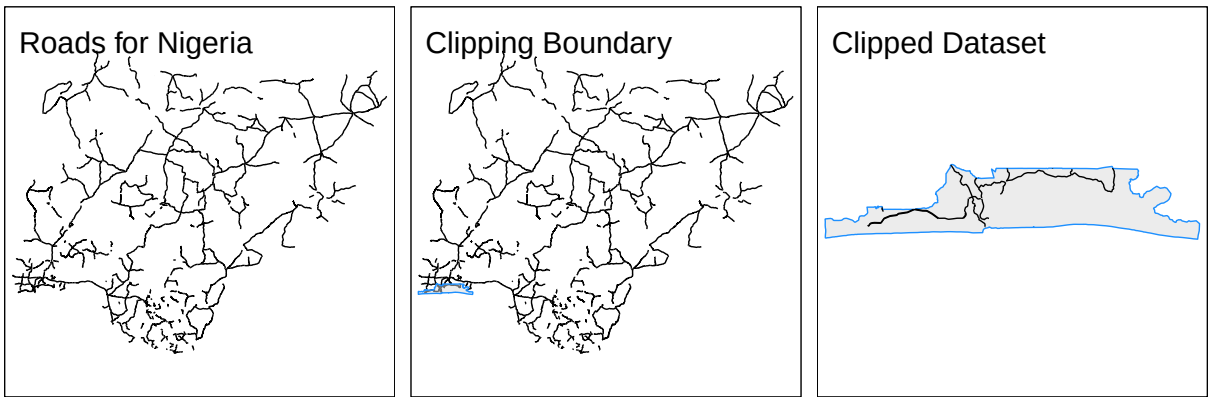
We have road data for the whole of Nigeria which we can use to show clipping. We will use the **st_intersection** function for this:

```
roads <- sf::read_sf("data/roads-cleaned/roads.shp")

roadsLagos <- sf::st_intersection(boundaries, roads)
```

The results of this clipping are shown below:

```
tmap_mode("plot")

# Plot Data
map_roads_nigeria <-
  tm_shape(roads) +
  tm_lines() +
  tm_layout(title = "Roads for Nigeria")
```

```
map_boundary_lagos <-
  tm_shape(roads) +
  tm_lines() +
  tm_shape(boundaries) +
  tm_polygons(alpha = 0.5, border.col = "dodgerblue") +
  tm_layout(title = "Clipping Boundary")

map_roads_lagos <-
  tm_shape(boundaries) +
  tm_polygons(alpha = 0.5, border.col = "dodgerblue") +
  tm_shape(roadsLagos) +
  tm_lines() +
  tm_layout(title = "Clipped Dataset")

tmap_arrange(map_roads_nigeria, map_boundary_lagos, map_roads_lagos, ncol = 3)
```



Clip Raster Data

We can combine our spatial datasets with the GRID3 population data. Unfortunately rasters do not play well with **sf** yet. We must therefore convert the data to an **sp** object. R requires two steps:

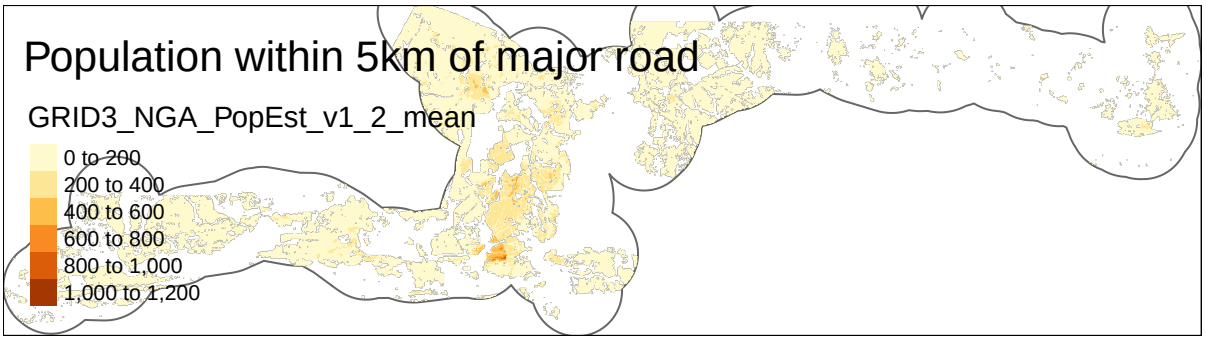
1. `raster::crop()`
2. `raster::mask()`

We will use these both to crop the population raster to Lagos state:

```
roads_buffer_sp <- as(roads_buffer, 'Spatial')
```

```
population_road_buffer <- raster::crop(population, roads_buffer_sp) %>%
  raster::mask(roads_buffer_sp)
```

```
tm_shape(roads_buffer_sp) +
  tm_polygons(alpha = 0) +
tm_shape(population_road_buffer) +
  tm_raster() +
  tm_layout(title = "Population within 5km of major road")
```


Population within 5km of major road
GRID3_NGA_PopEst_v1_2_mean
0 to 200
200 to 400
400 to 600
600 to 800
800 to 1,000
1,000 to 1,200