GP Catchment Demo

Tony Hirst, obo PDS, 22/2/17

A demonstration of how to generate a simple report containing static and dynamic (leaflet) choropleth maps using RMarkdown to generate an HTML or PDF output document.

If the interactive map code is commented out, the report can also be exported as a PDF.

The report to be generated shows the number patients registered to a specified GP practice by the LSOA of their home address.

The data can be found on the NHS Digital website: Numbers of Patients Registered at a GP Practice [CSV data]. *h/t Carl Baker, HoC Library.*

Importing the Data

Import the data into a dataframe, and preview the first few rows.

```
fp='/Users/ajh59/Dropbox/onthewight/github-openHealthDataDoodles/notebooks/data/gp-reg-patients-LSOA-al
#Alternatively, we could load it in directly from the CSV URL
df=read.csv(fp, sep=",")
head(df, 5)
```

```
##   PRACTICE_CODE                NAME LSOA_CODE MALE_PATIENTS
## 1        A81001 THE DENSHAM SURGERY E01012187             0
## 2        A81001 THE DENSHAM SURGERY E01012188            42
## 3        A81001 THE DENSHAM SURGERY E01012189            22
## 4        A81001 THE DENSHAM SURGERY E01012190            63
## 5        A81001 THE DENSHAM SURGERY E01012191            71
##   FEMALE_PATIENTS ALL_PATIENTS
## 1               4            4
## 2              45           87
## 3              28           50
## 4              86          149
## 5              83          154
```

We can also filter the data to just the data associated with a particular GP practice, if we know the practice code.

We can also move to prettier tables with the ***kable*** *function from the* ***knitr*** *library. (A package is currently under developer (****printr****) that bakes a lot of pretty pretting, such as* ***kable*** *styling, in to the* ***knitr*** *Rmd output document generation process as a default.)*

```
thisGP= subset(df,PRACTICE_CODE==gpcode)
kable(head(thisGP[,c('LSOA_CODE','MALE_PATIENTS','FEMALE_PATIENTS','ALL_PATIENTS')], 5),
      row.names=FALSE, caption=paste(thisGP[1,'NAME'],' (',gpcode,')',sep=''))
```

Table 1: BEECH GROVE SURGERY (J84020)

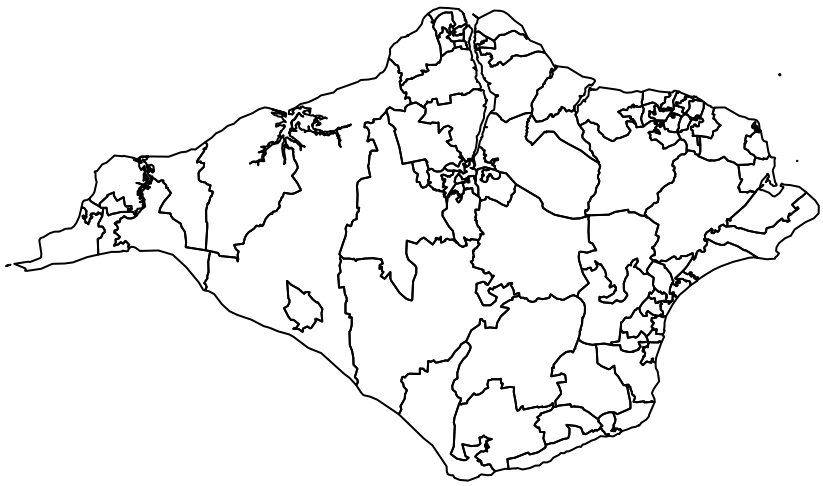
LSOA_CODE	MALE_PATIENTS	FEMALE_PATIENTS	ALL_PATIENTS
E01015466	0	1	1
E01017135	0	1	1
E01017282	5	4	9
E01017283	34	33	67
E01017284	45	51	96

Including Plots

Maps can be plotted from boundary files loaded in from a wide range of formats using the `rgdal` library.

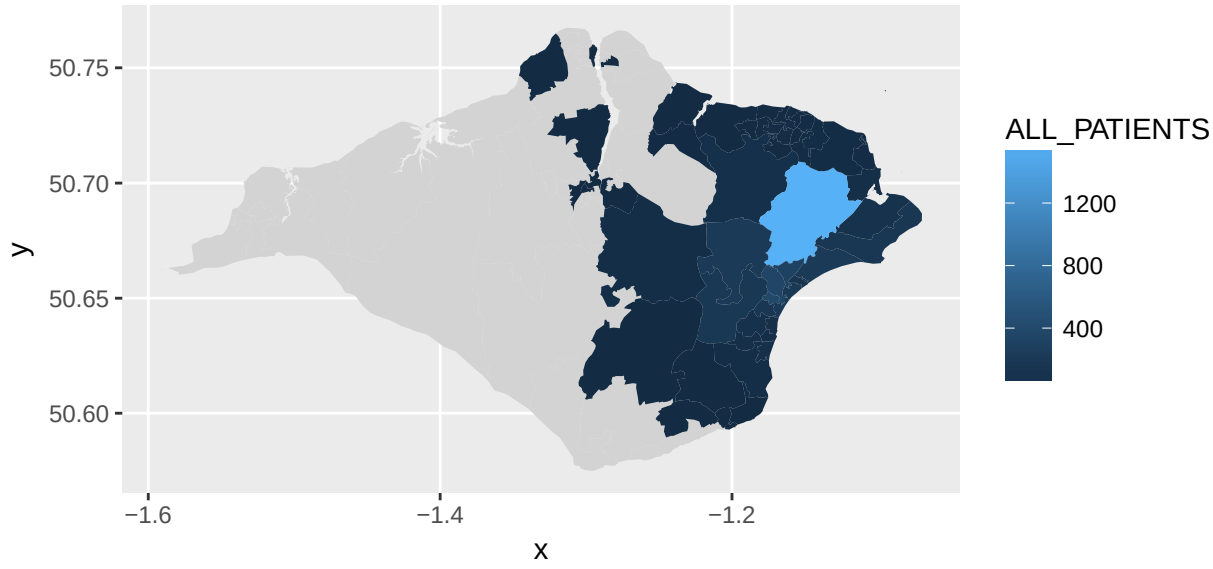
In this case, I'll load in the boundary file for LSOAs in the Isle of Wight local authority area as a *geoJSON* file.

```
library(rgdal)

geoj= readOGR("/Users/ajh59/Dropbox/onthewight/IWgeodata/lsoa_by_lad/E06000046.json", "OGRGeoJSON",
              verbose = FALSE) # The verbose switch prevents printing of read diagnostics

#Preview the boundary file
plot(geoj)
```



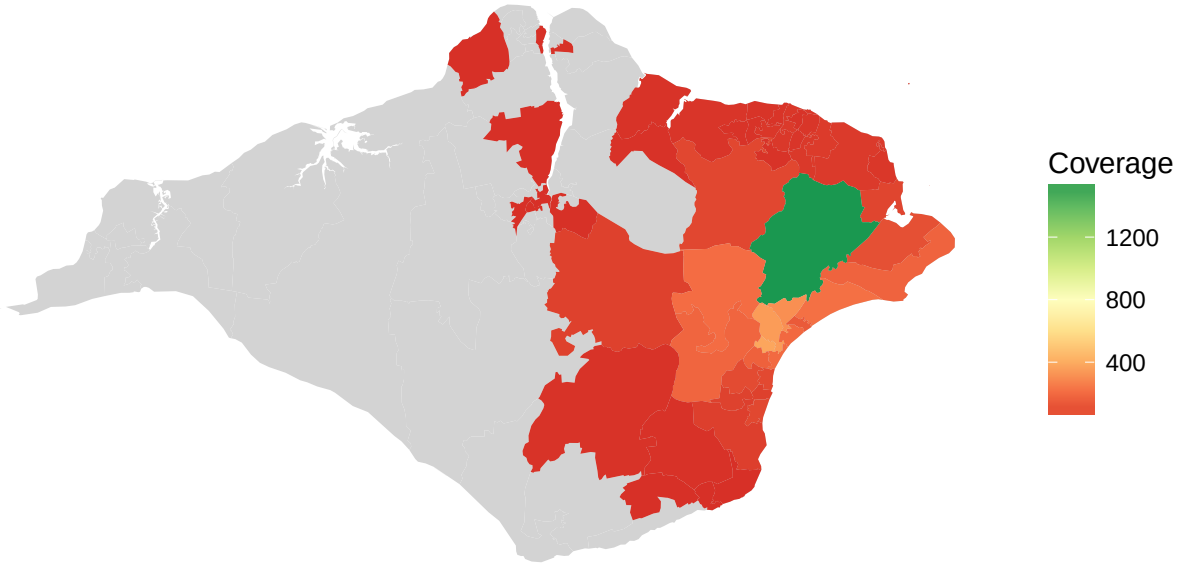
```
#http://bl.ocks.org/prabhasp/raw/5030005/
library(rgeos)
#requires maptools, mapproj
library(ggplot2)

#We need to make sure the data is presented in the correct format...
#The codes we want to map against in the boundary file are in the LSOA11CD column
geojf = fortify(geoj, region = "LSOA11CD")

#Plot the map as a choropleth map uwing the ggplot geom_map geom.
#The aesthetic identfies the column in the data file that identifies the boundary area code
#Use two geom_maps - one to render *all* the boundaries in the area...
```

```
g = ggplot() + geom_map(data = df, aes(map_id = LSOA_CODE), map = geojf, fill='lightgrey')

# the other to render the choropleth areas
g = g + geom_map(data = thisGP, aes(map_id = LSOA_CODE, fill = ALL_PATIENTS),  map = geojf )
#If we omit the first geom_map, only the boundaries of areas associated with the data file will be plot

#Set the zoom scale so we can see all the rendered boundaries
g = g + expand_limits(x = geojf$long, y = geojf$lat)  + coord_map()

#Display the map
g
```



We can override the default background and colour scale. I'm not suggesting the following color scale improves matters!

```
#https://rstudio-pubs-static.s3.amazonaws.com/78148_6dd49b5dab4c4f5a8b1a74e5893ff17d.html
library(RColorBrewer)
#Define a new color palette using a ColorBrewer scale
newpalette = brewer.pal(9,"RdYlGn")

#The theme_void() theme drops the axes and the grid
g + theme_void() + scale_fill_gradientn(name="Coverage", colours = newpalette)
```



Generating an Interactive Leaflet Map

As well as generating static "for print" maps that work well with PDF documents, we can also generate interactive maps using the `leaflet` Javascript library that work well within HTML web page outputs.

The easiest way of generating the plots is to use the *R* `leaflet` library and pass it a shape file that *includes* the data we want to plot as one of its columns.

Generate a sub-dataset containing just the geography codes and the values we want to plot, renaming the selected value column with a conventional name.

```
datadf = thisGP[ , c('LSOA_CODE','ALL_PATIENTS') ]

names(datadf)[names(datadf)=="ALL_PATIENTS"] <- "datacol"
head(datadf, 5)
```

```
##        LSOA_CODE datacol
## 480818 E01015466       1
## 480819 E01017135       1
## 480820 E01017282       9
## 480821 E01017283      67
## 480822 E01017284      96
```

We can now merge this data in a data value column to the object that represents the shapefile.

```
#Do a left outer join so we keep all the shapefile codes...
geoj@data = merge(geoj@data, datadf, by.x='LSOA11CD', by.y='LSOA_CODE', all.x = TRUE)
```

```
#We could actually be more defensive and just use the two original cols from the geojson file
#This prevents problems if we repeatedly merge...
#geoj@data=merge(geoj@data[,c('LSOA11CD','LSOA11NM')], datadf,
#                by.x='LSOA11CD',by.y='LSOA_CODE', all.x = TRUE)

head(geoj@data)
```

```
##    LSOA11CD           LSOA11NM datacol
## 1 E01017282 Isle of Wight 006A       9
## 2 E01017283 Isle of Wight 006B      67
## 3 E01017284 Isle of Wight 010A      96
## 4 E01017285 Isle of Wight 010B     163
## 5 E01017286 Isle of Wight 005A      14
## 6 E01017287 Isle of Wight 005B       9
```

The reason we do this is that the `leaflet` library plays nice if the data is all in one place...

Note that in the original *Rmd* document, the `eval=htmlout` chunk parameter is added to the chunk configuration; if the *Rmd* file is being knitted to a PDF output document, `hhtmlout` is set to `FALSE` and the chunk is not evaluated (that is, the code in the chunk is not executed and the interactive map is not generated).

```
library(leaflet)
#https://rpubs.com/walkerke/leaflet_choropleth
#http://rstudio.github.io/leaflet/choropleths.html

#Set up a colour palette of our own - we could probably auto generate bins if we wanted to...
bins = c(0,10,50,150,250,500, 1000, Inf)
pal = colorBin("YlOrRd", domain = geoj$datacol, bins = bins)

#Generate some popup text for each area
gp_popup<- paste0("<strong>GP Practice code: </strong>",
                      gpcode,
                      "<br/><strong>LSOA Code: </strong>",
                      geoj$LSOA11CD,
                  "<br/><strong>Value: </strong>",
                      geoj$datacol)

#The %>% operator lets us chain leaflet package methods
#Pass in the basemap
leaflet(data = geoj) %>%
  #THere is flexibility over what map tiles you use
  addProviderTiles(providers$CartoDB.Positron) %>%
  #Add the choropleth layer
  addPolygons(fillColor = ~pal(datacol),
              fillOpacity = 0.6,
              color = "#FFFFFF",  #colour of boundary line
              weight = 1, #weight of boundary line
              popup = gp_popup #Add the popup function - click an area to display it
              ) %>%
  #Add a legend
  addLegend(pal = pal, values = ~datacol, opacity = 0.7, title = NULL, position = "bottomright")
```

Generating Multiple Report Document Instances

By parameterising this Rmd file with the code of the GP practice we are interested in, and perhaps also the Local Authority code for idntifying an appropriate shapefile, we can use it to generate a set of reports, such as one separate report PDF document or HTML page per GP practice. See the documentation on parameterised reports for more information.