GIS with R
(or without ArcGIS)

Omar García-Ponce
New York University

Spring Lab Workshop

May 7, 2013

Why use R?

1. ArcGIS is expensive (only available for Windows users)
2. R is free (available for Linux, Mac and Windows users)
3. The simplest map in R requires no data files
4. R allows more “control” over features
5. You can run the same script and produce the same map

Getting started

Basic tools:

- **library(maps)** #for creating geographical maps
- **library(mapdata)** #contains basic data for 'maps'
- **library(maptools)** #tools for handling spatial objects
- **library(mapproj)** #for creating projected maps
- **library(raster)** #tools to deal with raster maps
- **library(ggplot2)** #to create maps
- **library(gpclib)** #general polygon clipper

Simple Maps

A very simple map

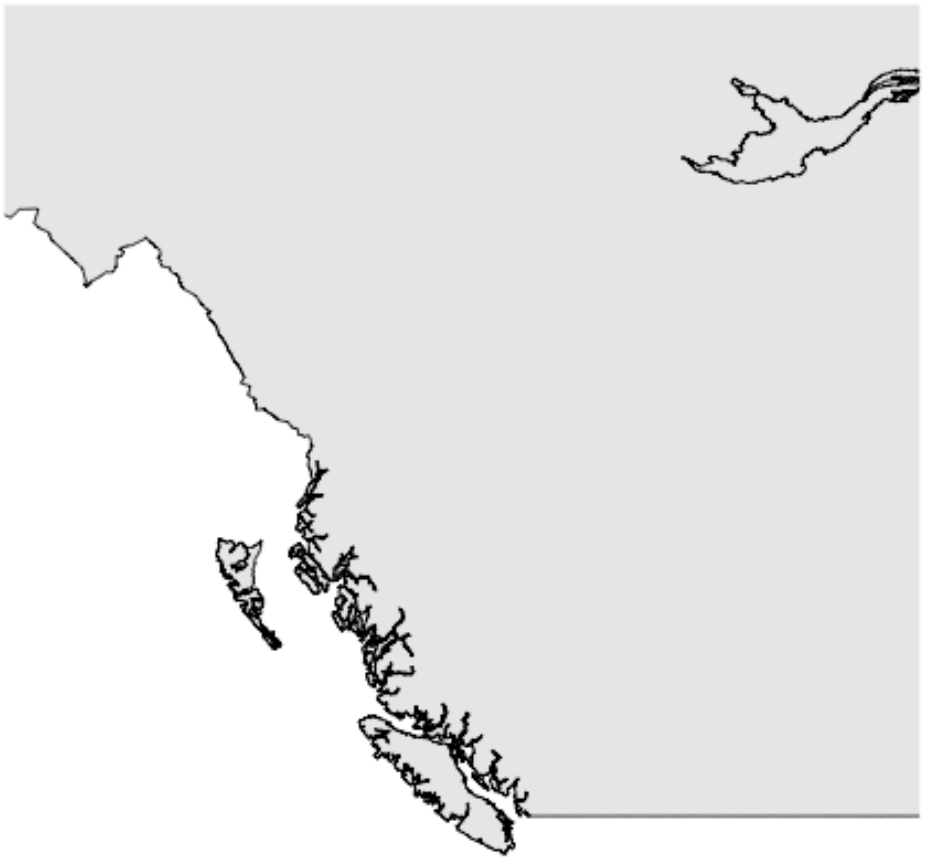
```
library(maps)
library(mapdata)


map("worldHires", "Mexico",
    xlim=c(-118.4, -86.7),
    ylim=c(14.5321,  32.71865),
    col="blue" , fill=TRUE)
```

Another simple map

```
library(maps)

library(mapdata)

map("worldHires","Canada",
    xlim=c(-141,-53),
    ylim=c(40,85),
    col="gray90",
    fill=TRUE)
```

Zoomed in

```
map("worldHires",
    "Canada",
    xlim=c(-140,-110),
    ylim=c(48,64),
    col="gray90",
    fill=TRUE)
```



Read in data: shapefiles

library(maptools) #for shapefiles

- 'readShapePoly'
  - Read in a polygon shape layer (e.g., administrative boundaries, national parks, etc.). This means the layer is of type "polygon" (i.e. not lines, such as roads or rivers)
- 'readShapeLines'
  - read in a line shape layer
- 'readShapePoints'
  - read in a point shape layer

Read in data: GPS points

- Read in your data, as you would any other .csv file.
- For GPS points, must have columns individually for latitude and longitude.
- Must be in decimal degrees.

Example (by Kimberly Gilbert)

```
library(maps)
library(mapdata)
library(maptools)  #for shapefiles
library(scales)  #for transparency
pcontorta <- readShapePoly("pinucont.shp")   #layer of data for
species range
samps <- read.csv("FieldSamples.csv")   #my data for sampling
sites, contains a column of "lat" and a column of "lon" with GPS
points in decimal degrees
```

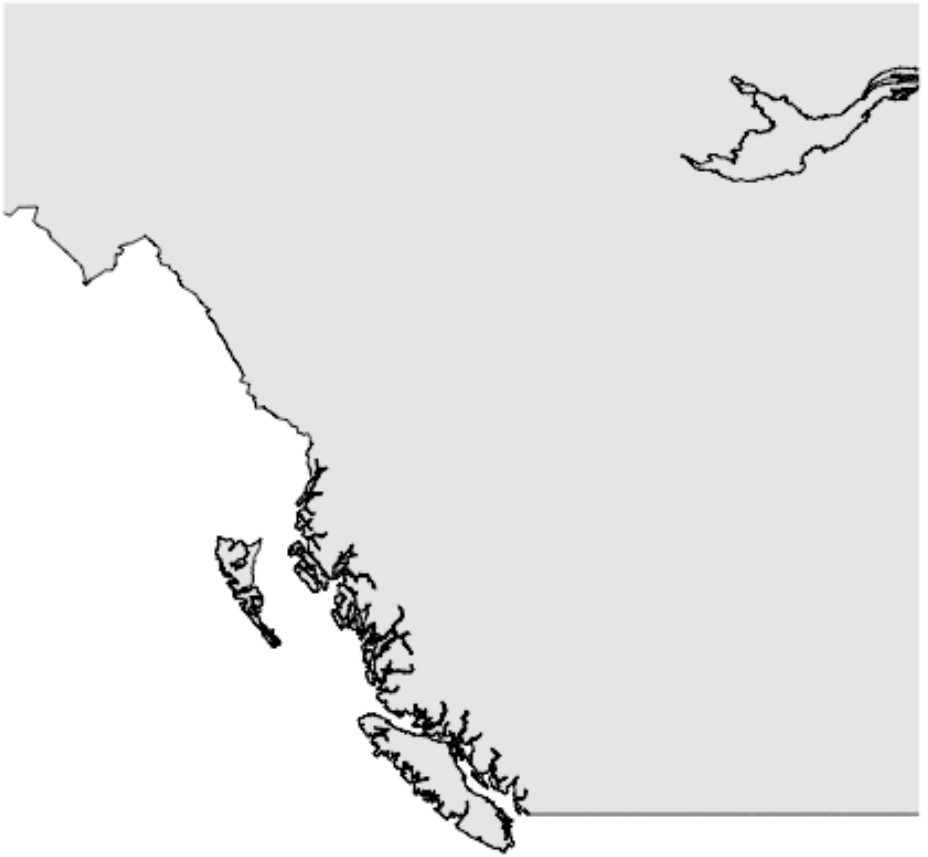
Adding layers

1. Plot the base map
   - This will be the final shape and size of your full map
2. Plot layers or data onto this map
   - Add your shape files or data points
   - Plotting occurs in order, unless you use transparency, you will cover things up.
3. Add any final touches you'd like

Remember to use "add=TRUE" for every step after the first

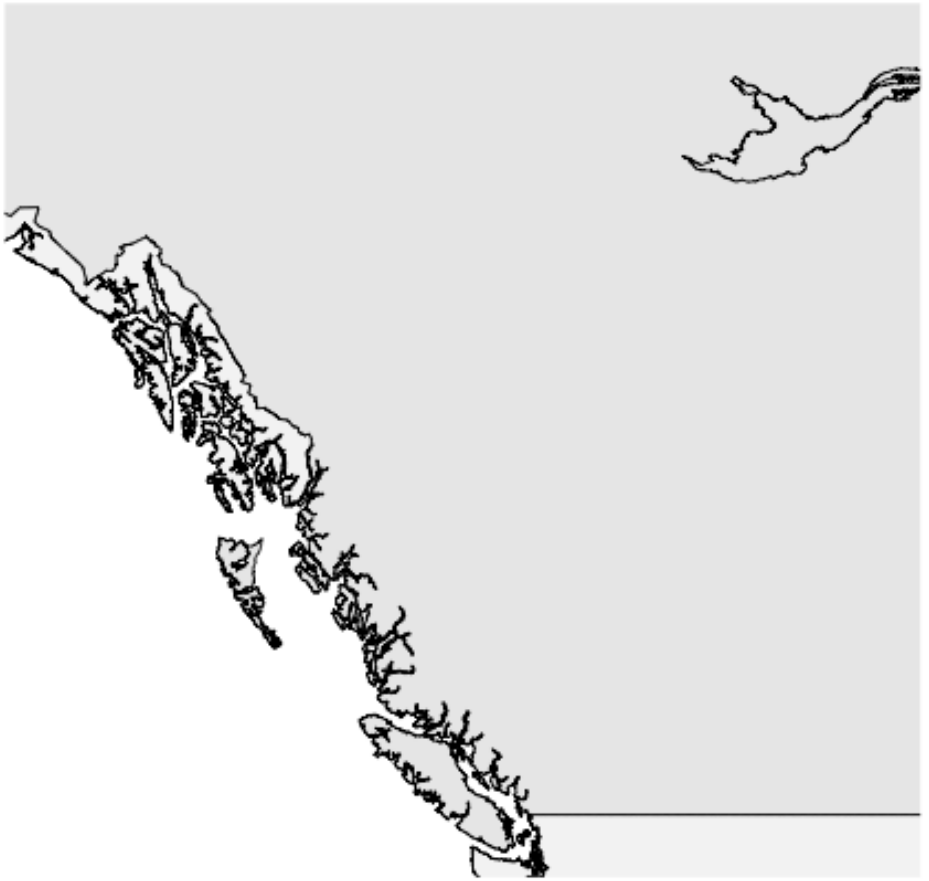
Plot map 1

```
map("worldHires",
    "Canada",
    xlim=c(-140,-110),
    ylim=c(48,64),
    col="gray90",
    fill=TRUE)
```



Add map 2

```
map("worldHires",
    "usa",
    xlim=c(-140,-110),
    ylim=c(48,64),
    col="gray95",
    fill=TRUE,
    add=TRUE)
```



Add shapefile layer

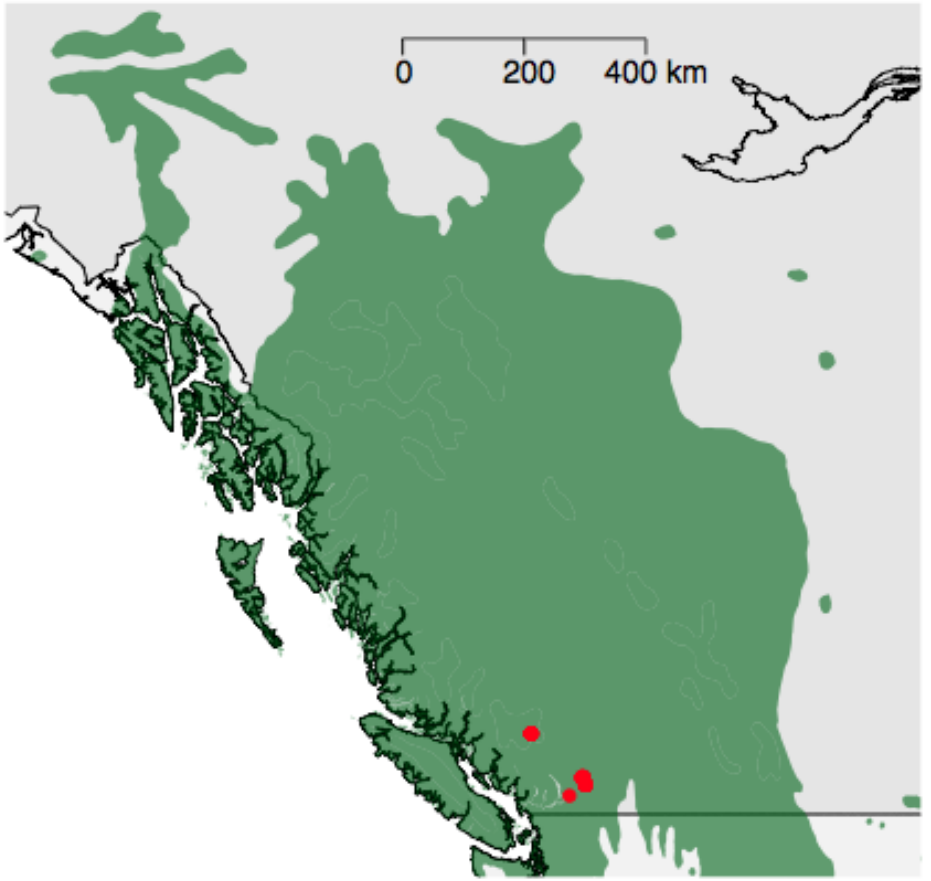
```
plot(pcontorta,
   add=TRUE,
   xlim=c(-140,-110),
   ylim=c(48,64),
   col=alpha("darkgreen",
    0.6),
   border=FALSE)
```

Add GPS points

```
points(samps$Long,
    samps$Lat,
    pch=19,
    col="red",
    cex=1)
```

Add extras

```
map.scale(-127, 63.5,
    ratio=FALSE,
    relwidth=0.2,
    cex=1.2)
```



More Involved Maps

Shapefiles

The data that comprise a shapefile are usually stored in (at least) three individual files:

1. **.shp:** shape format; the feature geometry itself
2. **.shx:** shape index format; a positional index of the feature geometry.
3. **.dbf**: attribute format; columnar attributes for each shape.

Manipulate DBFs to add attributes

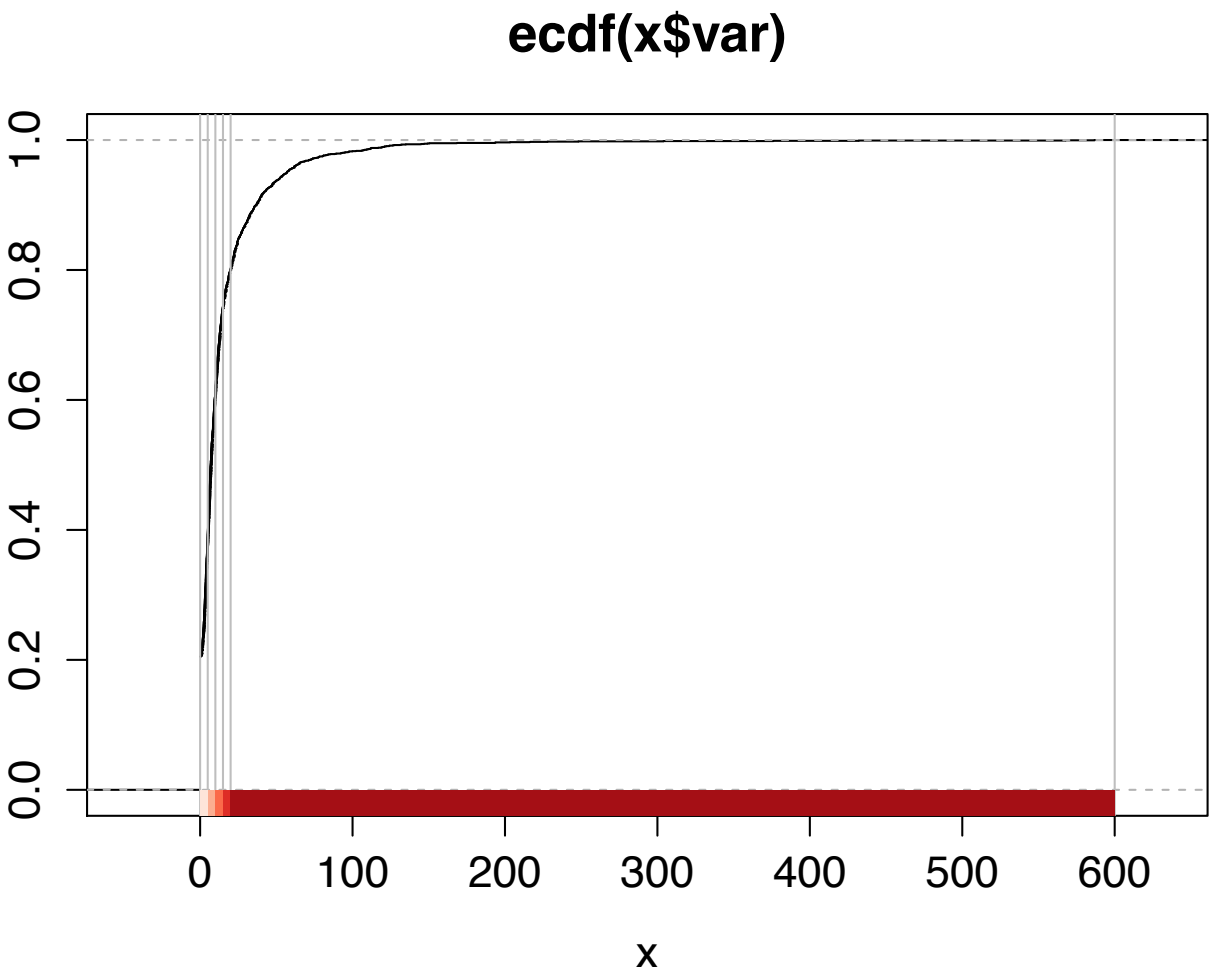
```
# Load required packages
library(maptools)
library(RColorBrewer)
library(classInt)

# Set the working directory
setwd("~/Dropbox/MEX_MAPS/")

# Load data frame
frame<- read.dbf("municipios.dbf")

# Load attributes file
killings <- read.dta("killings.dta")

# Join attributes to frame
joined <- merge(drugkillings, frame, by=c("ID"))

# Save old version of dbf file
write.dbf(frame, "~/Dropbox/MEX_MAPS/oldmunicipios.dbf")

# Save new version of dbf file
write.dbf(joined, "~/Dropbox/MEX_MAPS/municipios.dbf")
```

Load shapefile and set breaks to map the data

```
# Load new shapefile
mexico<- readShapePoly("municipios.shp")

# have a look at the attribute table headings
names(mexico)

# select a colour palette and the number of colours you wish to display.
colours <- brewer.pal(5, "Reds")

# We need to set breaks in the data in order that we have a representative
# colour palette. This can be done using the classIntervals function
# in the classInt package:
brks<-classIntervals(mexico$homds4,  n=6, style="fixed",
                     fixedBreaks=c(0, 5, 10, 15, 20, 600))
```

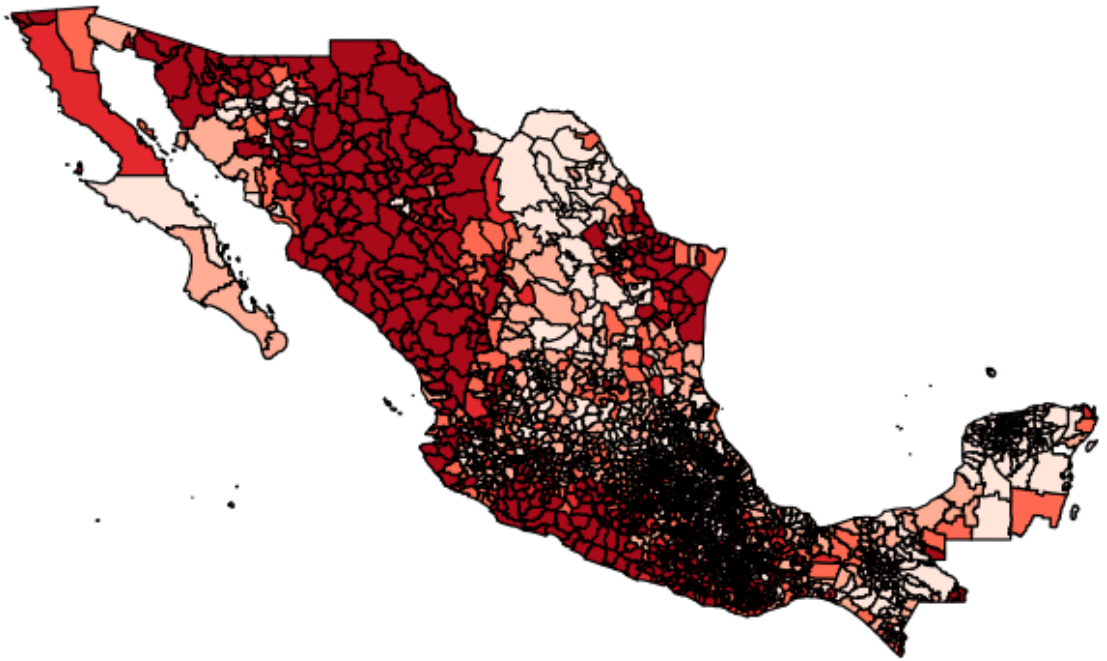
Plot the distribution of the data and colors assigned

```
# using the plot function you can plot the distribution of the data and
# view the colours assigned to each point

plot(brks, pal=colours)
```



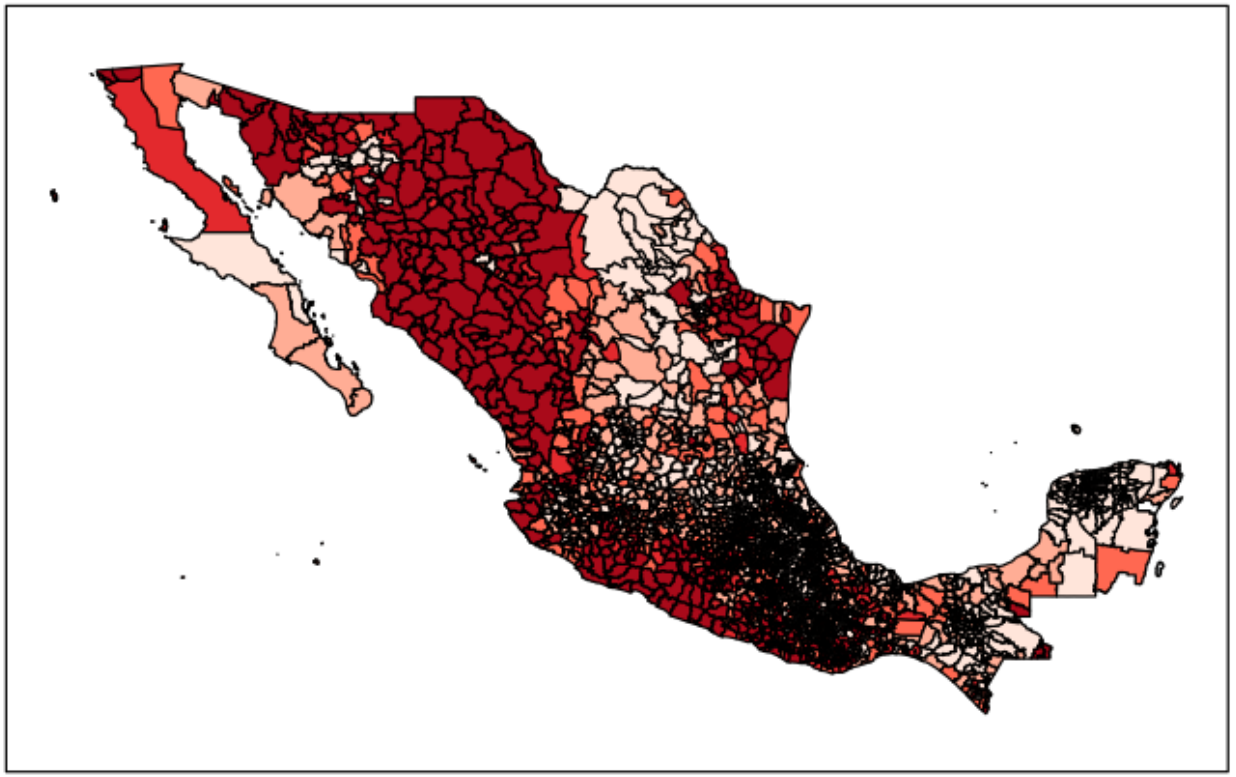
Plot the map

```
# from this point on we are only interested in the break values, we
# can therefore extract them from the brks object above:
brks<- brks$brks

##Now we can produce the map:
plot(mexico, col=colours[findInterval(mexico$homds4, brks, all.inside=TRUE)]
     , axes=F)
```



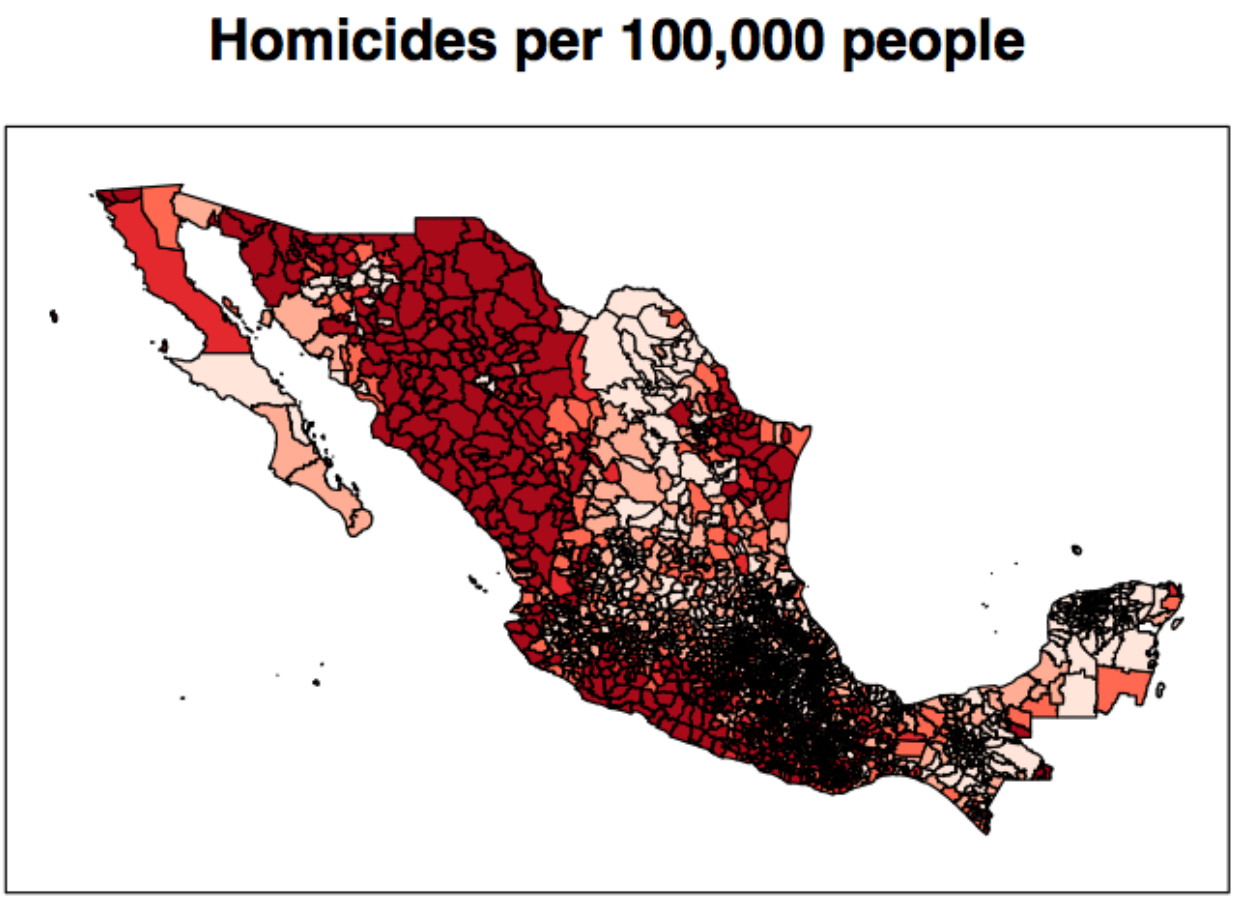
Add a border

```
## we can add a border:
box()
```



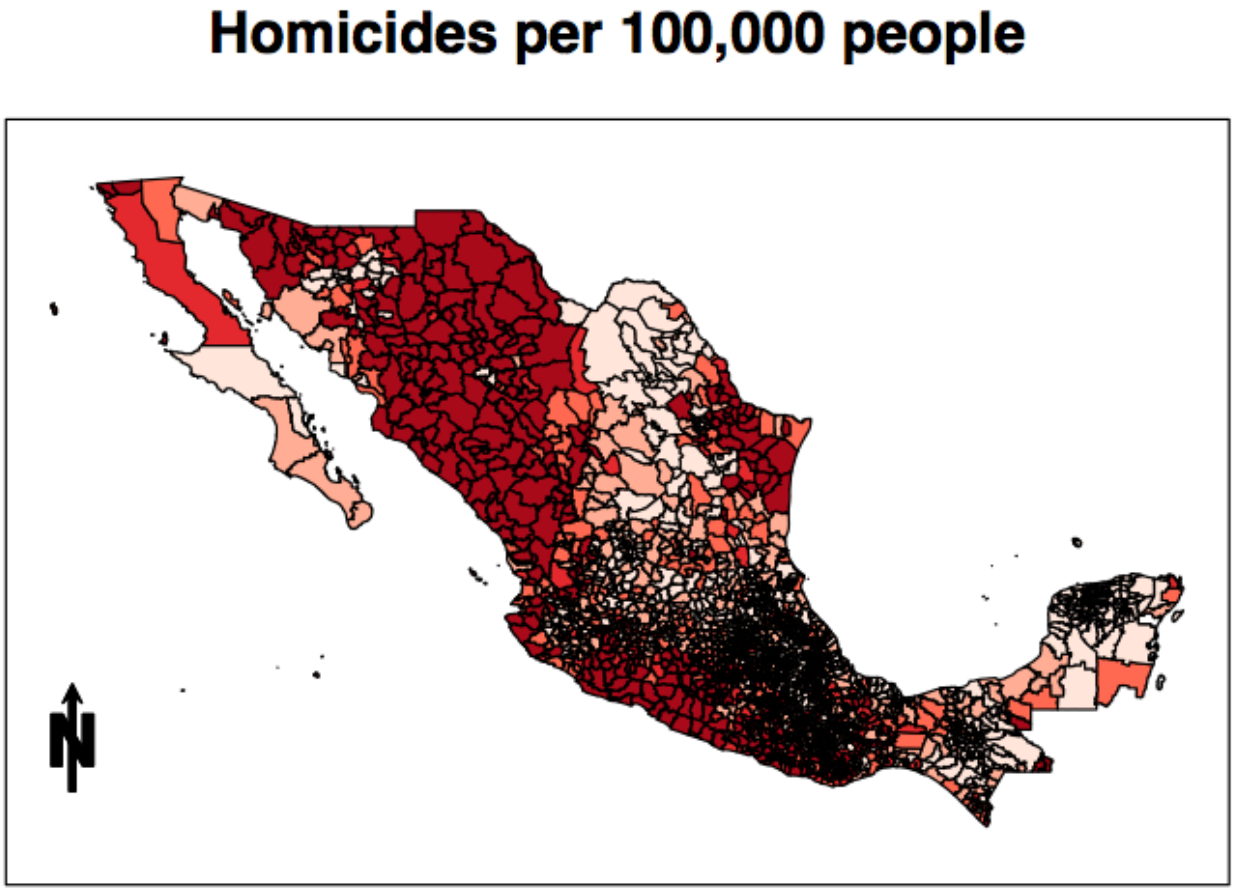
Add a title

```
## a title:
title(paste ("Homicides per 100,000 people"))
```



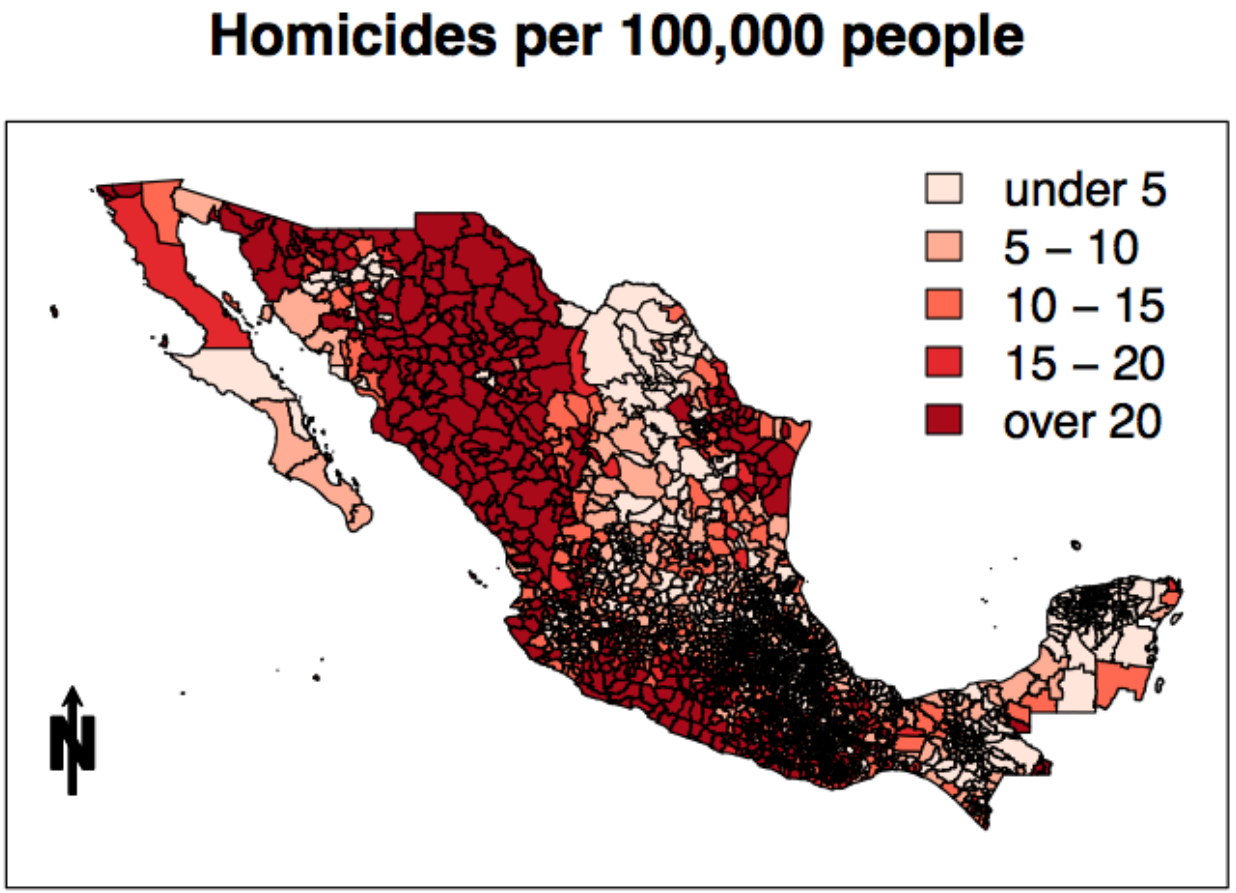
Add a north arrow

```
## a north arrow:
SpatialPolygonsRescale(layout.north.arrow(1), offset= c(-118.4, 15.53),
                       scale = 3, plot.grid=F)
```



Add a legend

```
# a legend
legend(x=-95.1, y=34.1, legend=leglabs(brks), fill=colours, bty="n")
```



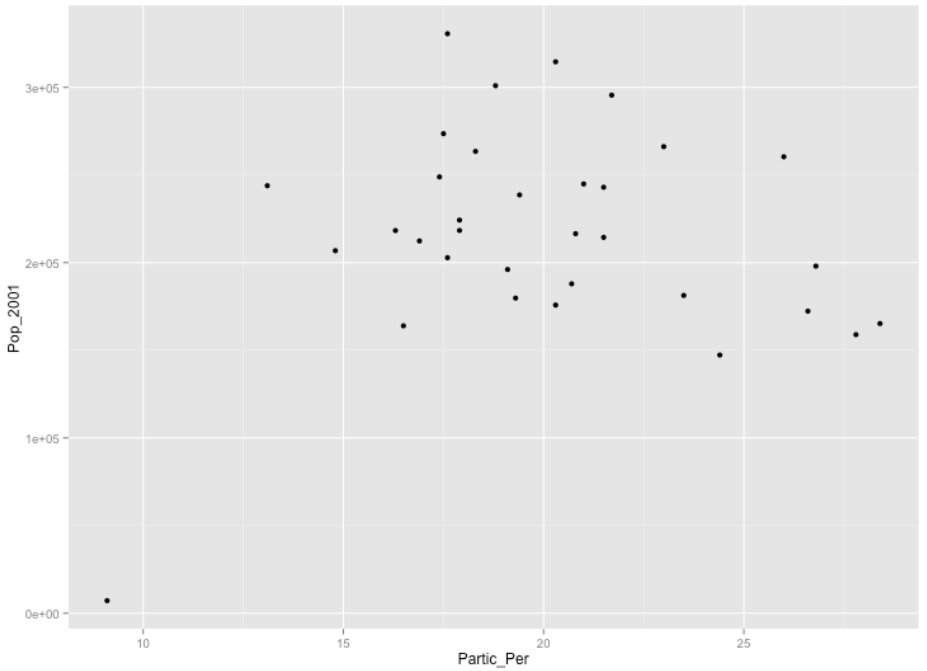
Maps with ggplot

Getting started (by James Cheshire)

```
library(maptools)
library(ggplot2)
library(gpclib)

# set the working directory
setwd("~/Dropbox/GISwR/")

## load the shapefile
sport<- readShapePoly("london_sport.shp")

names(sport)
```

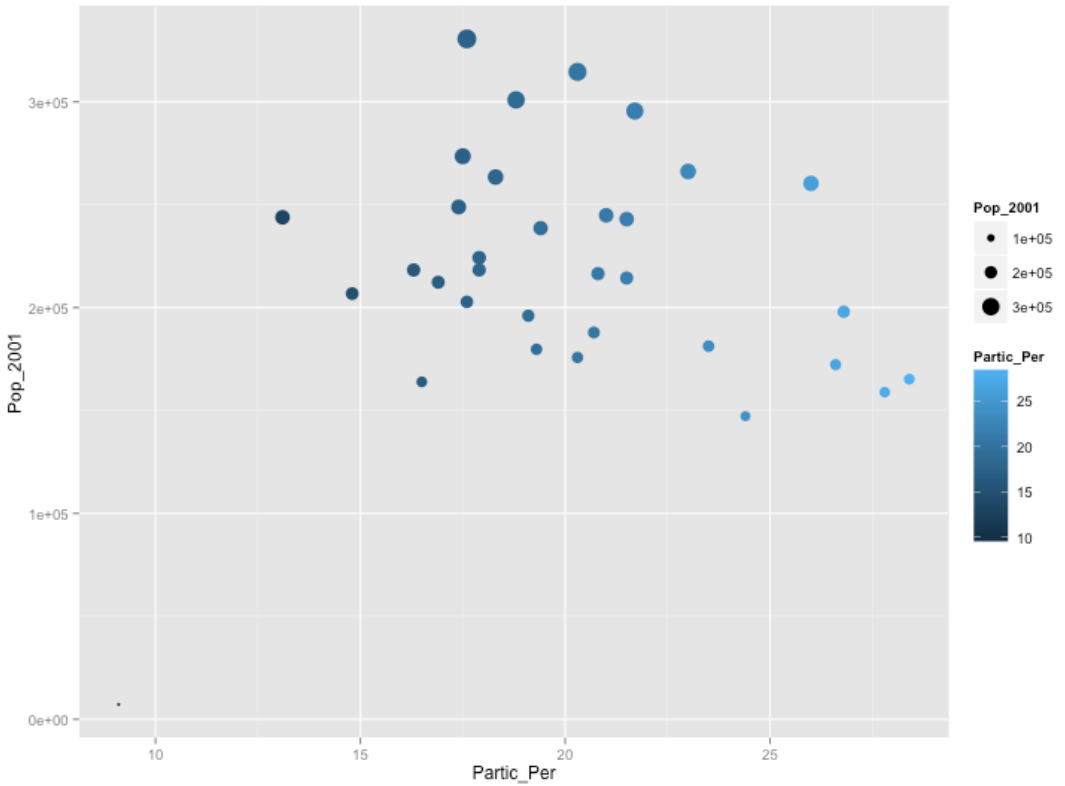
Plot the data to set up a ggplot object indicating the input data (i.e., the attribute table of the shapefile), and what parts wish to use:

```
# As a first attempt with ggplot2 we can create a scatter
# plot with the data

p<-ggplot(sport@data, aes(Partic_Per,Pop_2001))

p+geom_point()
```



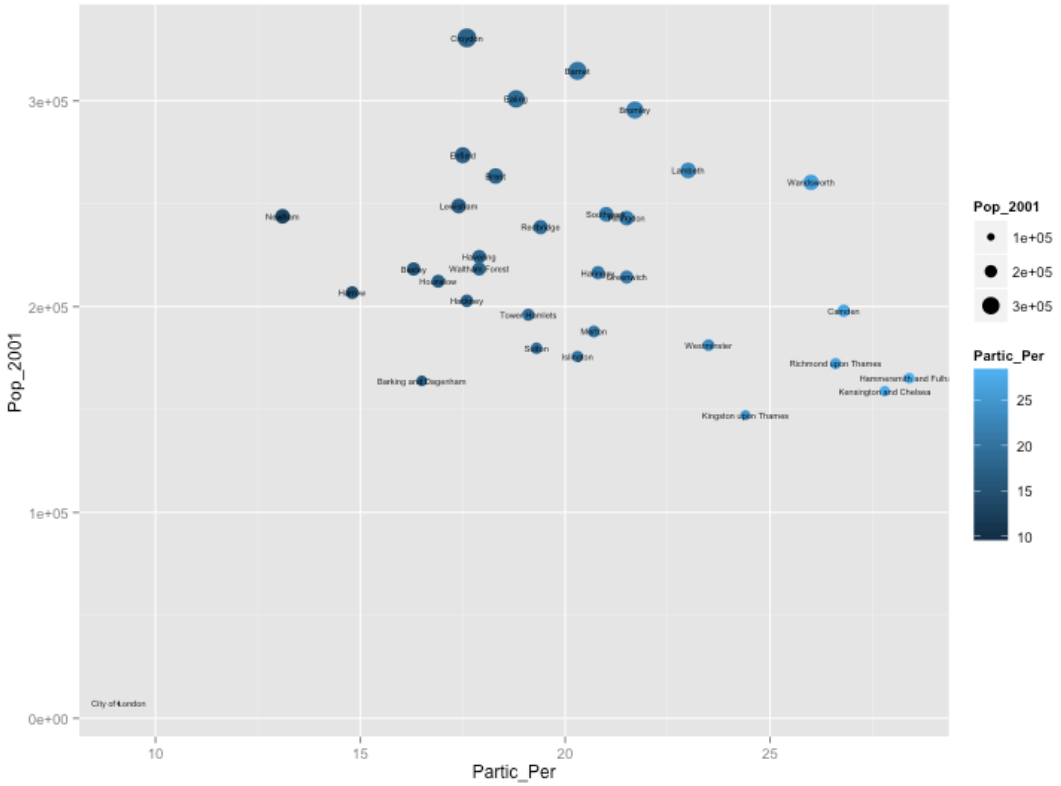
You can alter the nature of the points

```
# If you want to scale the points by borough population
# and colour them by sports participation this is also
# fairly easy by adding another aes()
p+geom_point(aes(colour=Partic_Per, size=Pop_2001))
```



You can add text

```
# The real power of ggplot2 lies in its ability to add layers to a plot.
# In this case we can add text to the plot

p+geom_point(aes(colour=Partic_Per,size=Pop_2001)) +
  geom_text(size=2,aes(label=name))
```

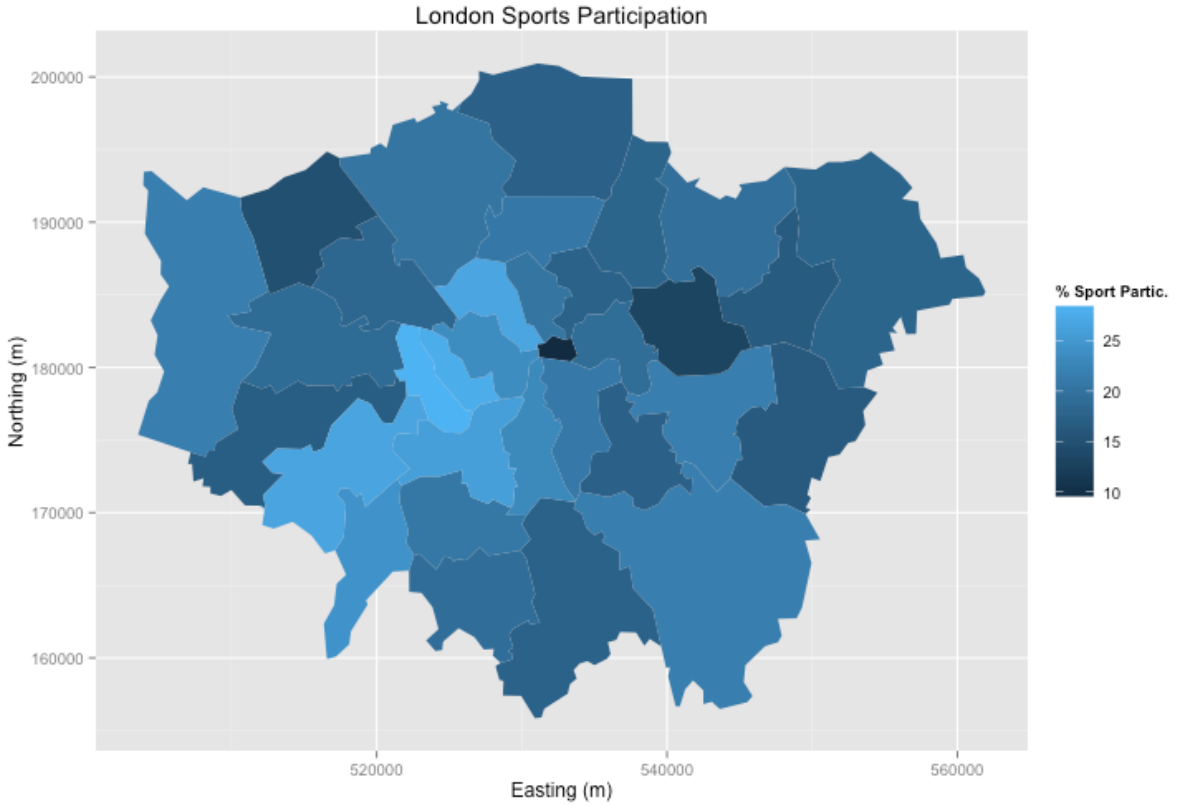


Now get the shapefiles into a format that can be mapped using the fortify( ) function...

```
# To get shapefiles into a format that can be plotted we have to use
# the fortify() function.

gpclibPermit()

sport_geom<- fortify(sport, region="ons_label")

# This step has lost the attribute information associated with
# the sport object. We can add it back using the merge function:

sport_geom<- merge(sport_geom, sport@data, by.x="id",
                   by.y="ons_label")
```

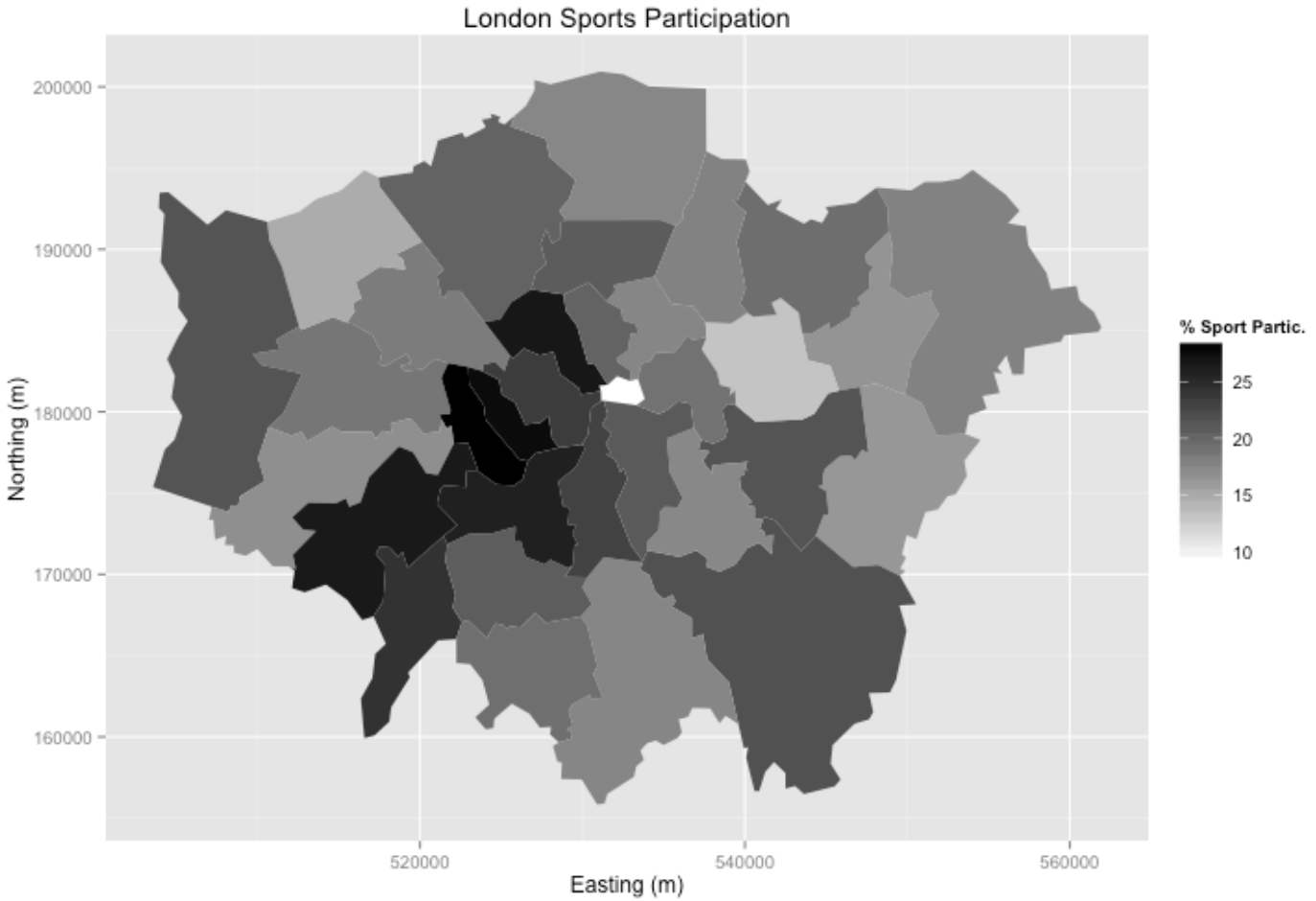
Create the map

```
Map<- ggplot(sport_geom, aes(long,lat, group=group, fill=Partic_Per)) +
  geom_polygon()+ coord_equal() +
  labs(x="Easting (m)", y="Northing (m)",fill= "% Sport Partic.") +
  ggtitle ("London Sports Participation")

Map
```



Customize the map

```
Map + scale_fill_gradient(low="white", high="black")
```



Zonal Statistics with QGIS