Maps in R

In this tutorial we'll cover the following topics:

* [Basic Maps](#basic)
* [Heat Maps](#heat)
* [Leaflet Maps](#leaflet)

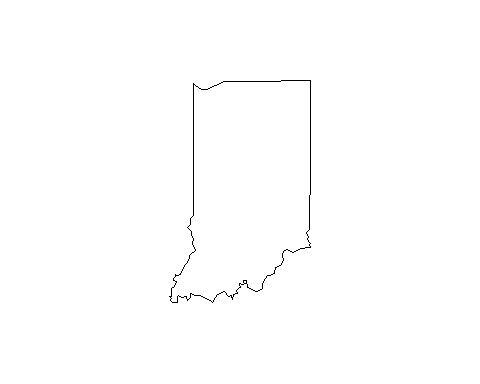
Advanced topic:

* [Spatial Data](#spatial)

## Basic Maps

There is no function for displaying maps in the base R functionality. To display a simple map, we use the maps package. First, let's create an Indiana map.

library(maps)  
map(database = 'state', regions = 'indiana')

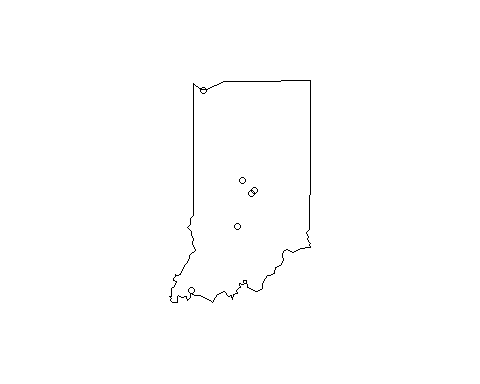


Now we can add a few monitors as points. For illustration, I've created a simple data frame below, with latitude and longitude information.

monitors <- read.table(header=T, text='  
 monitorid lat long name   
 1 41.60668 -87.304729 Gary-IITRI  
 2 39.811097 -86.114469 Indpls-Washington-Park  
 3 39.749019 -86.186314 Indpls-Harding-St.  
 4 38.013248 -87.577856 Evansville-Buena-Vista  
 5 39.159383 -86.504762 Bloomington  
 6 39.997484 -86.395172 Whitestown  
 ')

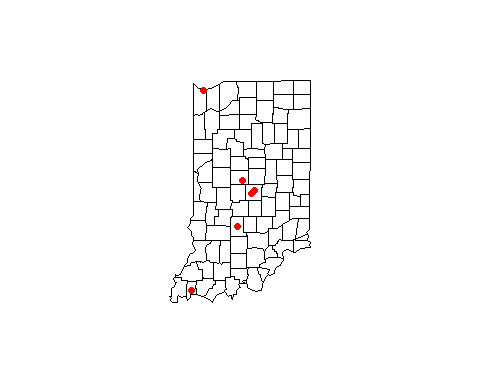
Now we can use the longitude column as the x-axis values and latitude for the y-axis values in the points() function. This function adds points to an already existing plot in R.

points(x = monitors$long, y = monitors$lat)



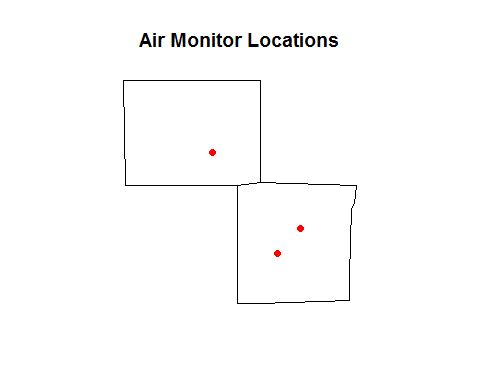
We can jazz it up a bit by making it a county map and changing the symbol type and color.

map(database = 'county', regions = 'indiana')  
points(x = monitors$long, y = monitors$lat, pch = 19, col = 'red')



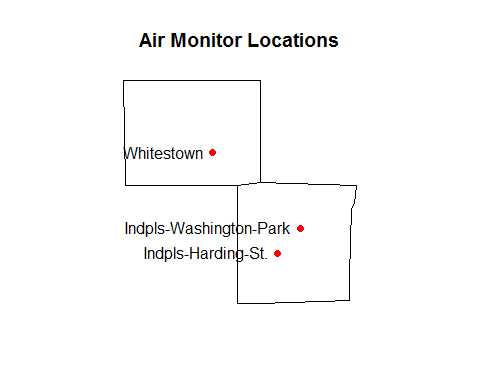
We can also select specific counties to look at, and give the map a title.

map(database = 'county', regions = c('indiana,marion', 'indiana,boone'))  
points(x = monitors$long, y = monitors$lat, pch = 19, col = 'red')  
title(main = "Air Monitor Locations")



If we want to label the monitors, we use the text() function.

map(database = 'county', regions = c('indiana,marion', 'indiana,boone'))  
points(x = monitors$long, y = monitors$lat, pch = 19, col = 'red')  
title(main = "Air Monitor Locations")  
text(x = monitors$long, y = monitors$lat, labels = monitors$name, pos = 2)



## Heat Maps

The choroplethr package makes it fairly easy to create heat maps in R. For illustration, here's a simple data frame of the number of nonattainment areas for 8-hr ozone (2008 standard) by state in Region 5 (reference).

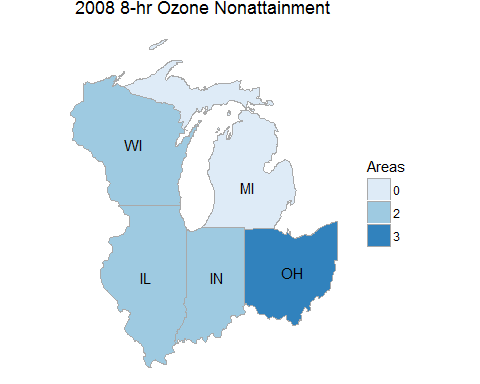
nonattain <- read.table(header=T, text='  
 state areas  
 illinois 2  
 indiana 2  
 michigan 0  
 ohio 3  
 wisconsin 2  
 ')

Now we'll use the choroplethr package to make a heat map. The state\_choropleth() function requires the column with state names to be labeled "region" and the column with the numeric values to be labeled "value". To do this, we'll introduce another handy dplyr function, rename().

library(dplyr)  
  
nonattain <- rename(nonattain, region = state, value = areas)

Now we can use the state\_choropleth() function.

library(choroplethr)  
  
nonattain\_map <- state\_choropleth(nonattain, zoom = nonattain$region,  
 title = "2008 8-hr Ozone Nonattainment",  
 legend = "Areas")  
nonattain\_map



## Leaflet Maps

Leaflet is an interactive map that can be created in R using the leaflet package.

library(leaflet)  
m <- leaflet()  
m <- addTiles(m)  
m <- addMarkers(m, lng=monitors$long, lat=monitors$lat,   
 popup=monitors$name)  
m

## Exercises

Exercises for this tutorial can be found here: http://rpubs.com/NateByers/MapsExercises.

## Advanced Topic

### Spatial Data

It is possible to perform any kind of geographical analysis in R.

To illustrate how to create spatial data objects in R, let's re-create the simple monitors data frame, with a small modification:

monitors <- read.table(header=T, text='  
 monitorid long lat datum name   
 1 -87.304729 41.60668 WGS84 Gary-IITRI  
 2 -86.114469 39.811097 WGS84 Indpls-Washington-Park  
 3 -86.186314 39.749019 WGS84 Indpls-Harding-St.  
 4 -87.577856 38.013248 NAD83 Evansville-Buena-Vista  
 5 -86.504762 39.159383 NAD83 Bloomington  
 6 -86.395172 39.997484 NAD83 Whitestown  
 ')

Now we have a "datum" column. We've also put the longitude column first, followed by the latitude column. And to be precise, before we lump all of these monitors together in a data object, we need to split them up by datum.

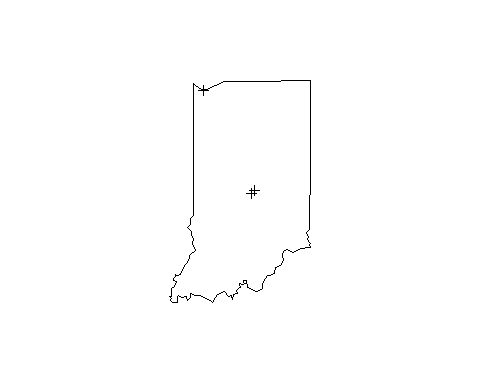
mon\_WGS84 <- filter(monitors, datum == "WGS84")  
mon\_NAD83 <- filter(monitors, datum == "NAD83")

sp is the foundational package for working with spatial data in R. Once we library the package we can create SpatialPoints data objects by using the SpatialPoints() function.

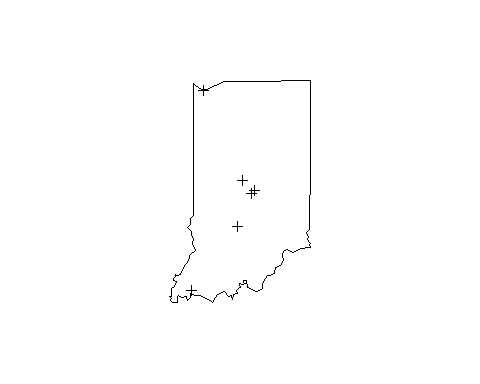
library(sp)  
  
mon\_WGS84\_sp <- SpatialPoints(coords = select(mon\_WGS84, long, lat),  
 proj4string = CRS("+proj=longlat +ellpsWGS84"))  
  
mon\_NAD83\_sp <- SpatialPoints(coords = select(mon\_NAD83, long, lat),  
 proj4string = CRS("+proj=longlat +ellpsNAD83"))

Now we can plot the SpatialPoints objects.

map(database = 'state', regions = 'indiana')  
plot(mon\_WGS84\_sp, add = TRUE)



plot(mon\_NAD83\_sp, add = TRUE)



### Further Reading

Below is a list of resources for learning more about what you can do with geographic data in R.

* For a good introduction to using GIS data in R, see this vignette.
* For a comprehensive overview of spatial data and analysis in R, see Applied Spatial Data Analysis with R.
* The CRAN Task View on Analysis of Spatial Data is a comprehensive list of R packages on CRAN that can be used for spatial analysis.