Maps and spatial data

Abhijit Dasgupta, PhD

Maps

Toolsets

Visualizing spatial and geographic data is a specialized area on its own

R has increasing capabilities in this regard, and its increasingly mature. Some of the packages that we might need are

Data

- `sf`
- `sp`
- `raster`
- `spData`
- `rnaturalearthdata`

Visualization

- `ggplot` + `ggmap` + `geom_sf`
- `tmap`
- `leaflet`

> Several parts of this lecture are inspired by Chapter 8 of Geocomputation with R by Lovelace, Nowosad and Muenchow (2019), also available on Amazon

Toolset

We'll start of loading the following packages

```
library(ggplot2)
library(sf)
library(spData)
```

The **sf** package provides simple features access for R, and helps to store and process geographic data within the tidyverse framework, while linking to several geospatial packages that are standard in the geography world.

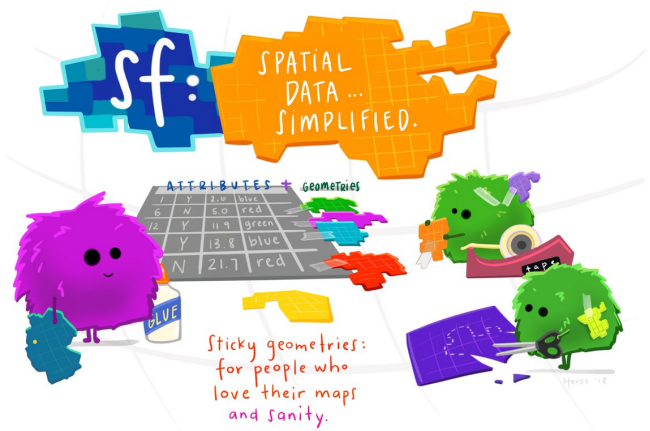


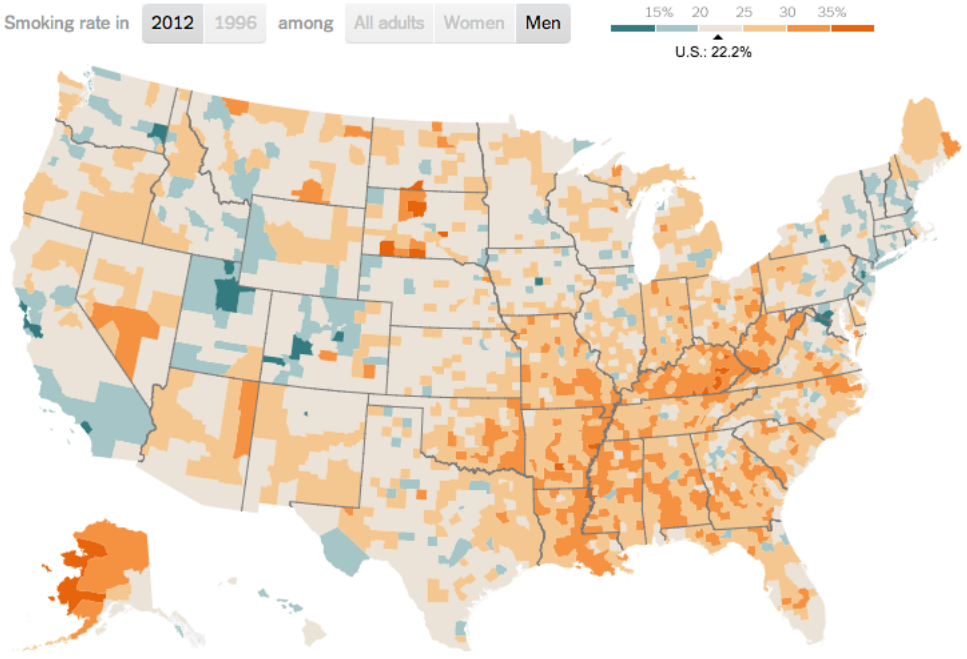
Illustration by Allison Horst

To use **sf** you may need to install some additional software. At the very least you will need to install the R packages **rgdal** and **rgeos**.

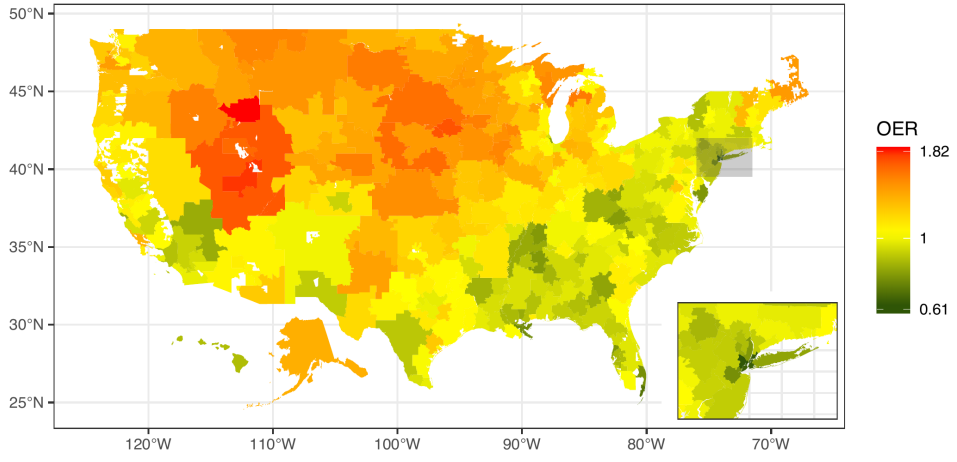
Additional information is available here

Chloropleth maps

Chloropleth maps are maps with some geometries filled in to signify levels of some variable.



Smoking rates in USA in 2012 (NY Times, March 24, 2014)



Observed to Expected Ratios (OERs) for Rates of Primary Total Knee Arthroplasty Among White Medicare Beneficiaries by Health Referral Region (Ward & Dasgupta, 2020)

A chloropleth of life expectancy

We'll start off with a world map

```
library(sf); library(spData)

ggplot(data = world) +
  geom_sf() # a special geometry for plotting maps
```

There are several ways of getting map geometries, which are specifications of polygons.

If you look at `world`, you'll see it's a data.frame, with one column named `geometry`. This column provides the shapes of the polygons, and what `geom_sf` looks for

A chloropleth of life expectancy

If you look at `world`, it also provides life expectancy estimates from 2014 (World Bank). The data set is tidy, with one row corresponding to one country. We'll use our known **ggplot2** way of filling things in.

```
ggplot(data = spData::world) +
  geom_sf(aes(fill = lifeExp)) # a special geometry for plotting maps
```

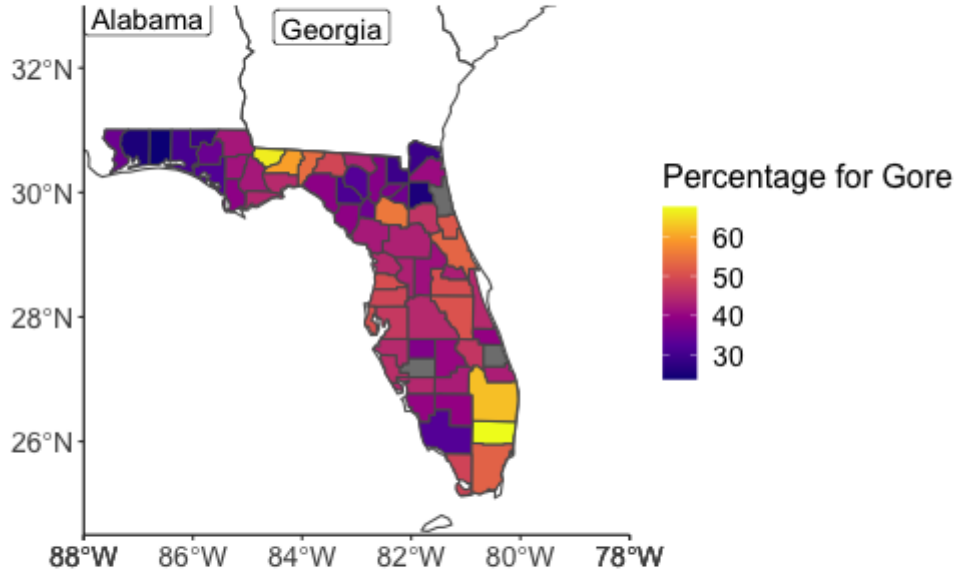
We need a more distinctive color palette.

A chloropleth of life expectancy

If you look at `world`, it also provides life expectancy estimates from 2014 (World Bank). The data set is tidy, with one row corresponding to one country. We'll use our known **ggplot2** way of filling things in.

```
ggplot(data = spData::world) +
  geom_sf(aes(fill = lifeExp)) + # a special geometry for plotting maps
  viridis::scale_fill_viridis('Life Exp', option='plasma',
                              trans='sqrt', labels = scales::label_number_si())
```

The electoral picture in Florida, 2000



Using tmap

Using tmap

The **tmap** package uses many synactical structures similar to **ggplot2**, but can be nicer in some ways

```
library(tmap)
tm_shape(spData::world) + tm_polygons(col = "lifeExp", style='jenks',
                                      title = 'Life expectancy')
```

It's more "publication-ready" by default

It makes some nice choices

tmap (interactive)

```
tmap_mode('view')
tm <- tm_shape(spData::world) + tm_polygons(col = 'pop', style='jenks',
                              palette='Reds', title='Population',
                              popup.vars = c('pop'), id='name_long')
```

Street maps

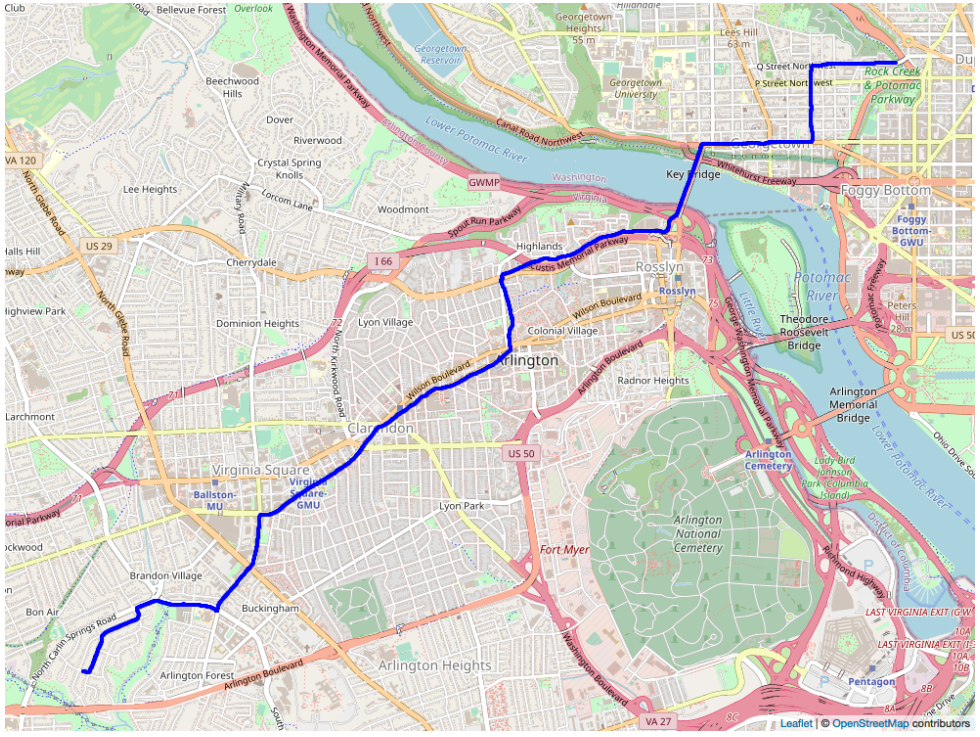
The easiest ways to overlay data on street maps is with the **leaflet** package.

```
library(leaflet)
library(leaflet.providers)
load(file.path(here::here('slides','lectures','data', 'exdata.rda')))
leaflet(gpx) %>% addTiles() %>% addCircleMarkers(~Longitude, ~Latitude, radius=1)
```

Street maps

You can also use the **mapview** package, which calls **leaflet** and has a bit more compact syntax

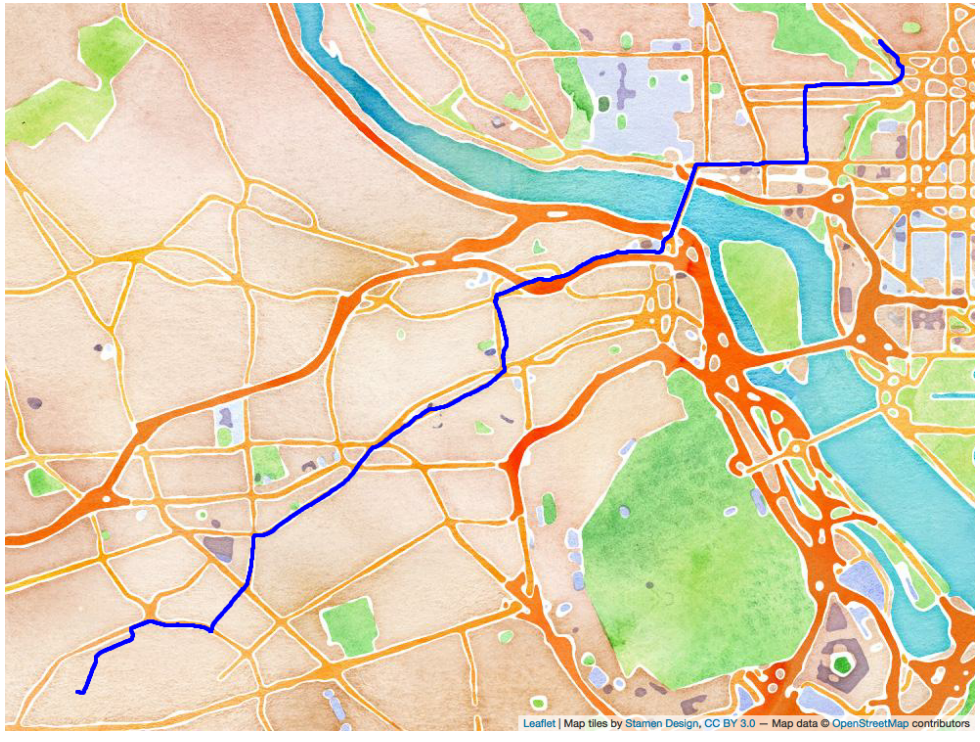
```
load(file.path(here::here('slides','lectures','data',
gpx <- sf::st_as_sf(gpx,
                coords=c("Longitude", "Latitude"),
                crs = 4267) # Need to make sf object
mapview::mapview(gpx, color='blue',
                 map.type = 'OpenStreetMap.Mapnik',
                 cex = 0.2, # size of points
                 legend=FALSE)
```



Street maps

You can also use the **mapview** package, which calls **leaflet** and has a bit more compact syntax

```
load(file.path(here::here('slides','lectures','data',
gpx <- st_as_sf(gpx,
                coords=c("Longitude", "Latitude"),
                crs = 4267) # Need to make sf object
mapview::mapview(gpx, color='blue',
                 map.type = 'Stamen.Watercolor',
                 cex = 0.2, # size of points
                 legend=FALSE)
```



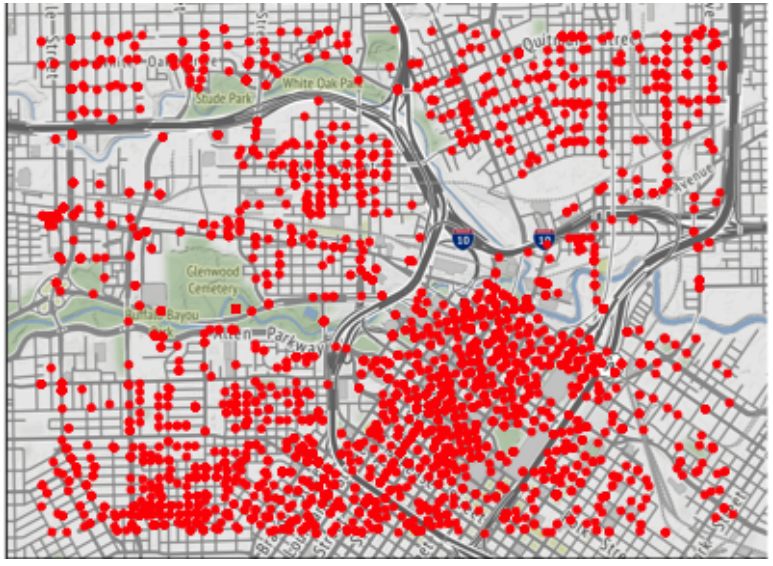
You can also have some stylistic fun with maps.

More possibilities at http://leaflet-extras.github.io/leaflet-providers/preview/

Dot density maps

```
library(ggmap)
crime1 <- crime %>%
  filter(between(crime$lon, -95.4, -95.34) &
           between(crime$lat, 29.746, 29.784))

qmplot(lon, lat, data=crime1, maptype='toner-lite',
       color = I('red'))
```

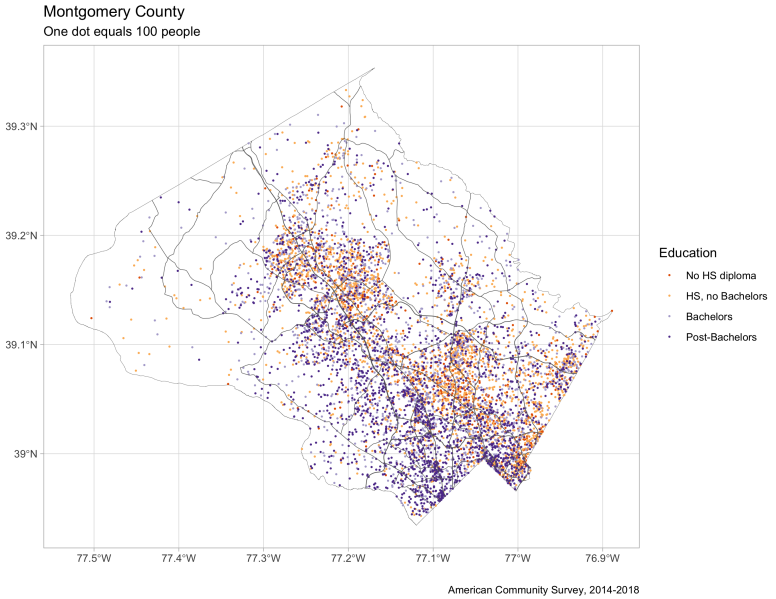


ggmap was built for Google Maps

Google Maps requires a credit card now

Better option is Stamen Maps, which uses OpenStreetMap data

Dot density maps

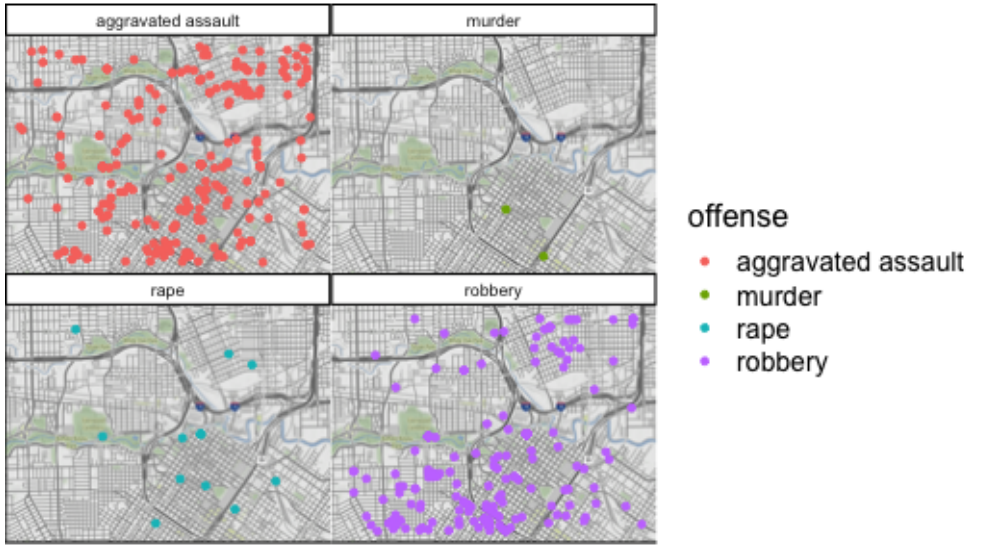


Code is available here

Based on this blog by Tarak Shah

Facetted maps

```
library(ggmap)
crime1 <- crime1 %>%
  filter(!(offense %in% c('auto theft','theft',
                          'burglary')))
qmplot(lon, lat, data=crime1,
       maptype='toner-background',
       color = offense)+
  facet_wrap(~offense)
```



Facetted maps

```
library(tmap)
world1 <- world %>%
  filter(continent %in% c('Europe','Asia',
                          'North America',
                          'South America')) %>%
  mutate(continent = fct_reorder(continent,
                                 lifeExp, na.rm=T))

tm_shape(world1)+tm_polygons(col='lifeExp') +
  tm_facets(by='continent', ncol=2)
```