Visualizing Spatial Data

Angela Li, Center for Spatial Data Science

January 21, 2020

Outline

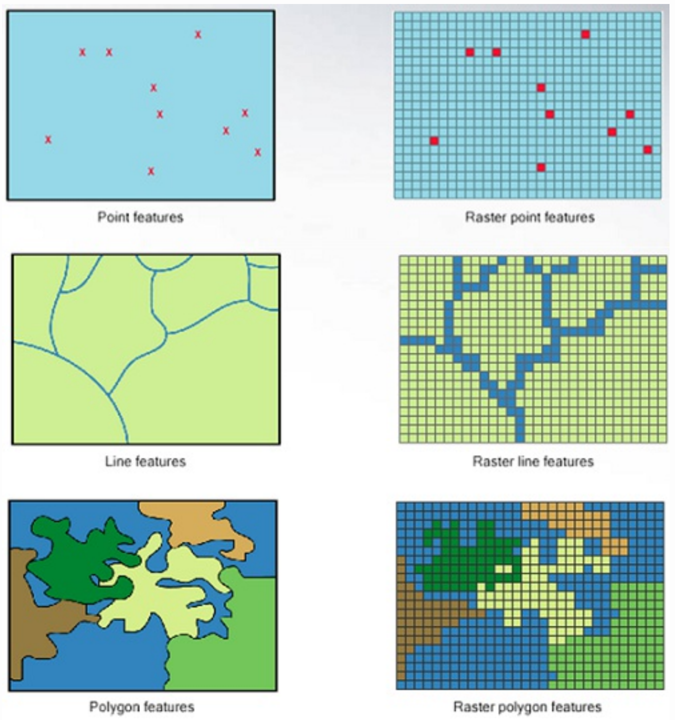
1. What Is Spatial Data?
2. Reading/Writing Spatial Data
3. Spatial Data Visualization
4. Spatial Data Transformation / Geoprocessing (if time)

What is Spatial Data?

Vector vs. Raster Data

Spatial data comes in two main formats: **vector** and **raster** data.

Vector data often represents discrete objects, while raster data often represents continuous surfaces.



Vector Data: Geocoding

Sometimes people give me a list of addresses and ask me to map it. I can't do that unless I have the **latitude and longitude**, so I'll need to **geocode** the addresses. For states (polygons), I'll need to find a **geospatial boundary file** with that information (Google around!)

```
bikeshare_addresses <- read.csv("data/bike_addresses.csv")
head(bikeshare_addresses)
```

```
##    ID                             ADDRESS
## 1 550                        McLean Metro
## 2 551                 Trinidad Rec Center
## 3 552                 Rosedale Rec Center
## 4 553                      11th & C St SE
## 5 554 New Hampshire & Gallatin St NW
## 6 555                United Medical Center
```

Geocoding in R

There are a few options, which usually limit you to 2000 queries. Here's one nice one that uses OpenStreetMap.

```
tmaptools::geocode_OSM("McLean Metro")
```

```
## $query
## [1] "McLean Metro"
## 
## $coords
##         x          y 
## -77.20791  38.92379 
## 
## $bbox
##      xmin      ymin      xmax      ymax 
## -77.20796  38.92374 -77.20786  38.92384
```

Geocoding will give me latitude and longitude for points

```r
bikeshare_latlon <- read.csv("data/bike_addresses_latlon.csv")
head(bikeshare_latlon)
```

```
##    ID                        ADDRESS LATITUDE LONGITUDE
## 1 550                   McLean Metro 38.92400 -77.20813
## 2 551           Trinidad Rec Center 38.90630 -76.98322
## 3 552            Rosedale Rec Center 38.89781 -76.97963
## 4 553                 11th & C St SE 38.88591 -76.99148
## 5 554 New Hampshire & Gallatin St NW 38.95160 -77.01281
## 6 555          United Medical Center 38.83574 -76.98314
##                                  geometry
## 1 c(-77.2081293496828, 38.9240097850969)
## 2     c(-76.983223279, 38.9063067901138)
## 3 c(-76.9796362771458, 38.8978157882255)
## 4  c(-76.9914782802471, 38.885915785567)
## 5  c(-77.012810290596, 38.9516037980159)
## 6 c(-76.9831462746571, 38.8357447756218)
```

Read in spatial data

To read in spatial data, use the `st_read()` or `read_sf()` function from the `sf` package:

```
library(sf)
```

```
## Linking to GEOS 3.7.2, GDAL 2.4.2, PROJ 5.2.0
```

```
st_read("data/dc_wards.shp")
```

```
## Reading layer `dc_wards' from data source `/Users/angela/Desktop/R-Projects/aaas-
## Simple feature collection with 8 features and 82 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: -77.1198 ymin: 38.79164 xmax: -76.90915 ymax: 38.99597
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
```

```
read_sf("data/dc_wards.shp") # if you like tidy data
```

```
## Simple feature collection with 8 features and 82 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: -77.1198 ymin: 38.79164 xmax: -76.90915 ymax: 38.99597
## epsg (SRID):    4326
```

Writing spatial data

To write spatial data, use `st_write()` or `write_sf()`:

```
st_write(dc_wards, "data-output/dc_wards.shp")
write_sf(dc_wards, "data-output/dc_wards.shp")
```

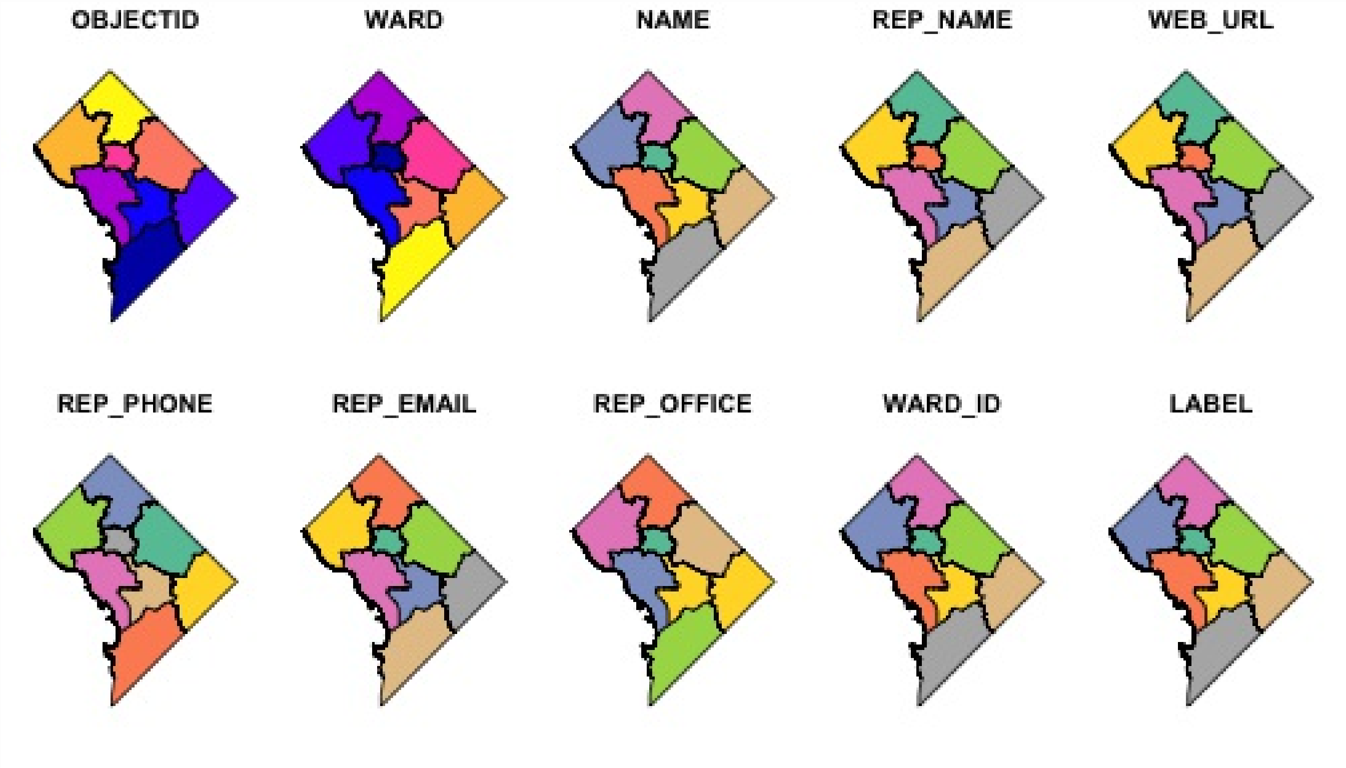
Spatial Data Visualization

Spatial Data Visualization

- Base plotting
- tmap
- ggplot2
- mapview

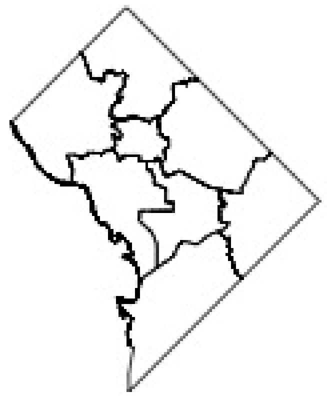
Base Plotting

```
plot(dc_wards)
```



Plot just the geometry

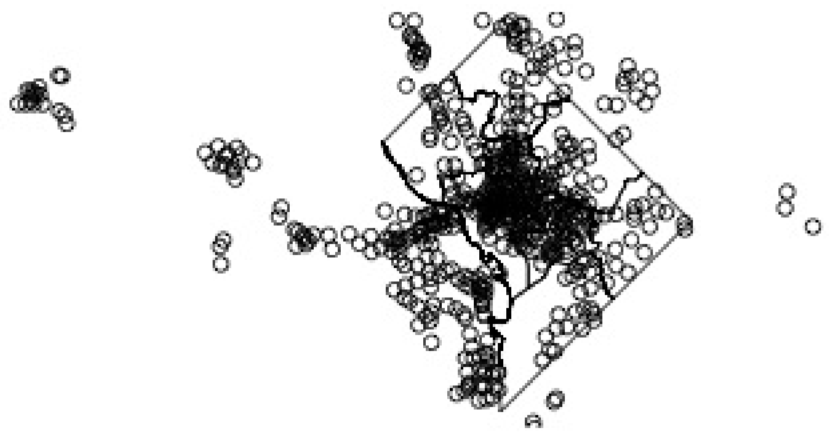
```
plot(st_geometry(dc_wards))
```



Plot more than one layer

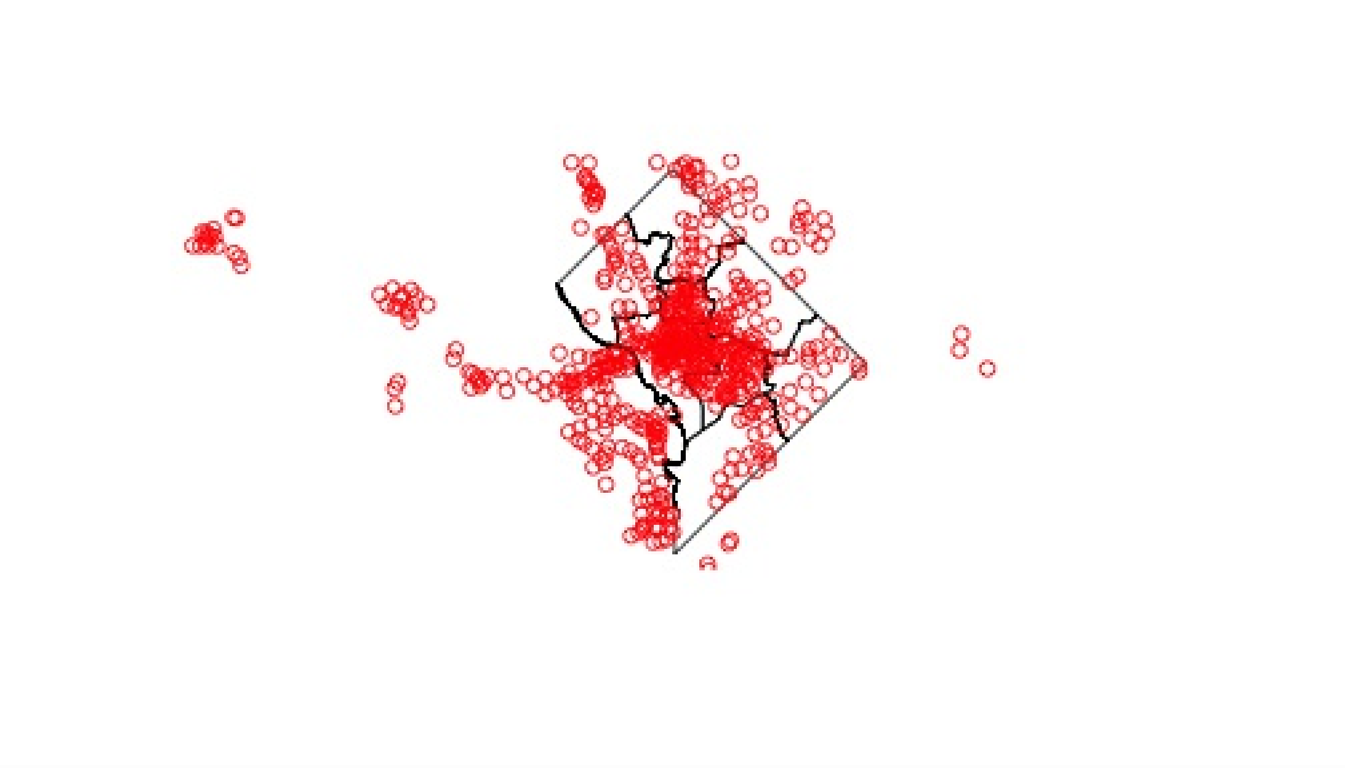
```
bikeshare_locations <- read_sf("data/bikeshare_locations.shp")
```

```
plot(st_geometry(dc_wards))
plot(st_geometry(bikeshare_locations), add = TRUE)
```



Change the color

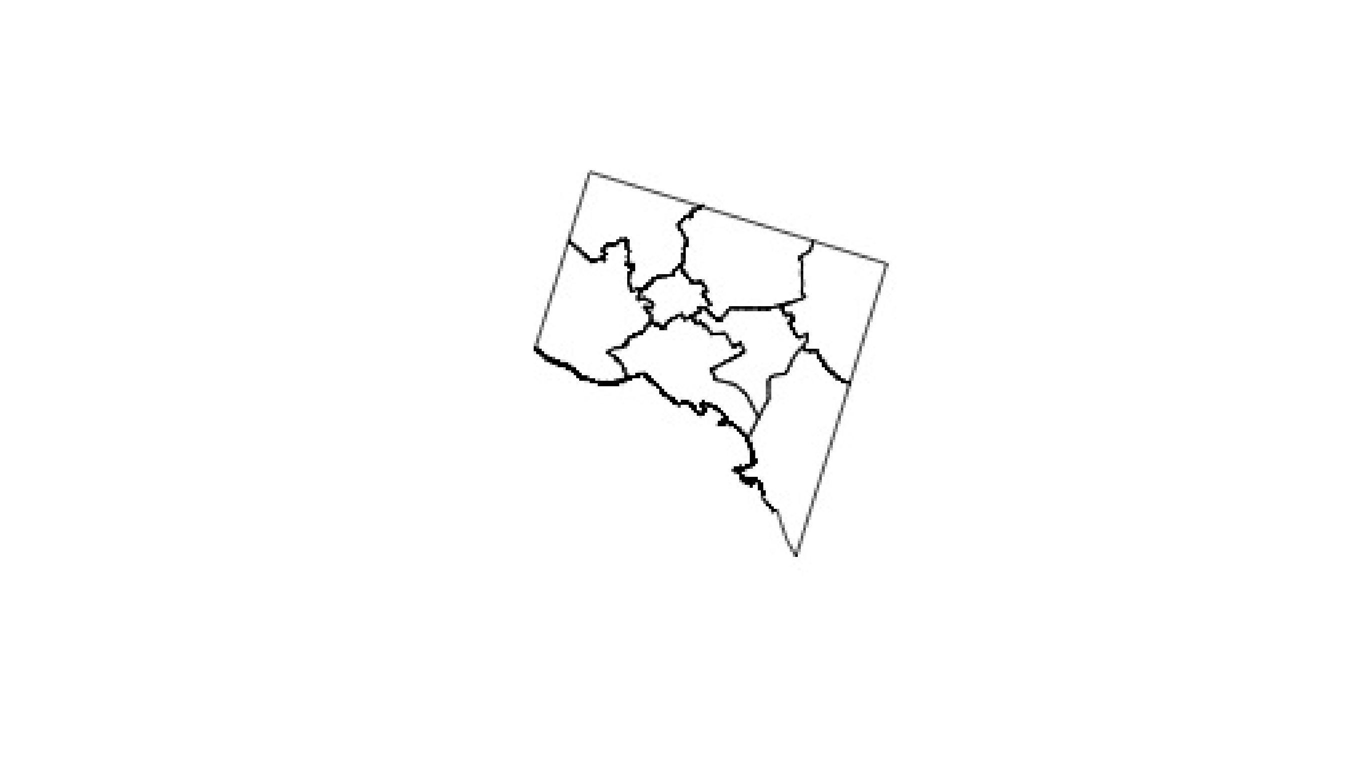
```
plot(st_geometry(dc_wards))
plot(st_geometry(bikeshare_locations), add = TRUE, col = "red")
```



Common issues: Projections and Coordinate Reference Systems

If your map layers won't plot on top of each other, you need to check that they are in the same map projection.

```
plot(st_geometry(dc_wards_proj))
plot(st_geometry(bikeshare_locations), add = TRUE)
```



Check the projection

These aren't the same!

```
st_crs(dc_wards_proj)
```

```
## Coordinate Reference System:
##   EPSG: 6654
##   proj4string: "+proj=utm +zone=11 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +v
```

```
st_crs(bikeshare_locations)
```

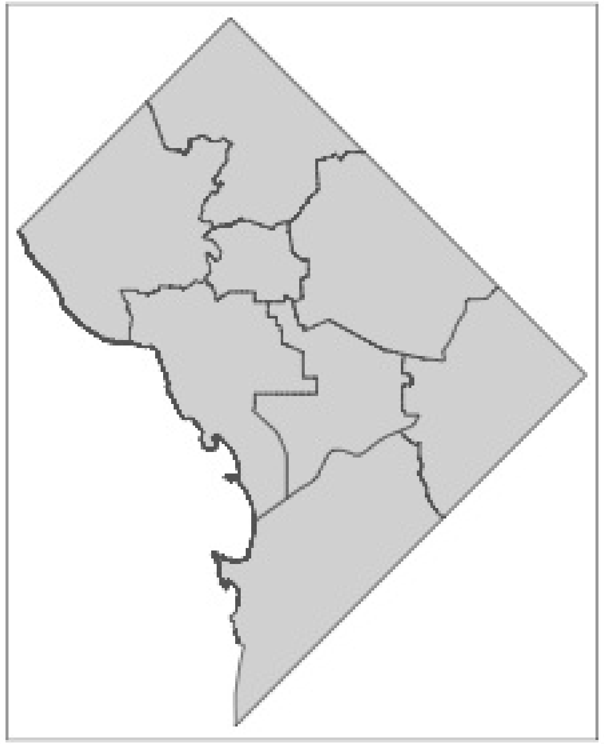
```
## Coordinate Reference System:
##   EPSG: 4326
##   proj4string: "+proj=longlat +datum=WGS84 +no_defs"
```

Project the data

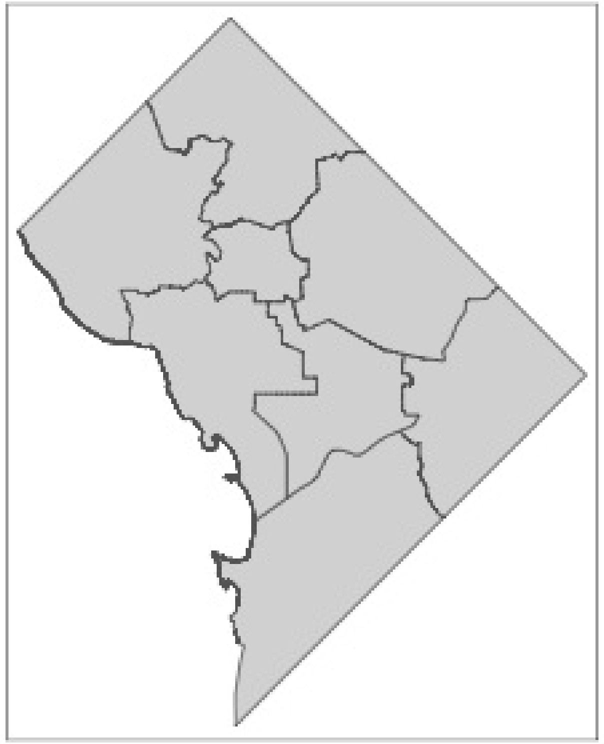
```r
bikeshare_locations <- st_transform(bikeshare_locations, 6654)
```

Other R map package options

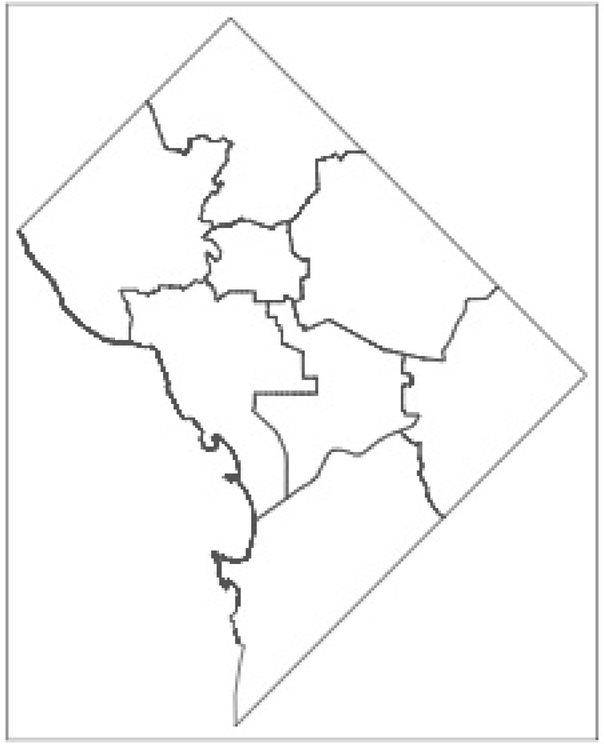
```
library(tmap)
qtm(dc_wards)
```



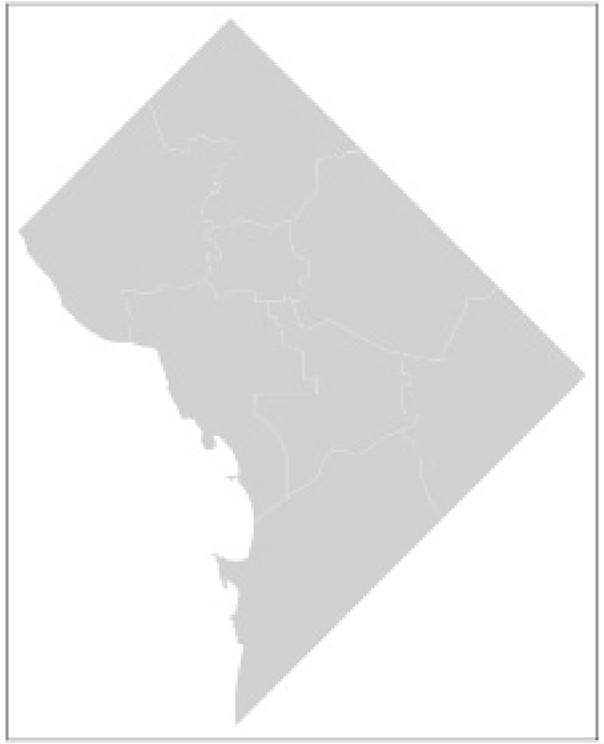
```
tm_shape(dc_wards) +
  tm_polygons()
```



```
tm_shape(dc_wards) +
  tm_borders()
```



```
tm_shape(dc_wards) +
  tm_fill()
```

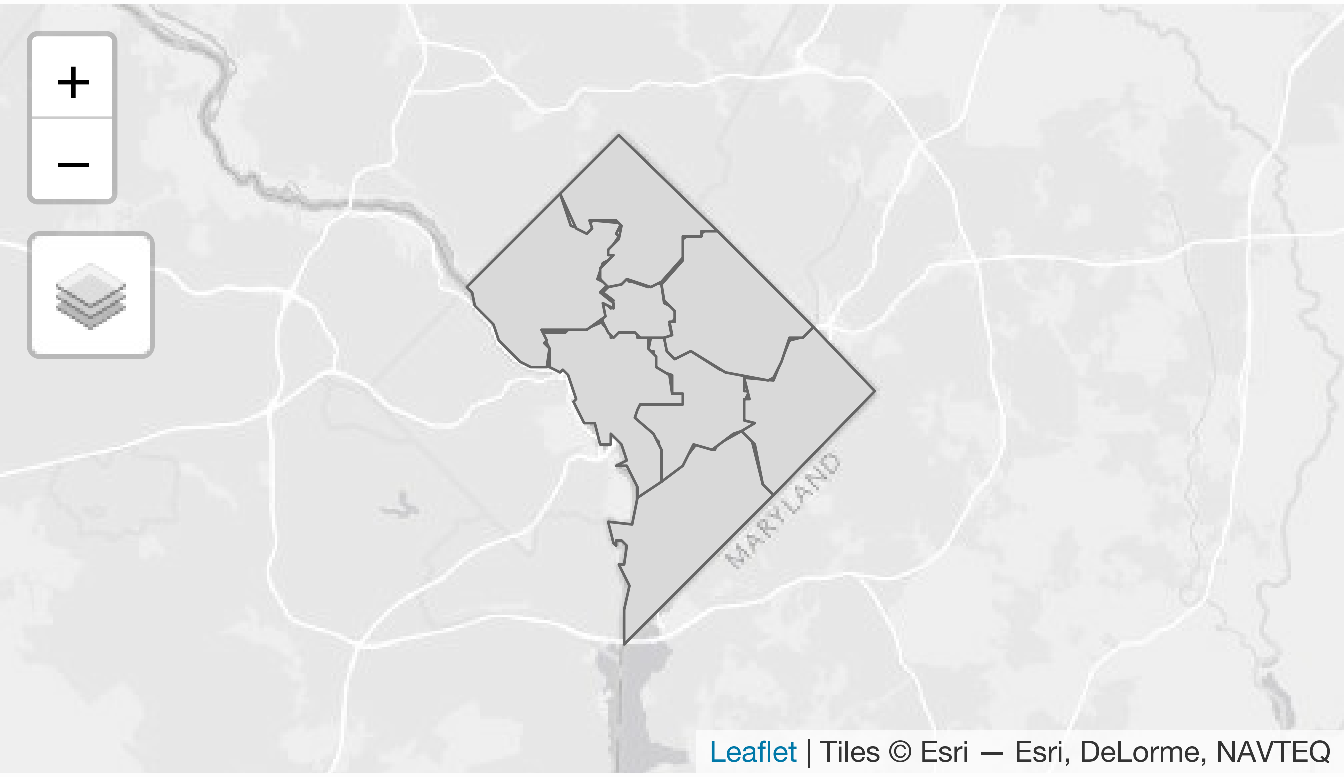


Change to an interactive mode

```
tmap_mode("view")
```

```
## tmap mode set to interactive viewing
```

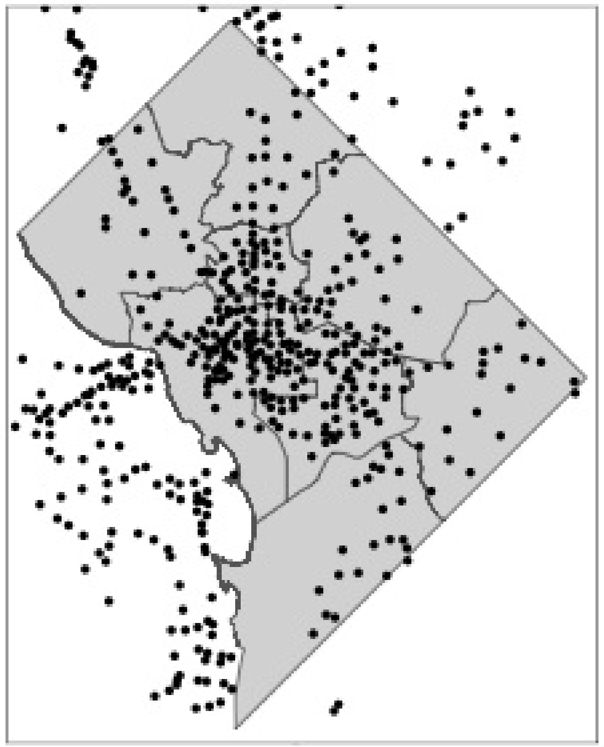
```
tm_shape(dc_wards) +
  tm_polygons()
```



```r
tmap_mode("plot")
```

```
## tmap mode set to plotting
```

```r
tm_shape(dc_wards) +
  tm_polygons() +
  tm_shape(bikeshare_locations) +
  tm_dots(size = 0.1)
```

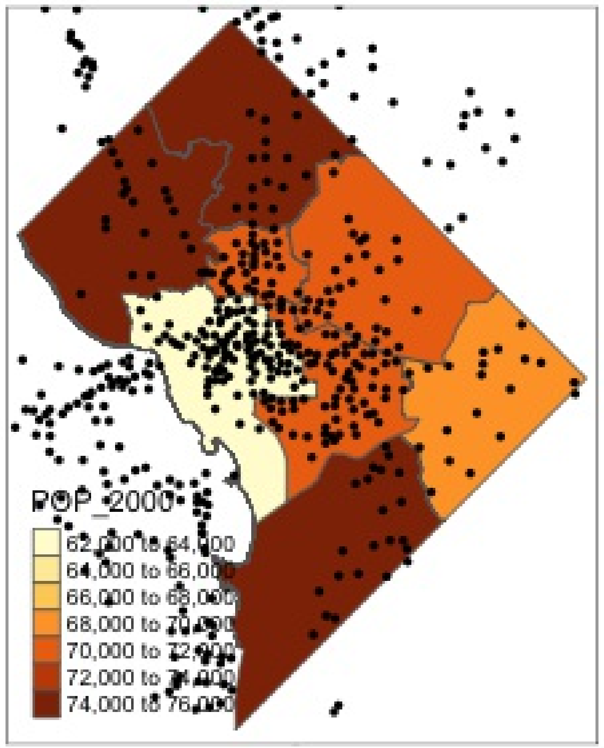


More complicated maps possible

```
tmap_mode("plot")
```

```
## tmap mode set to plotting
```

```
tm_shape(dc_wards) +
  tm_polygons("POP_2000") +
  tm_shape(bikeshare_locations) +
  tm_dots(size = 0.1)
```



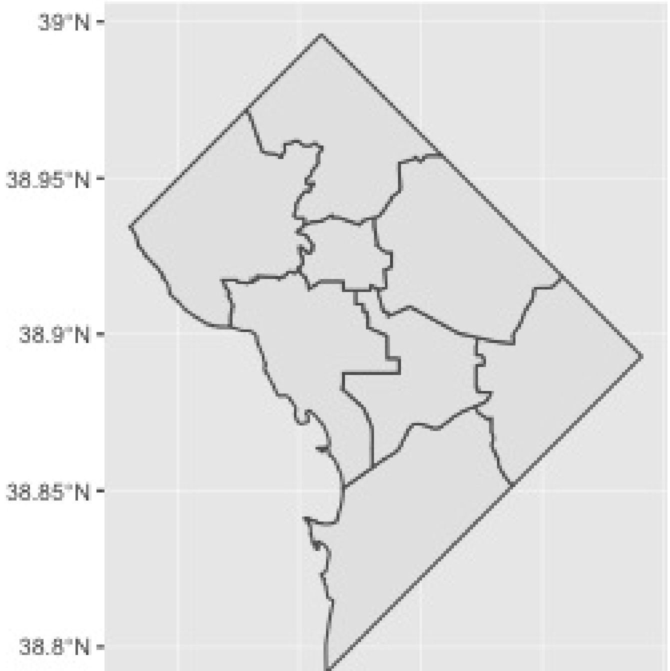
ggplot2 is also an option

More on this at "Drawing beautiful maps programmatically with R, sf and ggplot2".

```
library(ggplot2)
```

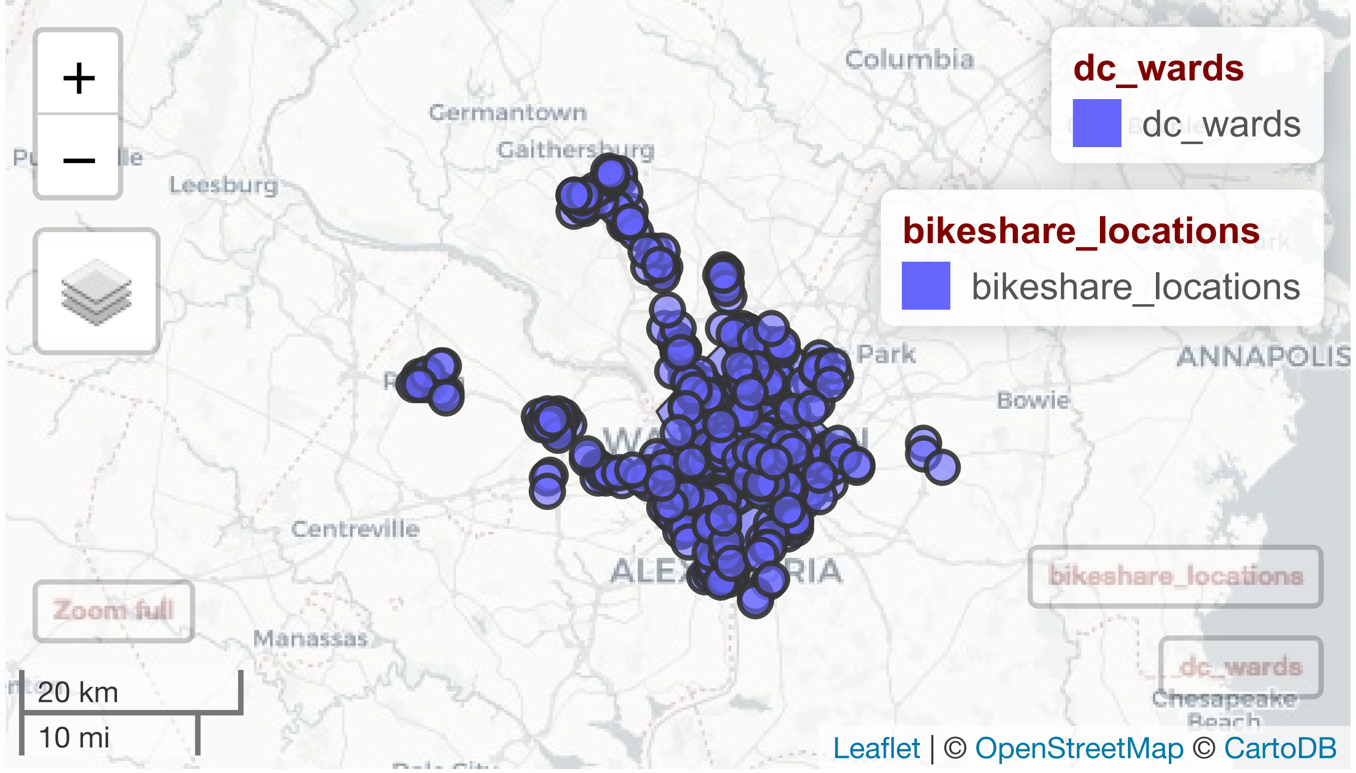
```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from 
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
ggplot(data = dc_wards) +
  geom_sf()
```



For an interactive map experience similar to a GIS, check out mapview

```
library(mapview)
mapview(dc_wards) +
  mapview(bikeshare_locations)
```



Spatial Data Transformation / Geoprocessing

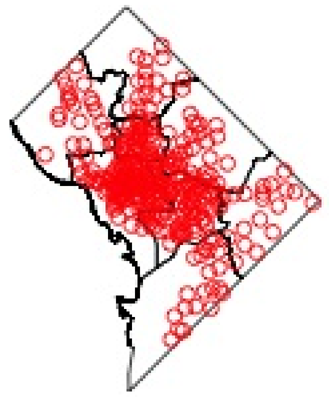
1. Crop locations to DC Wards
2. Count how many in each ward
3. Calculate bikeshare density by population

Crop locations to DC Wards

```
bikes_in_dc <- st_intersection(bikeshare_locations, dc_wards)
```

```
## although coordinates are longitude/latitude, st_intersection assumes that they are
```

```
plot(st_geometry(dc_wards))
plot(st_geometry(bikes_in_dc), col = "red", add = TRUE)
```



Count up number of bikes in wards

Two ways to do this...

1. Attribute join

```
st_intersection(dc_wards, bikeshare_locations) %>%
  dplyr::count(WARD) %>%
  st_drop_geometry() %>%
  dplyr::right_join(dc_wards)
```

```
## although coordinates are longitude/latitude, st_intersection assumes that they ar
```

```
## Joining, by = "WARD"
```

```
## # A tibble: 8 x 84
##    WARD     n OBJECTID NAME  REP_NAME WEB_URL REP_PHONE REP_EMAIL
##   <dbl> <int>    <dbl> <chr> <chr>    <chr>   <chr>     <chr>
## 1     8    22        1 Ward… Trayon … http:/… (202) 72… twhite@d…
## 2     6    68        2 Ward… Charles… http:/… (202) 72… callen@d…
## 3     7    19        3 Ward… Vincent… http:/… (202) 72… vgray@dc…
## 4     2    93        4 Ward… Jack Ev… http:/… (202) 72… jevans@d…
## 5     1    33        5 Ward… Brianne… http:/… (202) 72… bnadeau@…
## 6     5    32        6 Ward… Kenyan … http:/… (202) 72… kmcduffi…
## 7     3    21        7 Ward… Mary M.… http:/… (202) 72… mcheh@dc…
## 8     4    18        8 Ward… Brandon… http:/… (202) 72… btodd@dc…
## # … with 76 more variables: REP_OFFICE <chr>, WARD_ID <chr>, LABEL <chr>
```

Count up number of bikes in wards

Less steps -

1. Spatial join (use the geometry to perform a join):

```
st_join(dc_wards, bikeshare_locations) %>%
  dplyr::count(WARD)
```

```
## although coordinates are longitude/latitude, st_intersects assumes that they are

## Simple feature collection with 8 features and 2 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: -77.1198 ymin: 38.79164 xmax: -76.90915 ymax: 38.99597
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
## # A tibble: 8 x 3
##    WARD     n                                                       geometry
## * <dbl> <int>                                                  <POLYGON [°]>
## 1     1    33 ((-77.03523 38.93743, -77.0348 38.93743, -77.03436 38.93743,…
## 2     2    93 ((-77.04946 38.91999, -77.04919 38.91954, -77.04918 38.91952…
## 3     3    21 ((-77.05808 38.95676, -77.05807 38.95672, -77.05805 38.95672…
## 4     4    18 ((-77.04097 38.99597, -76.99144 38.9573, -76.99163 38.95726,…
## 5     5    32 ((-76.99144 38.9573, -76.94186 38.91854, -76.942 38.91842, -…
## 6     6    68 ((-77.0179 38.9141, -77.01786 38.914, -77.01784 38.91393, -7…
```

Find bike density in each ward

```
st_join(dc_wards, bikeshare_locations) %>%
  dplyr::count(WARD, POP_2011_2)
```

```
## although coordinates are longitude/latitude, st_intersects assumes that they are

## Simple feature collection with 8 features and 3 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: -77.1198 ymin: 38.79164 xmax: -76.90915 ymax: 38.99597
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
## # A tibble: 8 x 4
##    WARD POP_2011_2     n                                                 geometry
## * <dbl>      <dbl> <int>                                           <POLYGON [°]>
## 1     1      82859    33 ((-77.03523 38.93743, -77.0348 38.93743, -77.0343…
## 2     2      77645    93 ((-77.04946 38.91999, -77.04919 38.91954, -77.049…
## 3     3      83152    21 ((-77.05808 38.95676, -77.05807 38.95672, -77.058…
## 4     4      83066    18 ((-77.04097 38.99597, -76.99144 38.9573, -76.9916…
## 5     5      82049    32 ((-76.99144 38.9573, -76.94186 38.91854, -76.942 …
## 6     6      84290    68 ((-77.0179 38.9141, -77.01786 38.914, -77.01784 3…
## 7     7      73290    19 ((-76.94186 38.91854, -76.90915 38.89293, -76.961…
## 8     8      81133    22 ((-76.97229 38.87286, -76.97223 38.87273, -76.972…
```

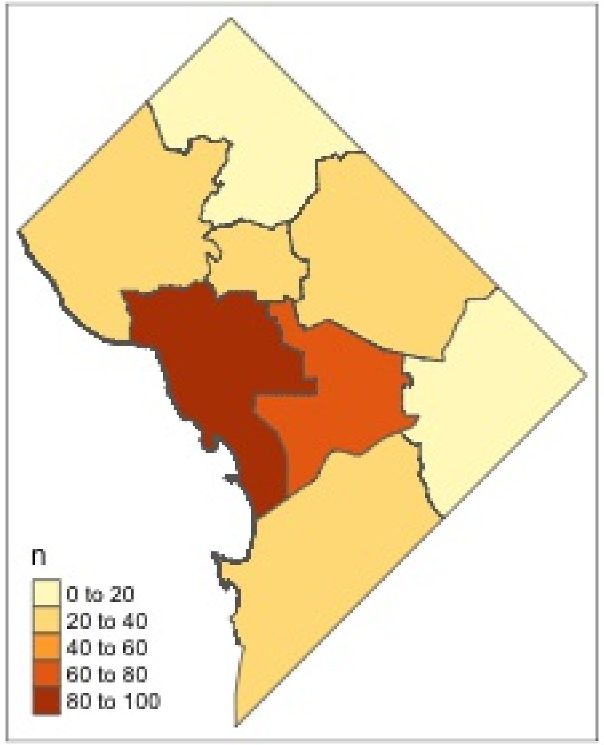
```
st_join(dc_wards, bikeshare_locations) %>%
```

Make a map!

```
bikeshare_density_per_ward <-
st_join(dc_wards, bikeshare_locations) %>%
  dplyr::count(WARD, POP_2011_2) %>%
  dplyr::mutate(bikeshare_density = n / POP_2011_2)
```

```
## although coordinates are longitude/latitude, st_intersects assumes that they are
```

```
tm_shape(bikeshare_density_per_ward) +
  tm_polygons("n")
```



Resources

- Tutorials developed by my research center
- Geocomputation with R
- Spatial Data Science
- Data Carpentry Geospatial Lesson (focuses on raster data)