SHINY WEB APPS

Day 2

Heike Hofmann

SETUP

- Materials (pdf, code) are located in GitHub:
  https://github.com/DSPG-ISU/Training
  Clone the directory onto your local machine

- Open an RStudio session
  (on Rivanna, locally or RStudio cloud: https://rstudio.cloud/project/1424983)

OUTLINE

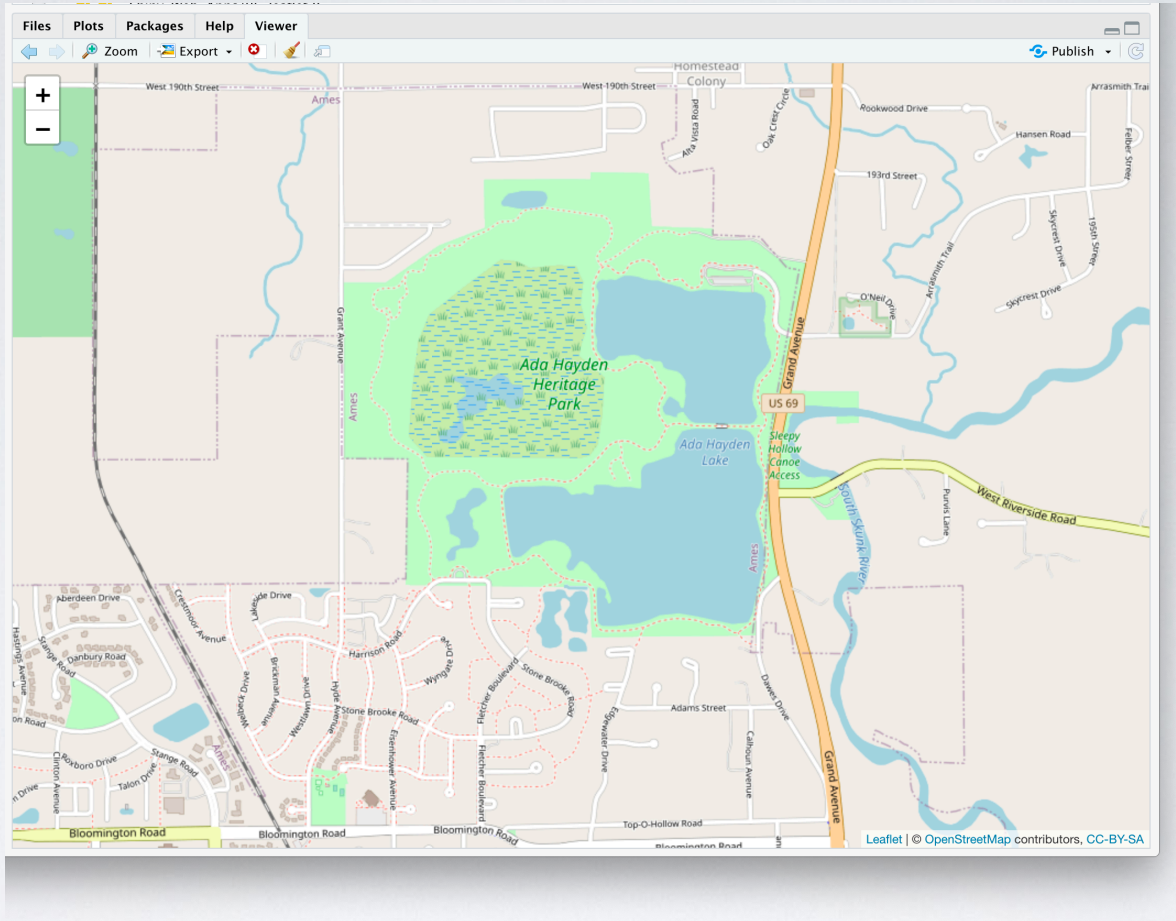
- **Day 1**
  - Getting ready
  - Inputs & Outputs
  - Elements of Reactivity
  - Layouts
  - Some practice

- **Day 2**
  - leaflets in shiny
  - Work on shiny apps for the DSPG projects

CHECKING YOUR SETUP

- Open the file 00_setup.R in RStudio, select all lines and run them. This
  - installs packages as needed and
  - activates packages.

- Don't forget about the command `devtools::session_info()` when something goes wrong



LEAFLET MAPS IN SHINY APPS

LEAFLET

- There is an R package **`leaflet`**, https://rstudio.github.io/leaflet/
- Simplest form: **`leaflet() %>% addTiles()`**
- The pipe operator from the tidyverse is incorporating naturally in the workflow
- Most leaflet functions request a map object as input and export an enhanced map object

YOUR TURN

- Load the leaflet package into your R session.
- Create a simple leaflet map
- `setView` or `fitBounds` allow you to set a specific location for the map and set the zoom level
- Read up on these two functions using the R help
- Use the functions to get a map of your university's campus

How do you get geographic latitude and longitude of a place? Google for it!

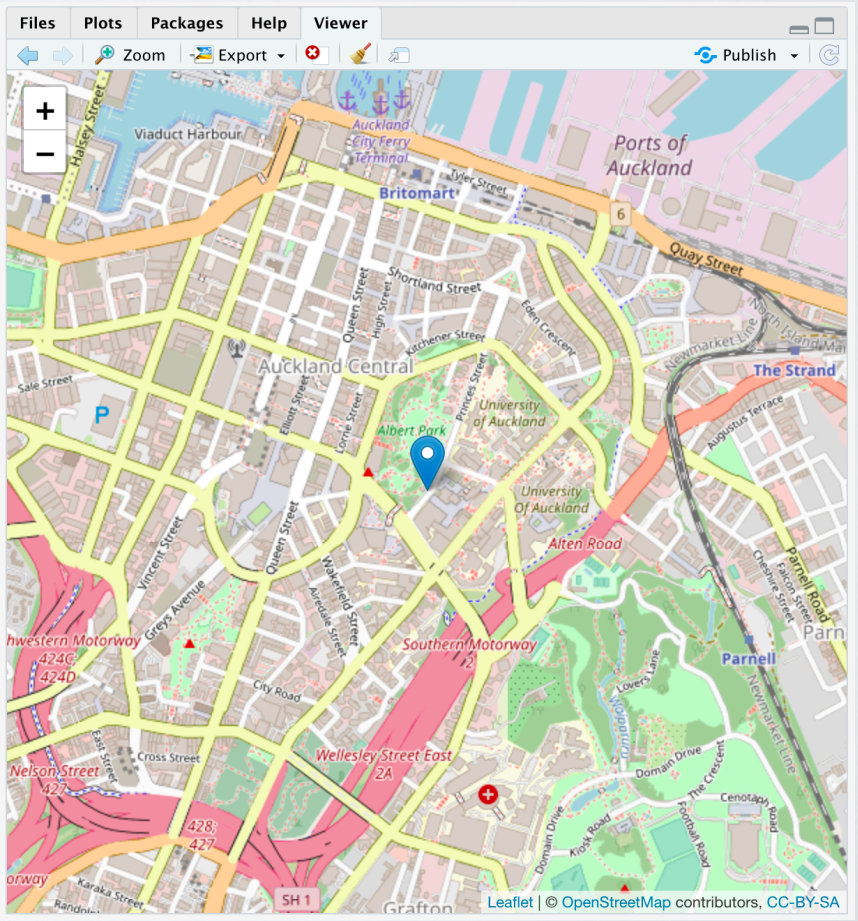
ADDING POINT LAYERS

- Similar to adding layers in ggplot2

```
m <- leaflet() %>%
  addTiles() %>%
  addMarkers(
    lng=174.768, lat=-36.852,
    popup="The birthplace of R")
m
```

- Markers could be piped in as data into leaflet

- `addCircleMarkers()` adds circles instead of pins



CONSIDERATIONS

- Make sure that your data is prominently visible and not dominated by the choice of provider tiles

- `names(providers) # list of possible options`

- ```
  leaflet() %>%
      addProviderTiles(provider = "CartoDB")
  leaflet() %>%
      addProviderTiles(
         provider = "CartoDB.PositronNoLabels")
  ```
  ```

AN EXAMPLE

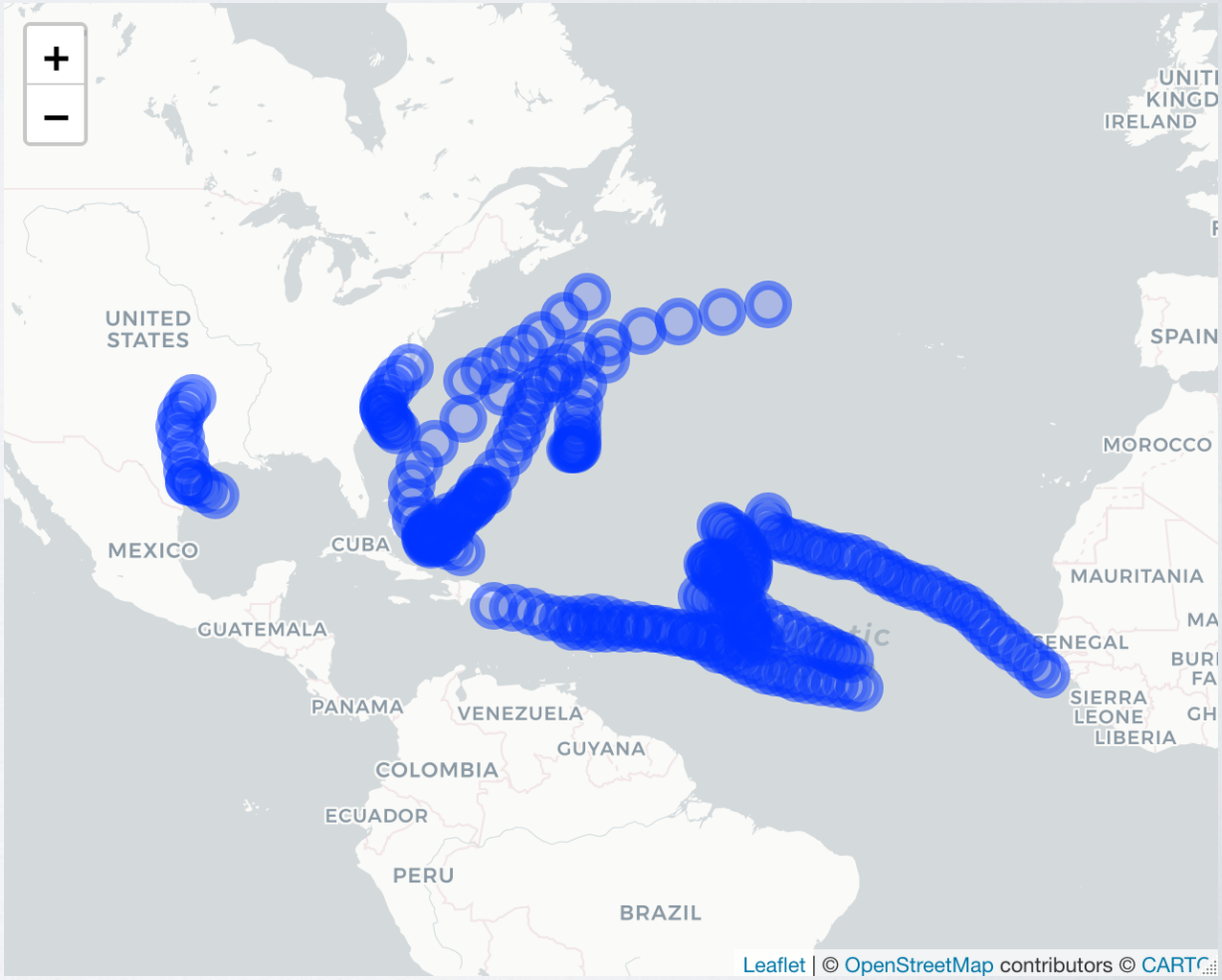
- Dataset: **`storms`** from dplyr package, named storms and tracks from 1975 to 2015
- `?storms`
- Follow along in `07_markers.R`

```
# let's focus on one year
st15 <- storms %>% filter(year == 2015)


leaflet() %>%
  addProviderTiles("CartoDB") %>%
  addCircleMarkers(lng = st15$long, lat = st15$lat)
```

Storm tracks become visible. We will look into giving each storm a different color.

First: let's set up the data pipeline better

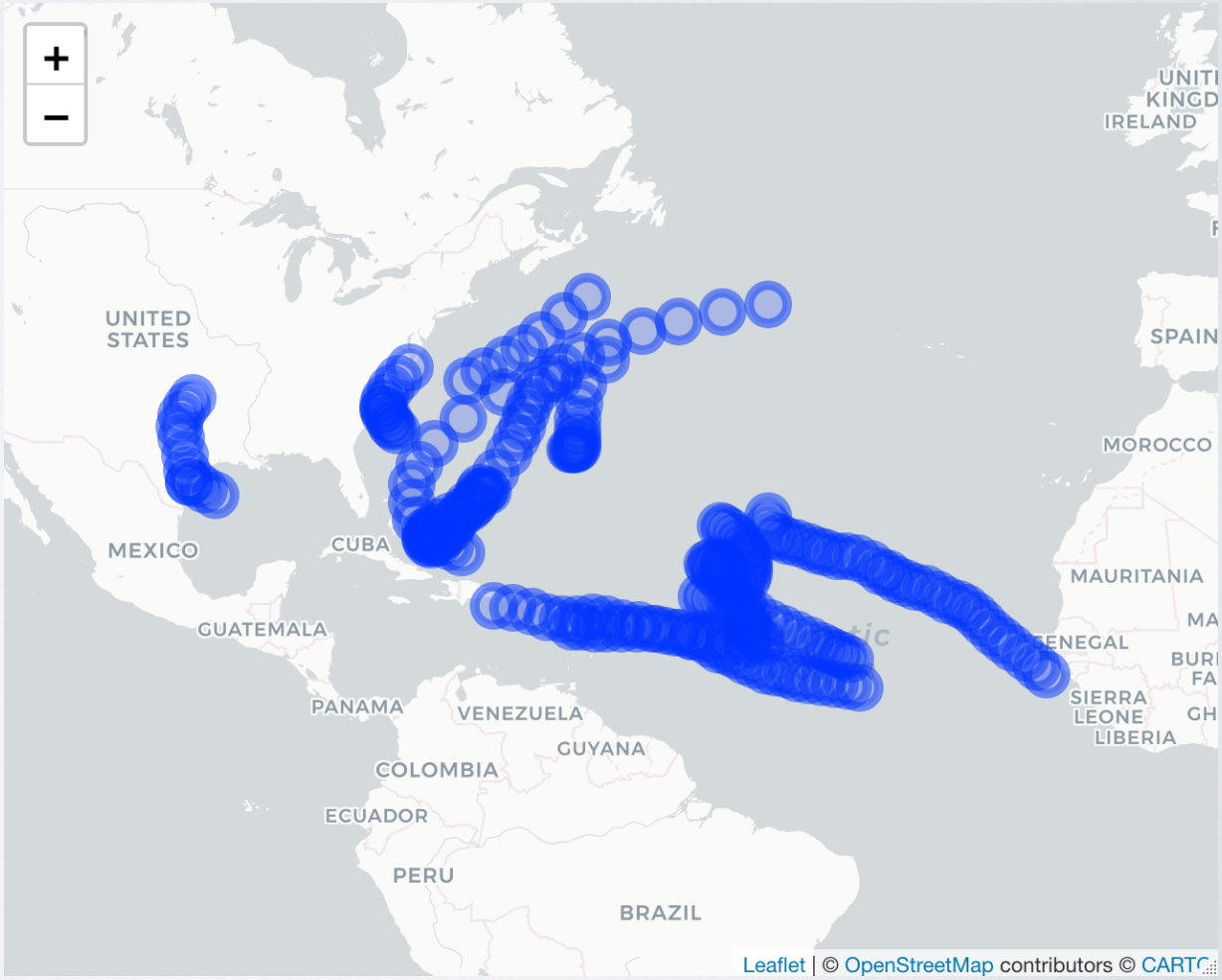


```
# let's focus on one year
st15 <- storms %>% filter(year == 2015)

st15 %>%
  leaflet() %>%
    addProviderTiles("CartoDB") %>%
    addCircleMarkers(lng = ~long, lat = ~lat)
```

We can pipe a dataset into a map - makes accessing variables easier.

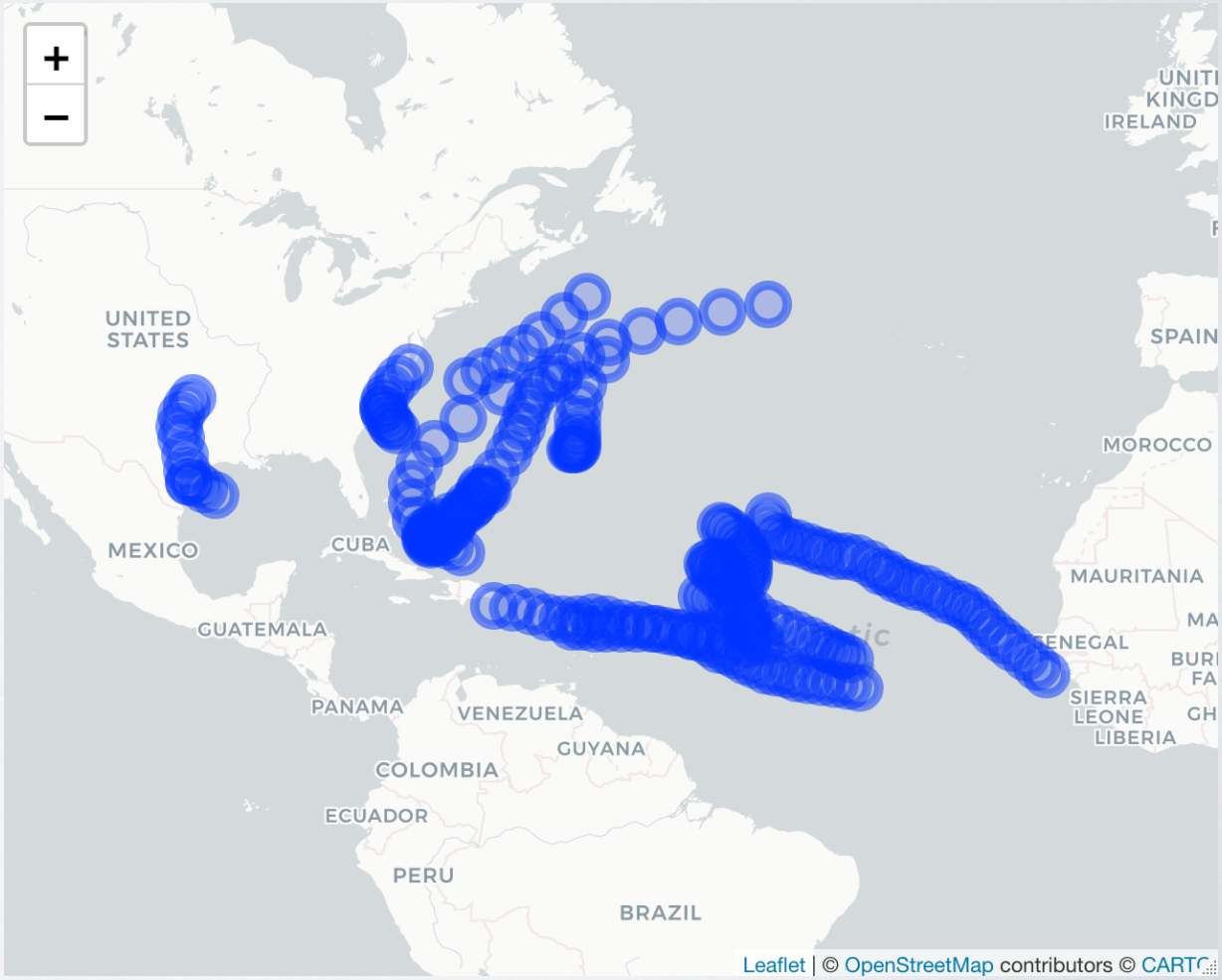
Note the use of the **~**



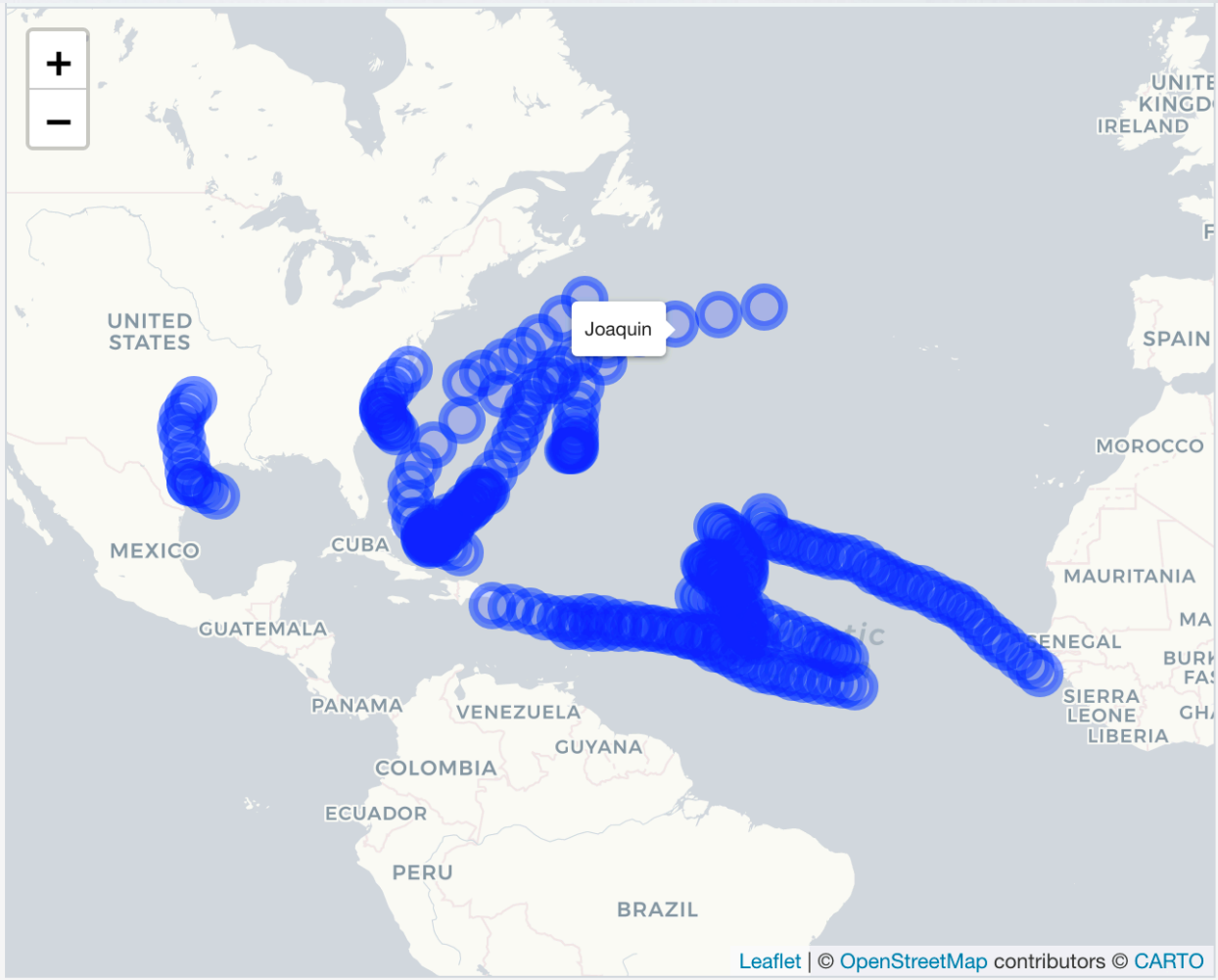
```
storms %>%
  filter(year == 2015) %>%
  leaflet() %>%
    addProviderTiles("CartoDB") %>%
    addCircleMarkers(lng = ~long, lat = ~lat)
```

We can also combine tidyverse data wrangling with the calls to leaflet

Doesn't this code look much prettier? 😊



```
storms %>%
  filter(year == 2015) %>%
  leaflet() %>%
    addProviderTiles("CartoDB") %>%
    addCircleMarkers(lng = ~long, lat = ~lat,
                     popup=~name)
```



YOUR TURN

- Use the **`storms`** data from the dplyr package to draw storm tracks of one of the years on a leaflet map (you pick the year!)
- Use **`label`** to show the name of a storm when you hover over a circle.
- Look into the help for **`addCircleMarkers`** to figure out how to change the size of the circles and their colors.

CREATING A COLOR PALETTE FOR A FACTOR VARIABLE

- Creating a color palette happens in two steps:
  - Create a color palette with colors and levels

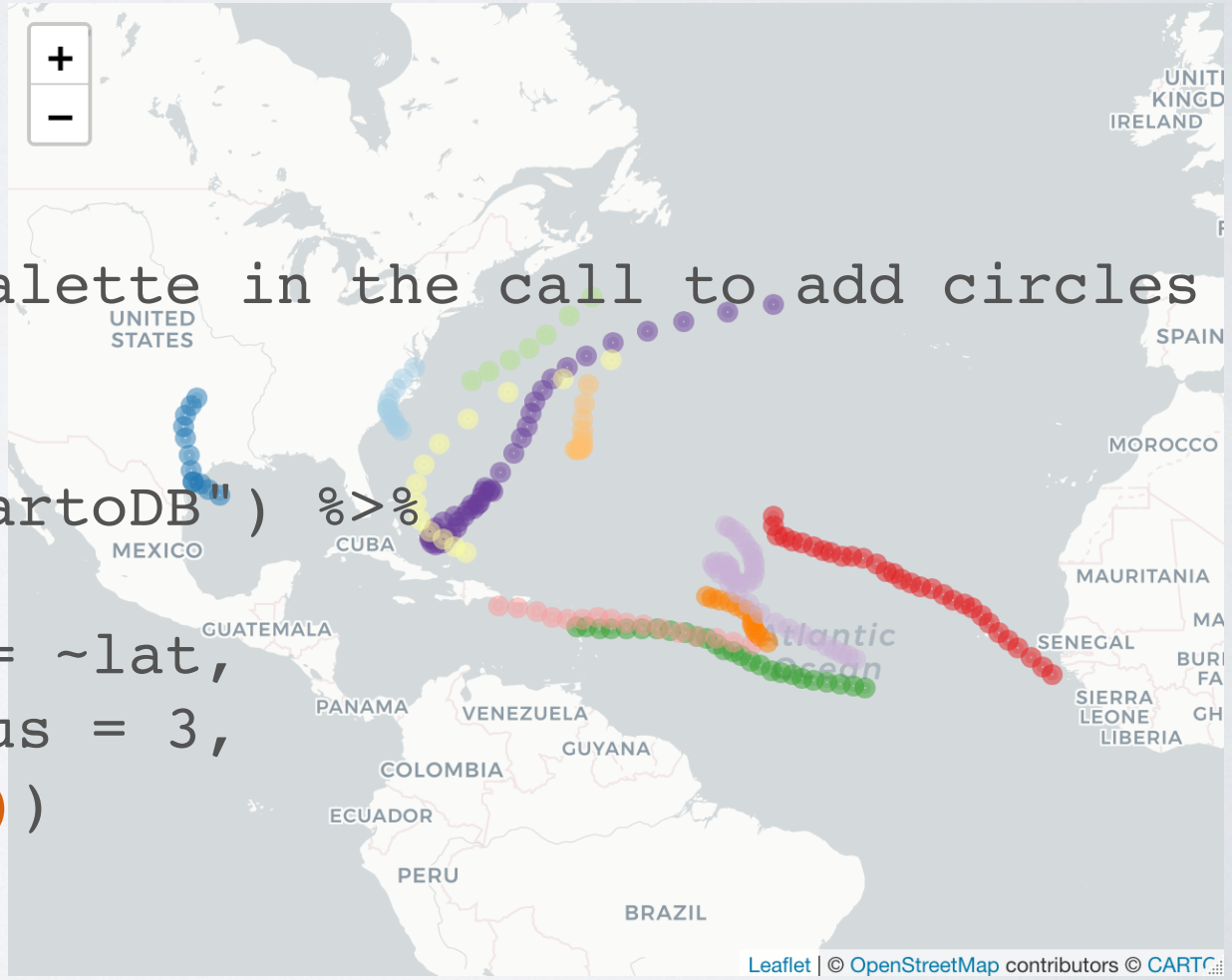
    `pal <- colorFactor(palette = …, levels = …)`
  - Apply the color palette on a variable in the data

    `color = ~pal(name)`
- Follow along in `08_color-markers.R`

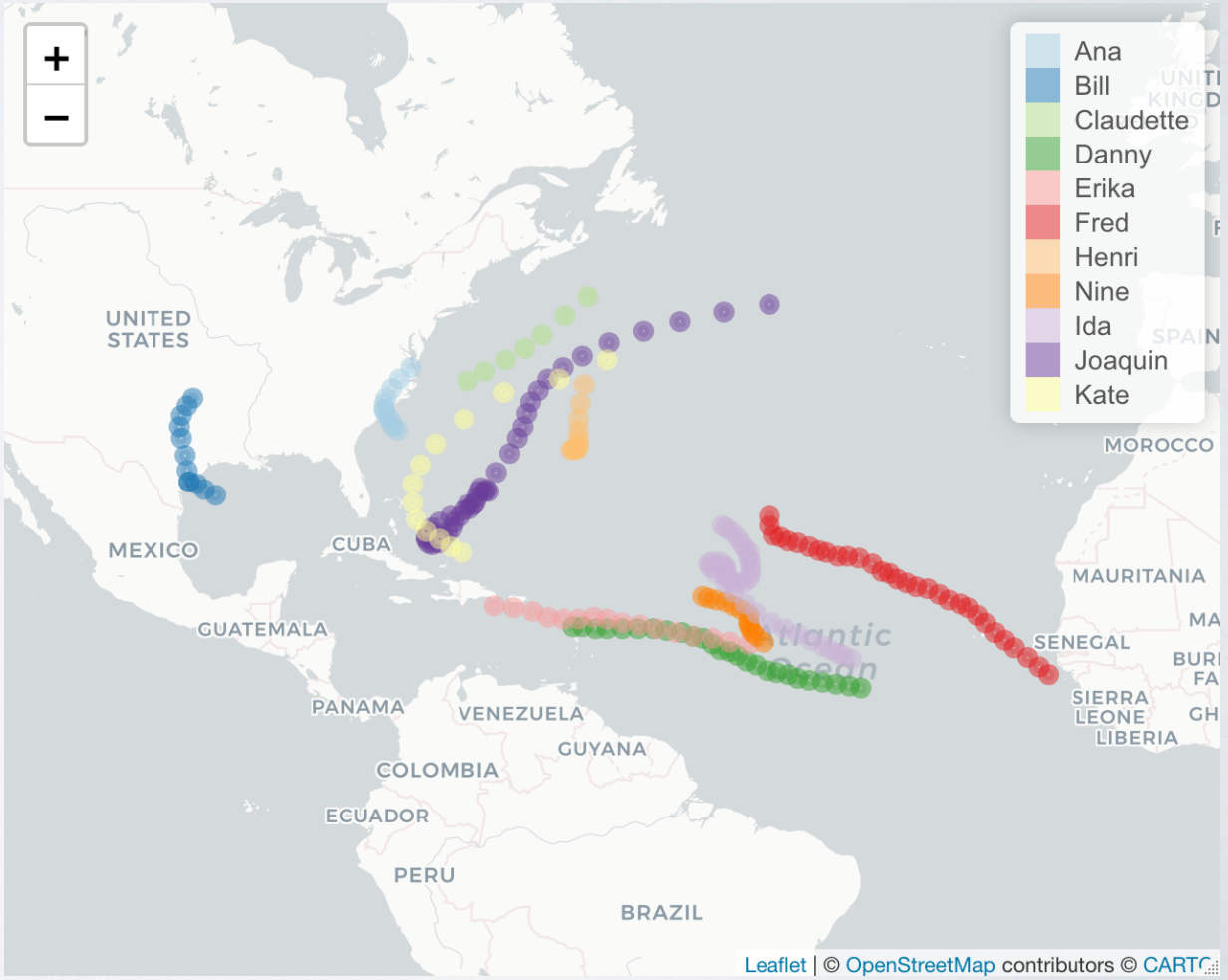
```
# create a color palette that connects colors with
levels:
(names <- unique(st15$name))
(colors <- RColorBrewer::brewer.pal(n = length(names),
name="Paired"))

pal <- colorFactor(
  palette = colors,
  levels = names)

# now use the color palette in the call to add circles
st15 %>%
  leaflet() %>%
  addProviderTiles("CartoDB") %>%
  addCircleMarkers(
    lng = ~long, lat = ~lat,
    label=~name, radius = 3,
    color = ~pal(name))
```



```
# and add a legend
st15 %>%
  leaflet() %>%
  addProviderTiles("CartoDB") %>%
  addCircleMarkers(lng = ~long, lat = ~lat, label=~name,
                   radius = 3, color = ~pal(name)) %>%
  addLegend(colors = colors, labels = names)
```



CREATING A COLOR PALETTE FOR A NUMERIC VARIABLE

- A palette for a numeric variable is created very similarly:
  - Create a color palette with colors and a range (domain)

```
pal <- colorNumeric(palette = …,
                    domain = …, reverse = TRUE)
```

  - Apply the color palette on a variable in the data

```
color = ~pal(name)
```

  - `addLegend(title = "some title", pal = pal, values = c(min, max))`

MORE FUNCTIONALITY

- Check the leaflet.js documentation
- R package leaflet.extras adds OSM capability:
  `addSearchOSM` adds search functionality
  `addReverseSearchOSM` allows geocoding with mouse click
  `addResetButton` resets the map to its original state

HOW DOES SHINY WORK?

LAYERS

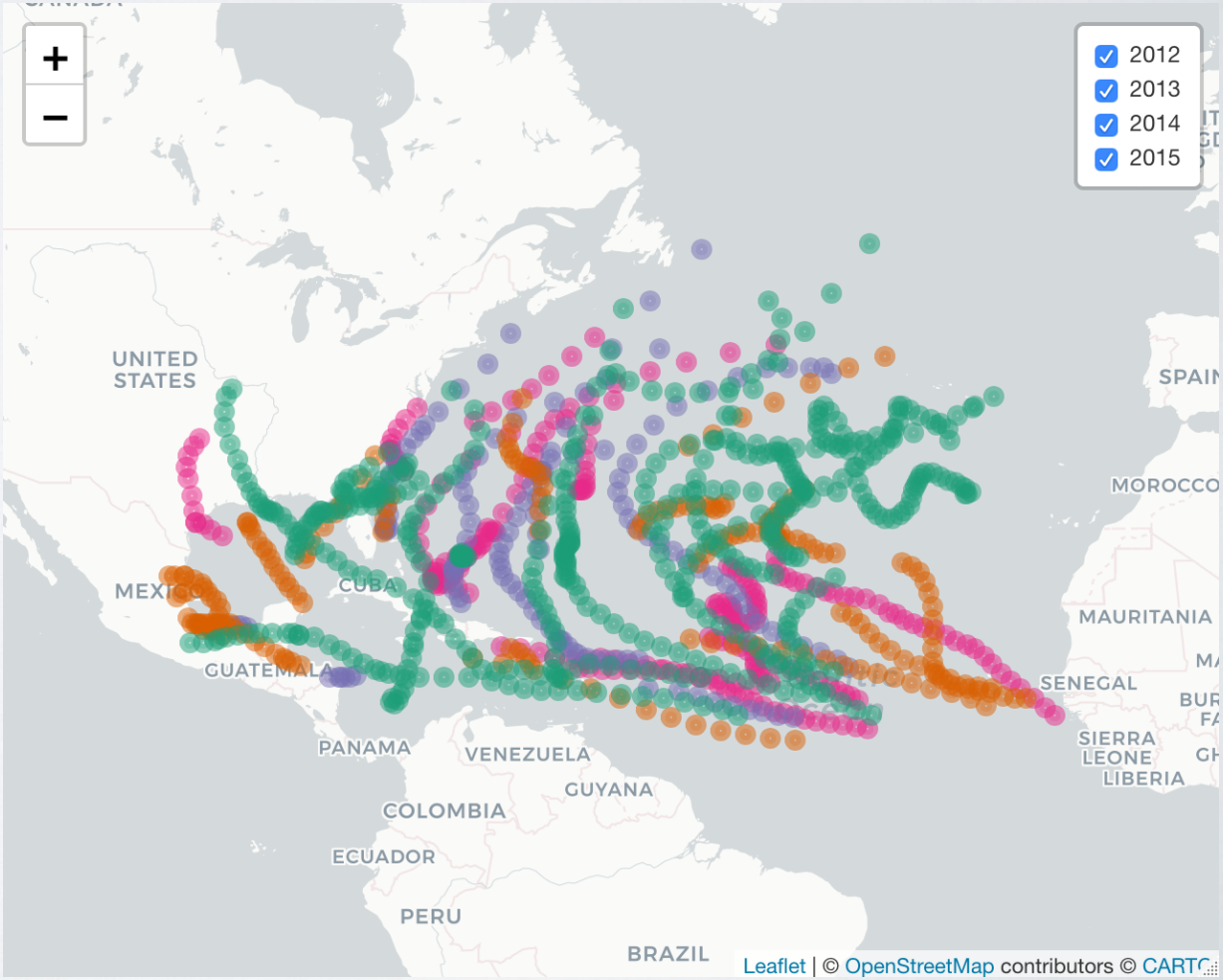
- Leaflet works in form of layers (similarly to ggplot2)
- By using the **`group`** parameter, we can give a layer a name
- **`addLayersControl`** then adds a control widget that allows us to interactively include or exclude that layer

ADDLAYERSCONTROL

```
addCircleMarkers(lng = ~long, lat = ~lat, label=~name,
                 radius = 3, color = ~pal(year),
                 data = recent %>% filter(year == 2012),
                 group = "2012")
```

Repeat code above for all years that should be included

```
addLayersControl(
  overlayGroups =
        c(2012:2015),
  options =
   layersControlOptions(
     collapsed = FALSE)
)
```



OTHER LAYERS

- Instead of adding markers in geographic locations, leaflet supports adding paths, polygons, polylines, … etc

- The storms data really consists of storm tracks … can you figure out how to get the 2015 storms show up as polylines?

YOUR TURN

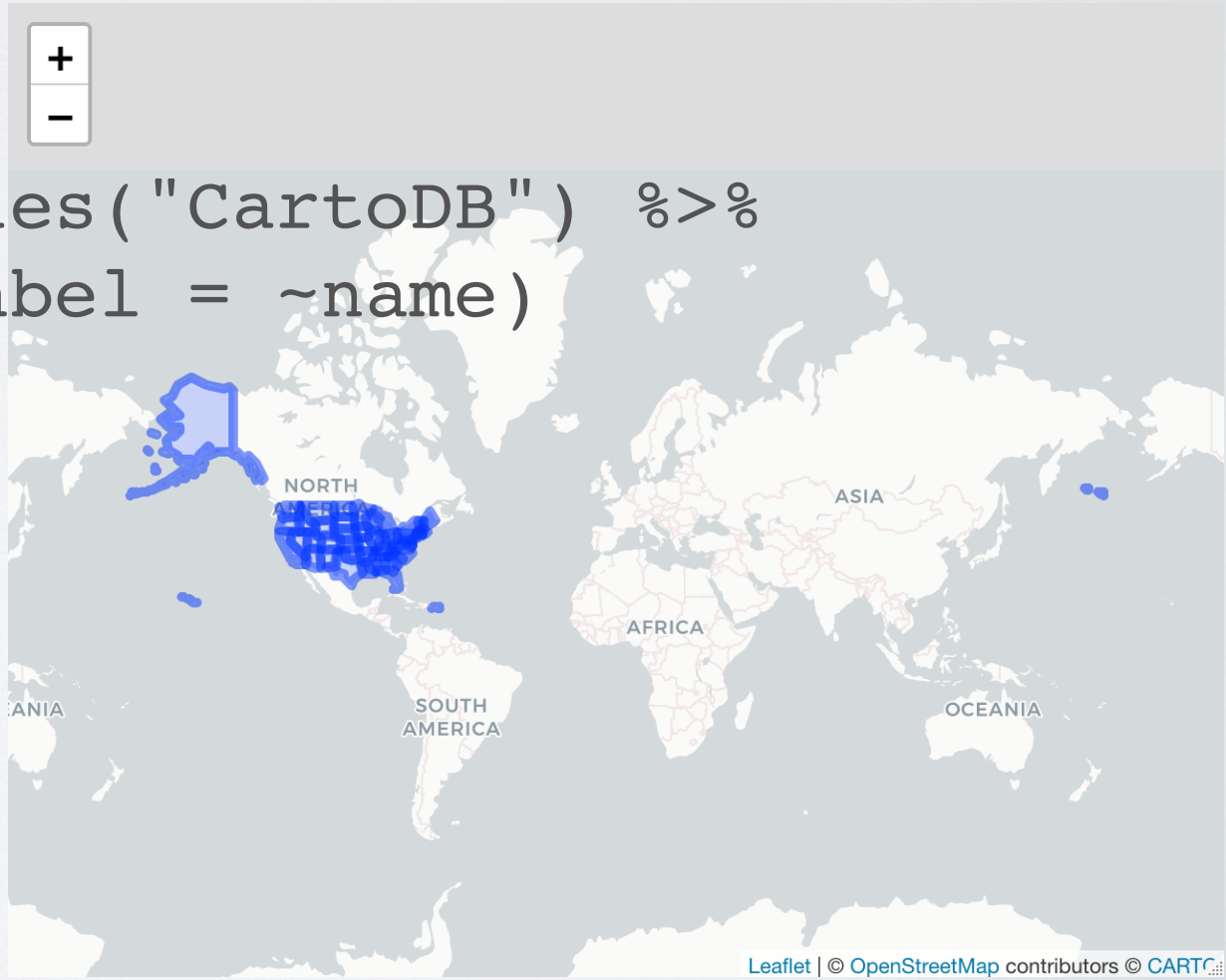
- Investigate the function **`addPolylines`** to add the storms of 2015 as storm tracks
- Keep the groups and addLayersControl in mind!
- Have a look at 10_layers-your-turn.R in case of need

POLYGONS

- GIS often deals with data in form of shapefiles
- The newest development in R regarding shapefiles is the package **`sf`** (simple features)
- The R package **`USAboundaries`** contains (historic) shapefiles of US states, counties, congressional districts, and zip code areas.
- High resolution shape files are included in an add-on package **`USAboundariesData`** that can be downloaded from ropensci (on github)

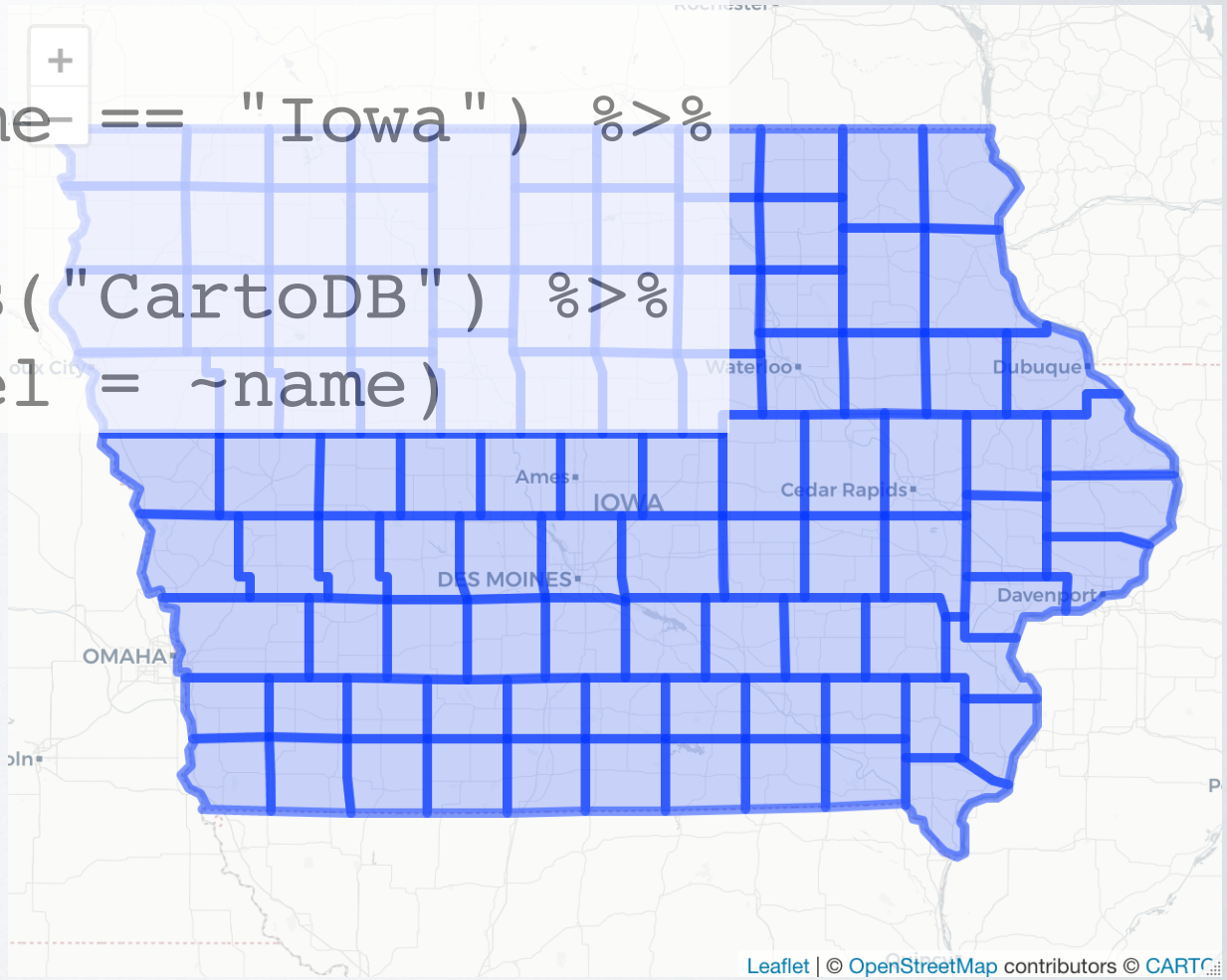
INCORPORATING POLYGONS

```
states <- us_states()
states %>%
  leaflet() %>%
    addProviderTiles("CartoDB") %>%
    addPolygons(label = ~name)
```



INCORPORATING POLYGONS

```
counties <- us_counties()
counties %>%
  filter(state_name == "Iowa") %>%
  leaflet() %>%
  addProviderTiles("CartoDB") %>%
  addPolygons(label = ~name)
```



Now we need to get some data in …

US CENSUS POPULATION ESTIMATES

- File co-est2019-alldata.csv contains estimates for each US county
- We use variables COUNTY, STATE, and POPESTIMATE2019
- Refer to co-est2019-alldata.pdf for explanations of the other variables.

MERGE DATA INTO SHAPES

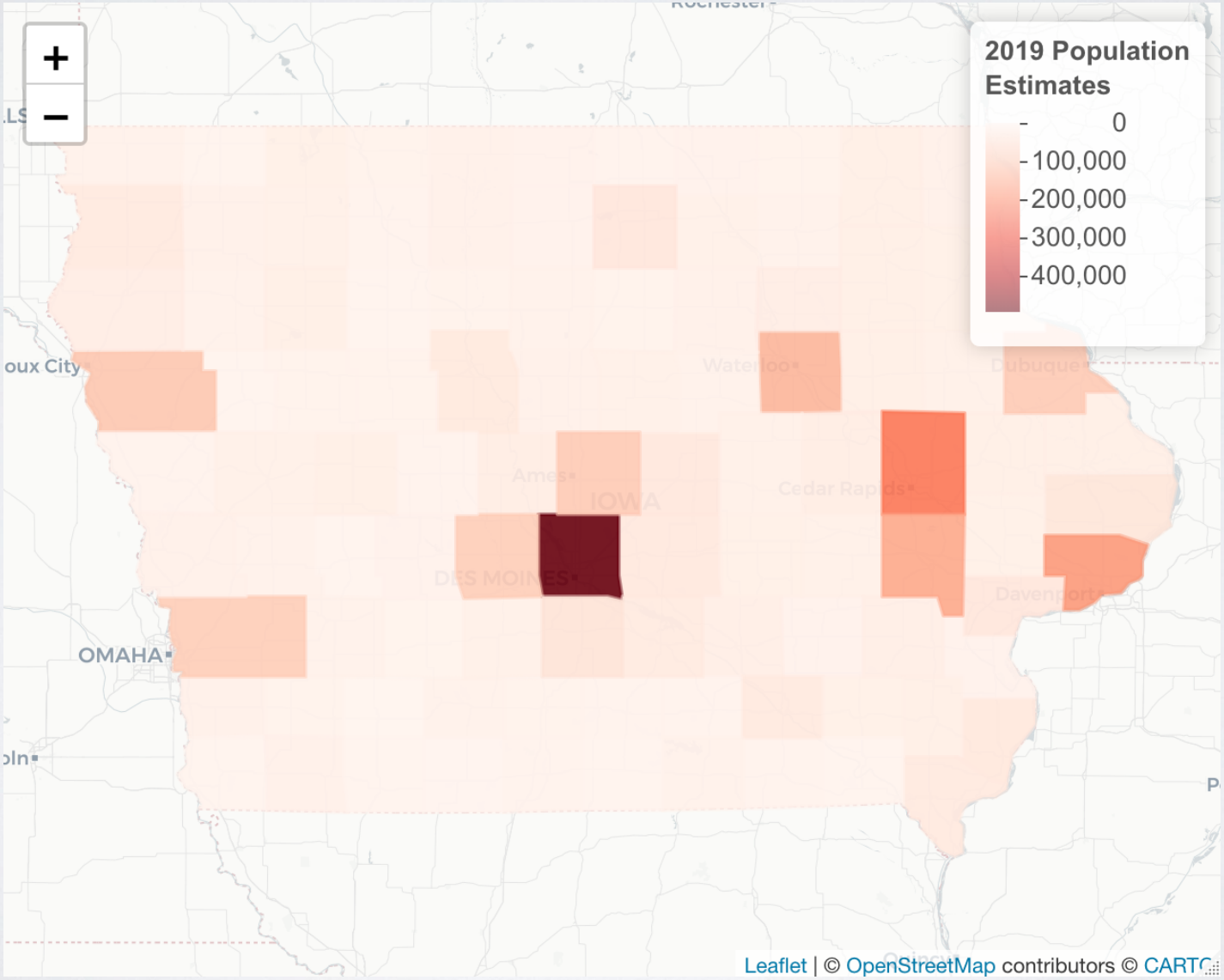
```
population <- read.csv("data/co-est2019-alldata.csv")

plotdat <- counties %>%
  filter(state_name == "Iowa") %>%
  left_join(population %>%
              select(STATE, COUNTY, STNAME, POPESTIMATE2019),
            by=c("COUNTY", "STATE"))
```

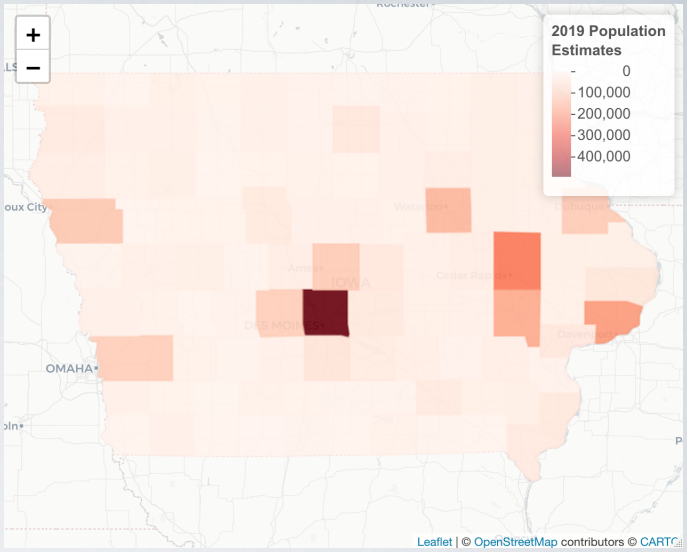
- Adding a numeric color scheme

```
scale_range <- c(0,max(plotdat$POPESTIMATE2019))
# missing values are bright green, so we can see them and
fix them :)
pal <- colorNumeric("Reds", scale_range,
                    na.color = "#aaff56", reverse=FALSE)
```

POPULATION DENSITY



POPULATION DENSITY




```
scale_range <- c(0,max(plotdat$POPESTIMATE2019))

plotdat %>%
  leaflet() %>%
  addProviderTiles("CartoDB") %>%
  addPolygons(color = ~pal(POPESTIMATE2019),
              fillOpacity = 0.9, weight = 2,
              popup = ~paste0(name,", ",
                              str_extract(state_name, "^([^,]*)"),
                              "<br>", POPESTIMATE2019)) %>%
  addLegend(title="2019 Population<br>Estimates",
            position = "topright",
            pal = pal,
            values = scale_range)
```

11_shapefiles.R

YOUR TURN

- Open file 11_shapefiles.R
- Re-create the population density map for a different state
- Try out a different color scheme …
- Next up: incorporating leaflet into shiny!

LEAFLET IN SHINY

- There are only two commands necessary to include leaflet maps into shiny:
  - Defining a leaflet output: `leafletOutput`
  - Rendering a leaflet map: `renderLeaflet`

AN EXAMPLE

- Open the file12_shiny-leaflet.R in RStudio and run it …
- The app incorporates multiple aspects of what we have discussed

!! NOW IT'S YOUR TURN !!

YOUR TURN - HOW & WHEN?

- We will use the break rooms to give each team a private room
- Haley, Mattie, and Heike will check in periodically and help solve problems
- Work on the app until 1:15.
- From 1:15 - 2:00 each team will present their app for a few minutes (3-5mins) to all of us