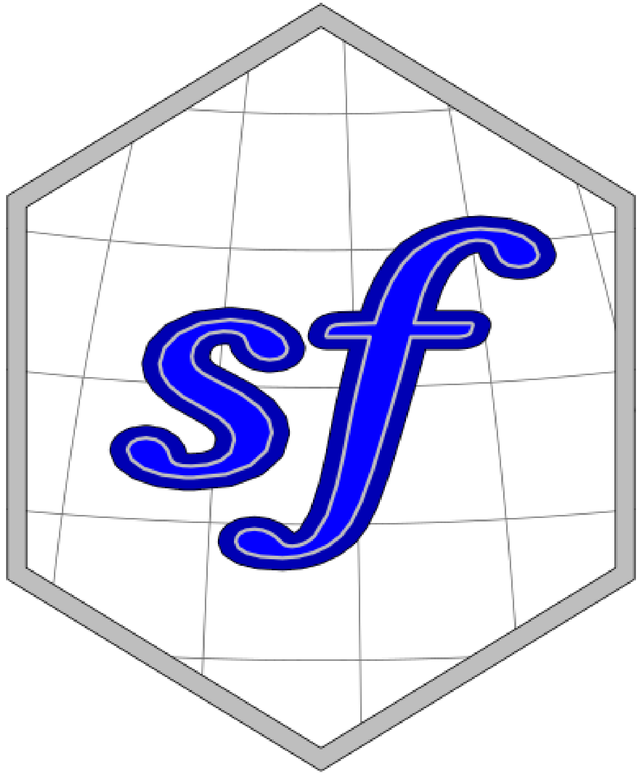
MAP DATA WITH SF & LEAFLET



OUTLINE

- Motivation
- sf
- ggmap

- Leaflet
  - Basemaps
  - Lines
  - Points
  - Shapes
  - Legends & Colors

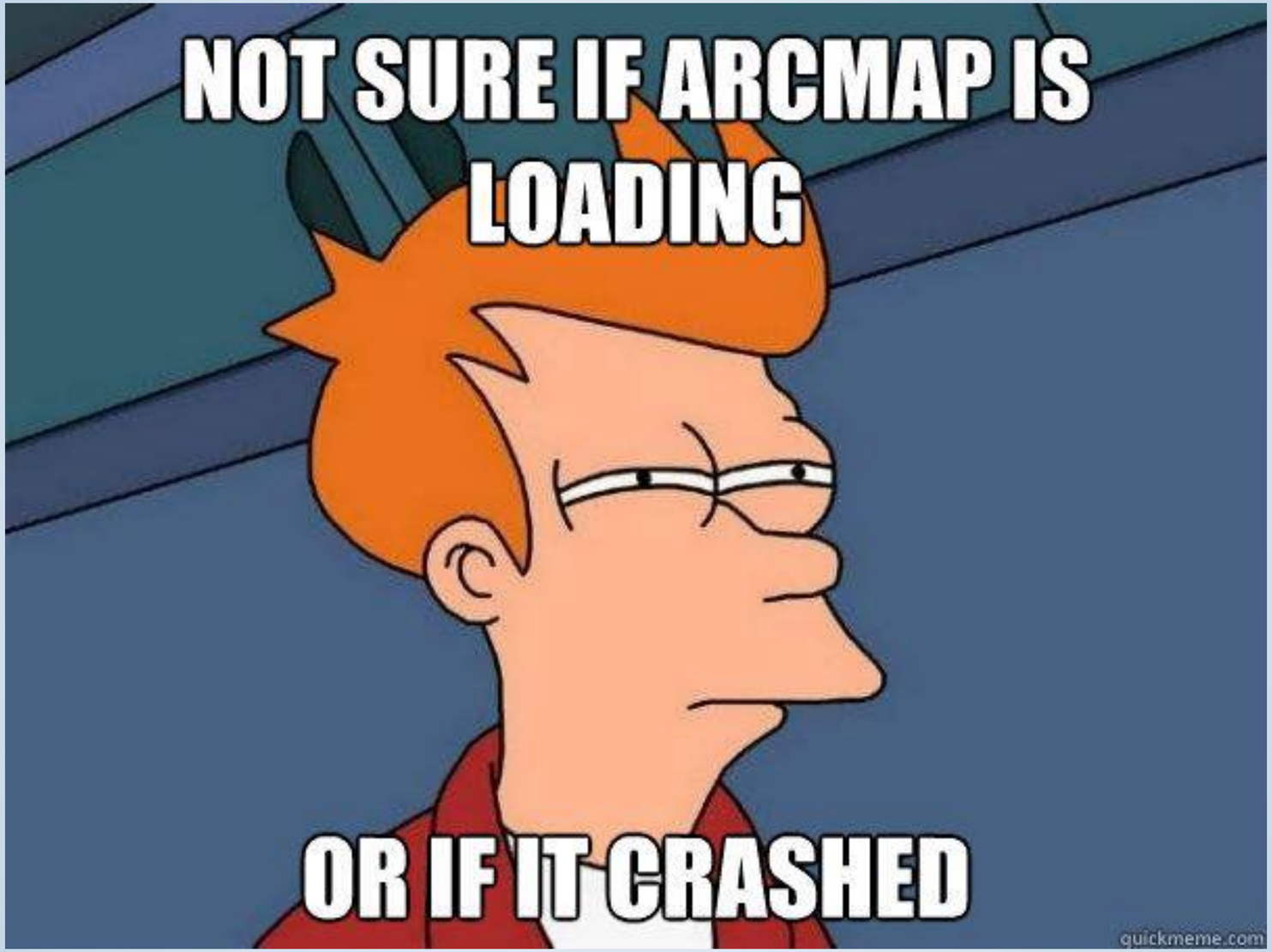
- leaflet.extras
  - Heatmaps

Motivation

Map of the 1854 Cholera outbreak - John Snow

*“‘I have the same problem’ is a famous **last** post in many forum-threads on the esri forum.”*

–sebastian


NOT SURE IF ARCMAP IS LOADING
OR IF IT CRASHED
quickmeme.com

“R and Leaflet are free and open-source, ArcMap is very much not.”

–Geoffrey Arnold

PACKAGES

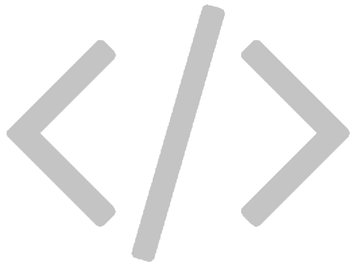
- Install these packages:
  - sf
  - leaflet
  - leaflet.extras

Spatial Data with sf

LOADING SPATIAL DATA

- Sometimes your source data will be a data frame which you will need to join to spatial data, other times it will be included

- It may also be a CSV with coordinates, in those instances no further cleaning needs to take place



DEMO

08-Maps-Exercises.Rmd

Go to loading code chunk

IMPORTING DATA

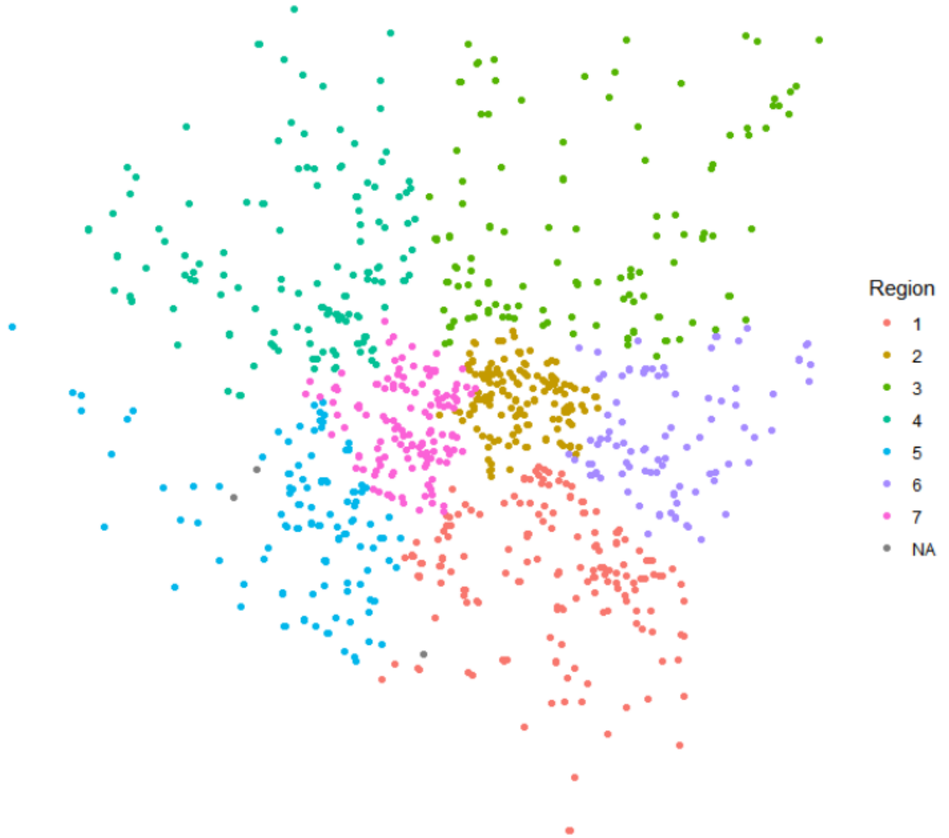
```
polls <- st_read("Allegheny_County_Polling_Place_Locations_November_2016.geojson")


cds.load <- st_read("./cb_2015_us_cd114_500k/cb_2015_us_cd114_500k.shp")
plot(cds.load)
```

BASIC PLOT

```
plot(cds.load)
```

BASIC GGPLOT

```
ggplot(data=polls, aes(color = Region)) +
  geom_sf() +
  theme_void()
```



ADDING A BASEMAP

- Request an API Key here: https://client.stadiamaps.com/signup/
- Check your email and enter the API key into the code below and run in the Console

```
register_stadiamaps(“Your-API-Key-here", write = TRUE)
```

GET A STAMEN API KEY

```
# Get the boundaries of your file to get the basemap
bbox <-st_bbox(polls) %>% setNames(c("left", "bottom", "right", "top"))

# Get the API Key from the .Renviron
ggmap::register_stadiamaps(Sys.getenv("GGMAP_STADIAMAPS_API_KEY"))

# Load API Basemap
map <- get_stadiamap(bbox, maptype = "stamen_toner_lite")

# Add map info to basemap
map %>%
    ggmap() +
    coord_sf(crs = st_crs(3857)) +
    geom_sf(data=polls, aes(color = Region), inherit.aes = FALSE) +
    theme_map()
```


Leaflet

WHAT IS LEAFLET?

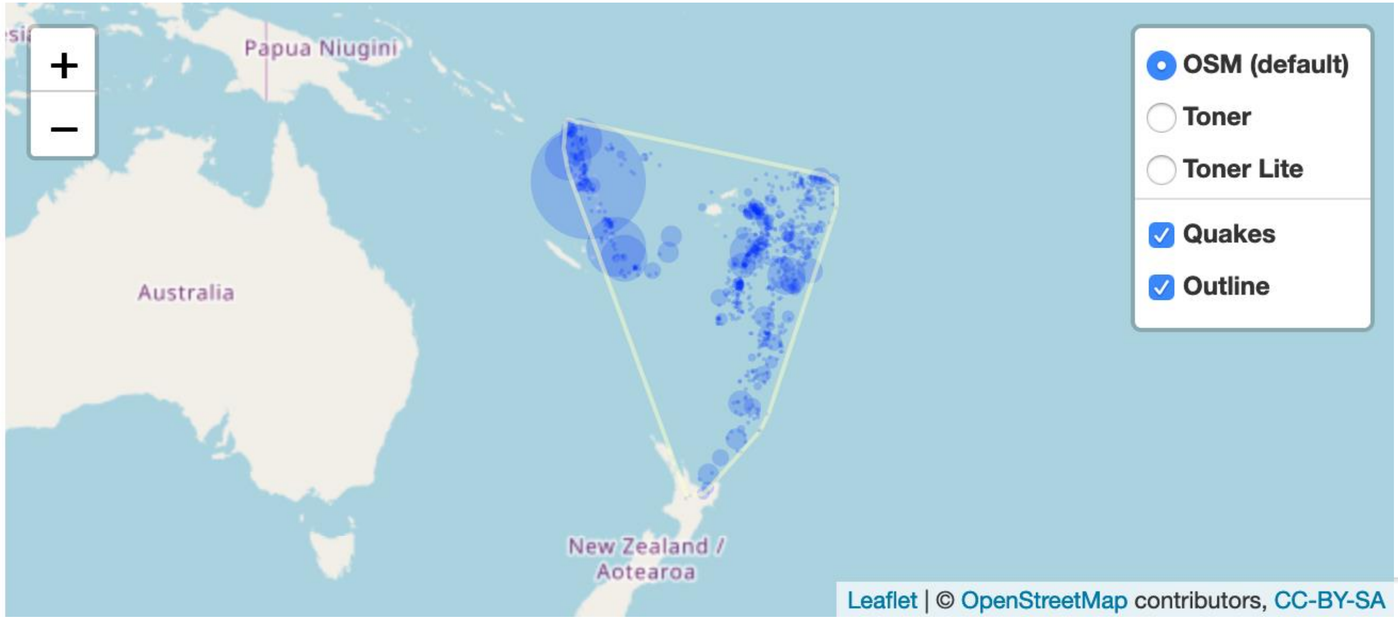
- R version of the Javavscript API of Leaflet
- One of the ways to get interactive maps
  - Others include: tmap, mapview and plotly
- Documentation: https://rstudio.github.io/leaflet/

GROUP CONTROLS

In leaflet you can add groups and give them a name.

Group names are what show up in the layer control

```
addProviderTiles(providers$Stamen.Toner, group = “Toner”) %>%
addLayersControl(
   baseGroups = c("OSM (default)", "Toner", "Toner Lite"),
   overlayGroups = c("Quakes", "Outline"),
   options = layersControlOptions(collapsed = FALSE)
 )
```



EXERCISE

- Create a map with layer controls
  - Get basemap provider names from here: https://leaflet-extras.github.io/leaflet-providers/preview/
    - Create 3 named basemap groups
    - Add a layer control that has the named base groups

5m 00s

Shapes

TYPICAL ARGUMENTS

- lng (if a column)
- lat (if a column)
- layerId
- group (for layerControls)
- stroke (boolean, shape outline)
- color (outline color, hex values)
- weight (outline width)
- opacity (alpha of the line)
- fill (boolean)
- fillColor (hex color)
- fillOpacity (alpha of fill)
- And more!

EXERCISE

- Using the example cds to create a polygon layer
  - Make sure to include a basemap

5m 00s

SOLUTION

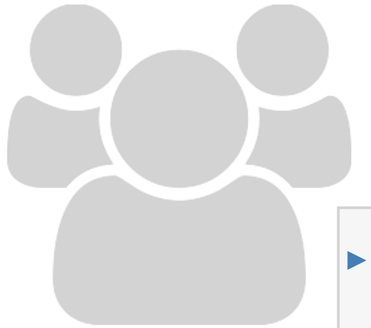
```
leaflet(data = cds) %>%
  addProviderTiles("Stamen.Toner") %>%
  addPolygons()
```

PALETTES

- palette (color brewer palette)
- domain (values to be colored)
- na.color
- alpha
- reverse (values of palette)
- bins
- pretty
- right (for cutting)
- n (number of quanitites)
- probs
- levels
- ordered

LEGENDS

- position (“bottomright” etc…)
- pal (palette from colorBrewer)
- values (domain from data)
- na.labels
- bins (buckets)
- colors
- labFormat (separate function labelFormat)
- title
- className (for custom CSS to apply)
- layerId (for input usage)
- group (for layerControls)
- digits
- big.mark
- transform (function to be applied to labels)

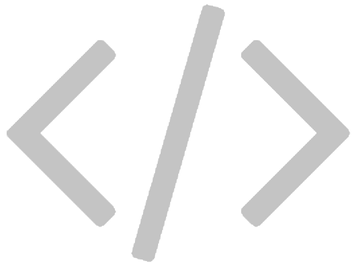


EXERCISE

- Choropleth Polygon map with
  - Create a color palette for Life Expectancy at Birth (years) column in the merged Congressional District data.
    - addPolygons to a leaflet map and apply palette to column
      - Stretch goal: Add a legend

5m 00s

Hint: https://rstudio.github.io/leaflet/choropleths.html

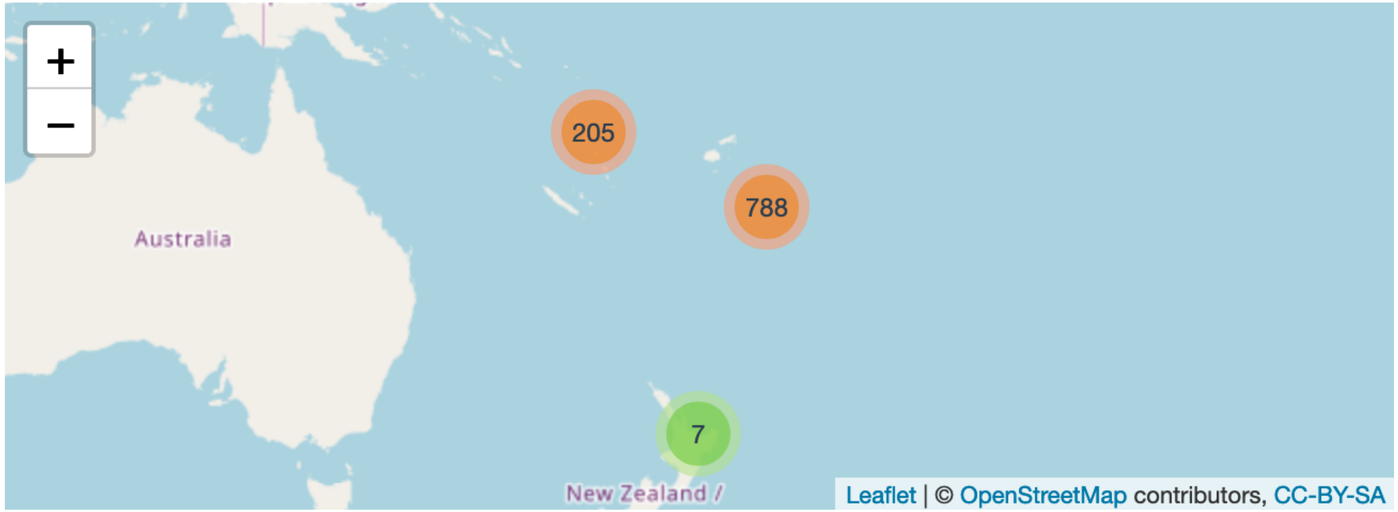


DEMO

08-Maps-Exercises.Rmd
Go to polygon code chunk

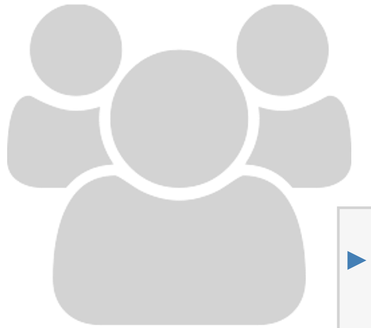
CLUSTERS

```
leaflet(quakes) %>%
addTiles() %>%
addMarkers(
  clusterOptions = markerClusterOptions()
)
```



WHY CLUSTERS?

- Sometimes there’s just a ton of data to show and trying to visualize them will bring any browser window it a crawl.



EXERCISE

- Go to 08-Maps-Exercises.Rmd clusters code chunk
  - Use 311 data to create a cluster map
    - Create a factor palette
      - Create a legend
        - Make sure to include a basemap

5m 00s

Hint: https://rstudio.github.io/leaflet/markers.html

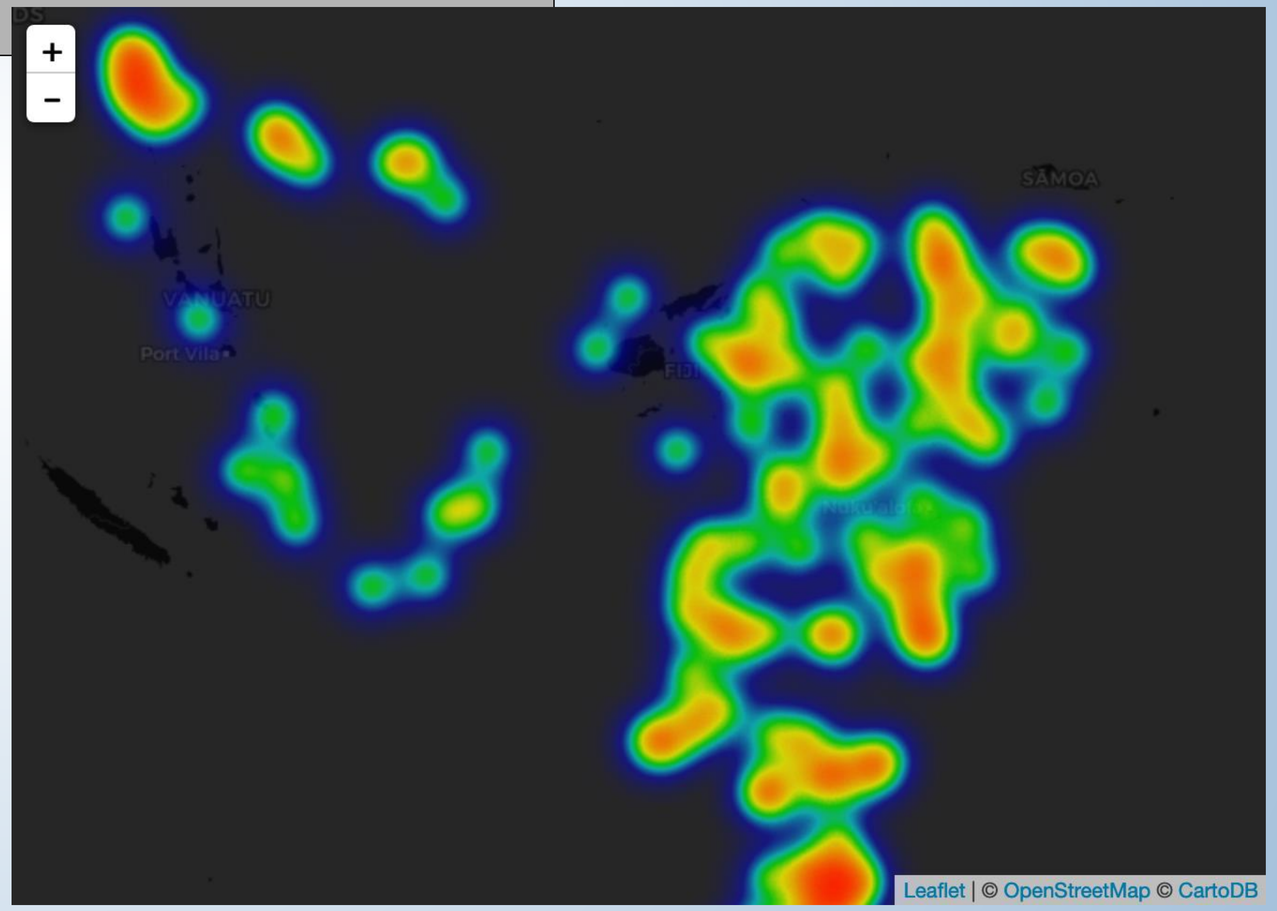
leaflet

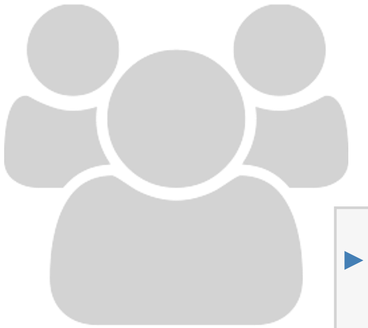
.extras

LEAFLET.EXTRAS

- Adds some pretty nice functions to complement leaflet
  - Tile caching: http://rpubs.com/bhaskarvk/TileLayer-Caching
  - weather icons: http://rpubs.com/bhaskarvk/leaflet-weather
  - Pulse icons: http://rpubs.com/bhaskarvk/leaflet-pulseIcon
  - and heat maps!

```
leaflet(quakes) %>%
addProviderTiles(providers$CartoDB.DarkMatter) %>%
  setView( 178, -20, 5 ) %>%
  addHeatmap(lng = ~long, lat = ~lat, intensity = ~mag,
             blur = 20, max = 0.05, radius = 15)
```

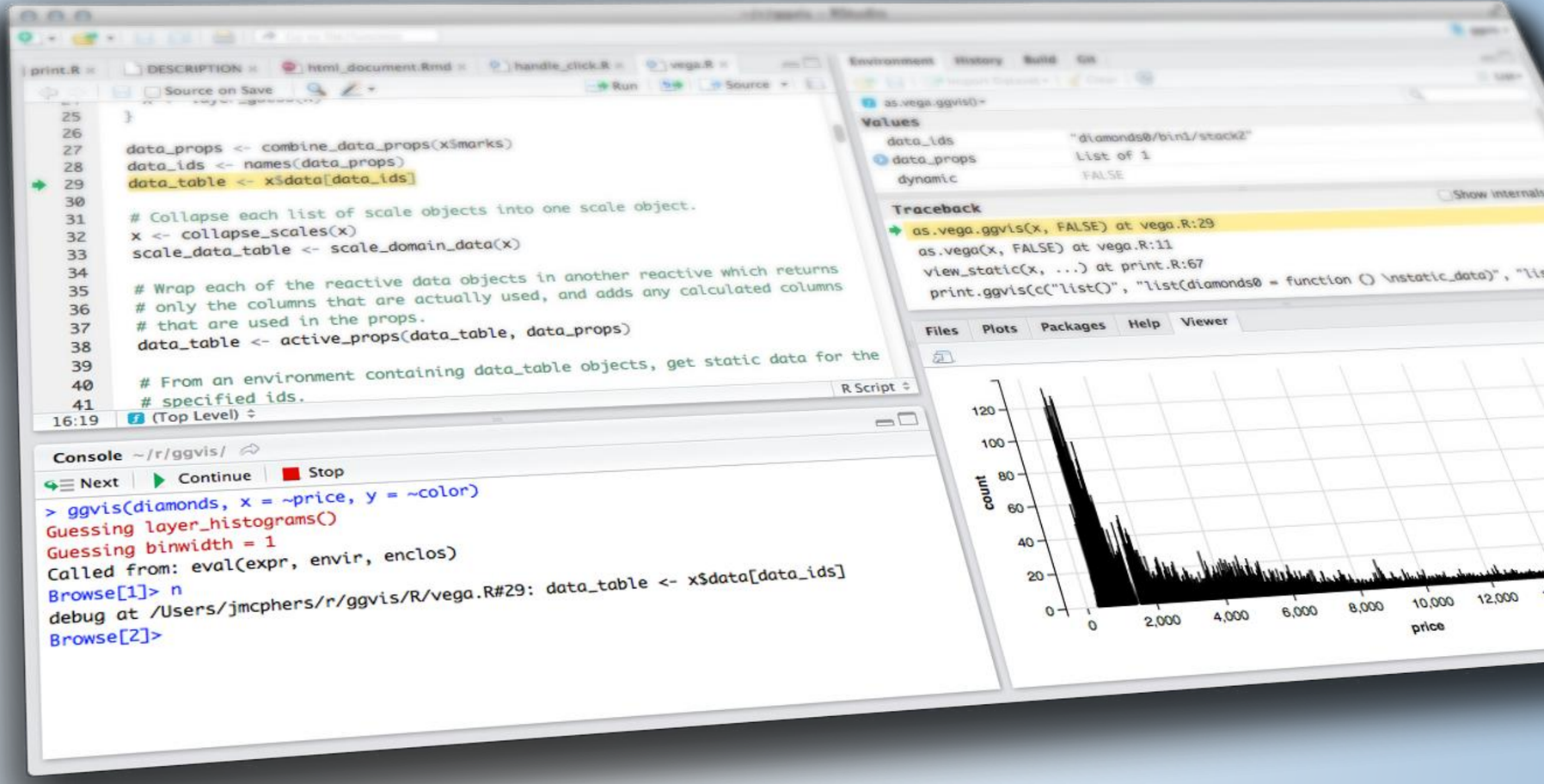




EXERCISE

- Using the pothole data create a heat map
  - This won’t be very different from your cluster map from before.
    - Play around with the arguments for heat maps and find a radius that looks good.
- Compare your outcome with your neighbor

5m 00s


print.R
DESCRIPTION
html_document.Rmd
handle_click.R
vega.R
Source on Save
Run
Source
25 }
26
27 data_props <- combine_data_props(x$marks)
28 data_ids <- names(data_props)
29 data_table <- x$data[data_ids]
30
31 # Collapse each list of scale objects into one scale object.
32 x <- collapse_scales(x)
33 scale_data_table <- scale_domain_data(x)
34
35 # Wrap each of the reactive data objects in another reactive which returns
36 # only the columns that are actually used, and adds any calculated columns
37 # that are used in the props.
38 data_table <- active_props(data_table, data_props)
39
40 # From an environment containing data_table objects, get static data for the
41 # specified ids.
16:19
(Top Level)
R Script
Environment
History
Build
Git
as.vega.ggvis()
Values
data_ids "diamonds0/bin1/stack2"
data_props List of 1
dynamic FALSE
Traceback
as.vega.ggvis(x, FALSE) at vega.R:29
as.vega(x, FALSE) at vega.R:11
view_static(x, ...) at print.R:67
print.ggvis(c("list()", "list(diamonds0 = function () \nstatic_data)", "li
Files
Plots
Packages
Help
Viewer
Console ~/r/ggvis/
Next
Continue
Stop
> ggvis(diamonds, x = ~price, y = ~color)
Guessing layer_histograms()
Guessing binwidth = 1
Called from: eval(expr, envir, enclos)
Browse[1]> n
debug at /Users/jmcphers/r/ggvis/R/vega.R#29: data_table <- x$data[data_ids]
Browse[2]>
count
price
0
20
40
60
80
100
120
2,000
4,000
6,000
8,000
10,000
12,000

LEAFLET