Spatial Analysis with R

Working with Spatial Data in Scripts

March 19 & 20, 2025

DARTMOUTH

A roadmap for this workshop

- Why Scripting?
- Basics of Reproducible Research, applied to spatial data
- Some hurdles of reproducible research with spatial data, and some methods to overcome these hurdles
- Geographic data and spatial analysis
- Tools of the trade for spatial analysis
- Live-coding using R and R Studio with a reproducible spatial analysis
- Questions and assistance: contact us at researchdatahelp@dartmouth.edu

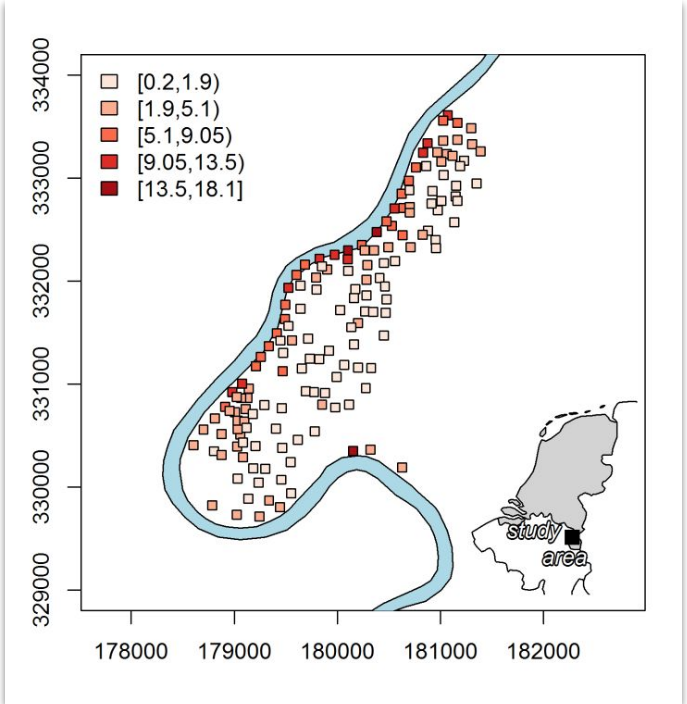
Basics of Reproducible Research - Spatial Data

To make our research reproducible:

- Provide data, with metadata, and the code and software or software version used to run the analysis
- Be transparent about the research
- Test that you can generate the same result more than once
- Other researchers can generate the same result, given the data and the scripts or programs that capture the methodology of the analysis
- Run the same analysis steps, given new data with an identical data structure. For example, the data is updated with new observations(rows) and the analysis is re-run.

Why Scripting?

- Allows an analysis to be repeated, either with new data or original data
- Can save time, save money
- Can help make research more reproducible
- Prevents wasted efforts
- Increased scientific credibility
- Often required by granting agencies and organizations
- Allows researchers to innovate at a faster rate with fewer errors



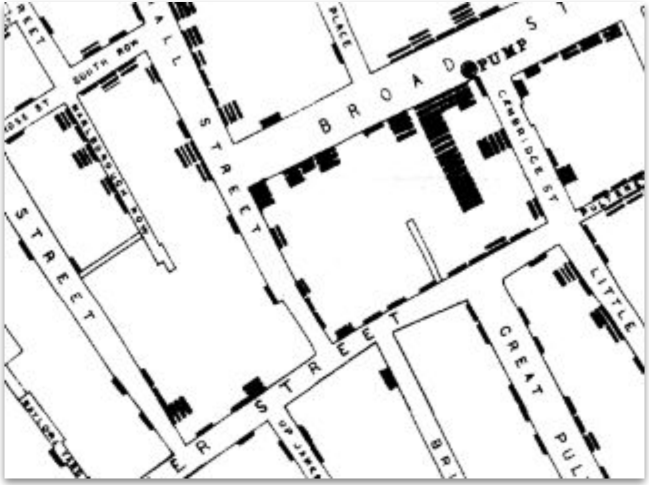
Heavy metal concentrations in contaminated soils along the Meuse River in Europe
See: https://rpubs.com/liem/63374
Image: https://www.r-bloggers.com/2016/07/creating-inset-maps-using-spatial-objects/

DARTMOUTH

Spatial Data from 1854

Map from the book "On the Mode of Communication of Cholera" by Dr. John Snow, originally published in 1854 by C.F. Cheffins, Lith, Southampton Buildings, London, England.

This early form of disease tracking and spatial analysis has led some to call Dr. John Snow the 'father of epidemiology'



Some hurdles of reproducible research with spatial data

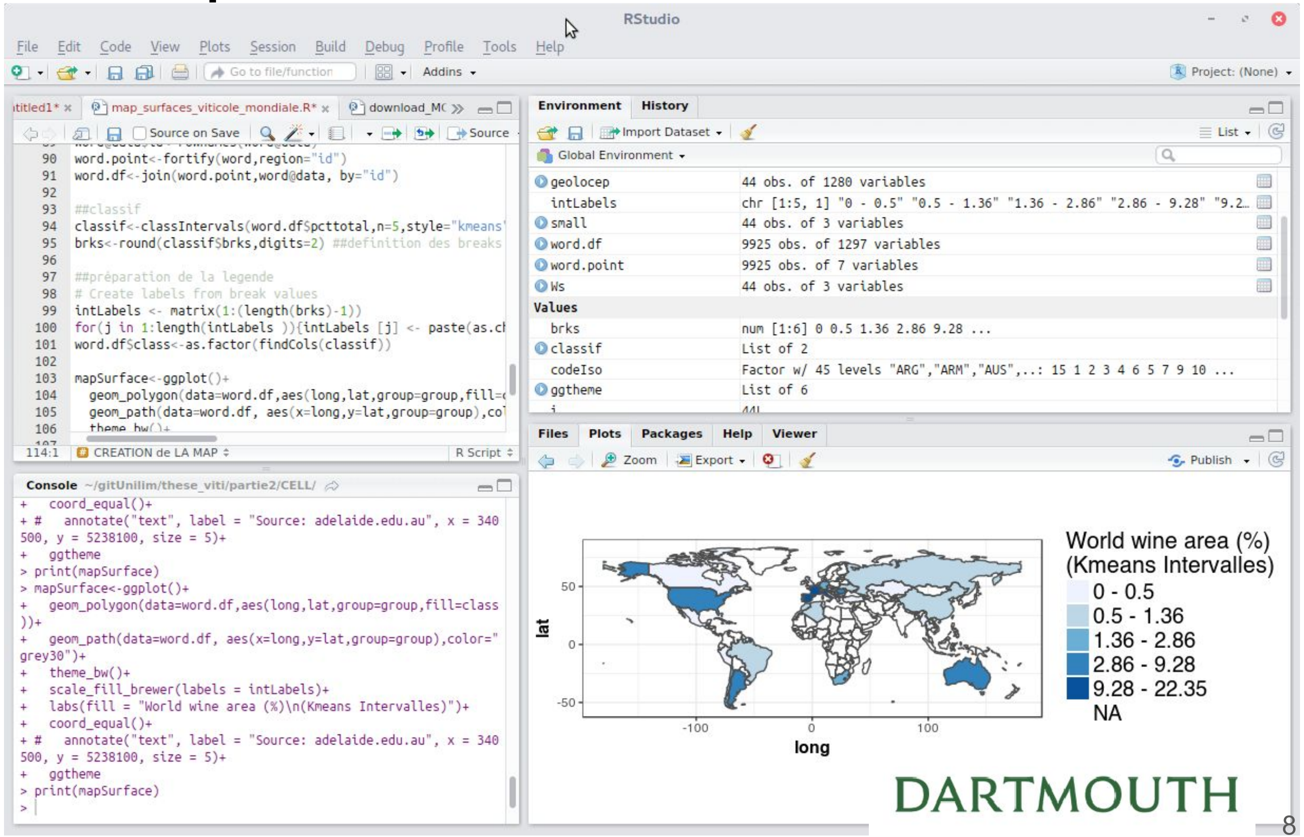
- Proprietary software
- Point-and-click software
- Large, very large and extremely large datasets
- Messy datasets that require multiple steps to 'tidy' up
- Data passed through various people without proper metadata or documentation
- Human error & basic forgetfulness
- Project organization and management of project resources

Ways to get past these hurdles

- When possible, use non-proprietary software and libraries
- Reduce or eliminate the use of point-and-click steps that are not easily written in to scripts or code
- For analyses with large datasets, consider running analysis scripts on a subset of the data, and make sure that this process is reproducible.
- Stay organized
- Document processes, data gathering techniques, script code
- Have both machine-readable processes and human-readable documentation

R with open-source spatial libraries

- Spatial Overlay
- Spatial analysis
- Geostatistics
- Point pattern analysis
- Spatial regression



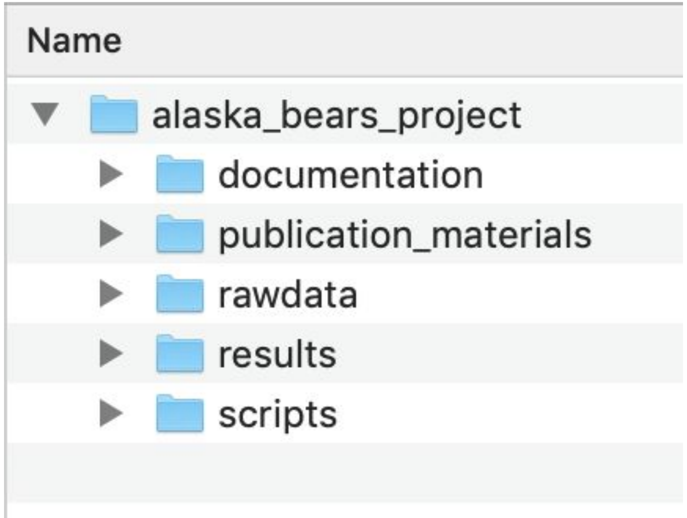
```r
word.point<-fortify(word,region="id")
word.df<-join(word.point,word@data, by="id")

##classif
classif<-classIntervals(word.df$pcttotal,n=5,style="kmeans'
brks<-round(classif$brks,digits=2) ##definition des breaks

##préparation de la legende
# Create labels from break values
intLabels <- matrix(1:(length(brks)-1))
for(j in 1:length(intLabels )){intLabels [j] <- paste(as.ch
word.df$class<-as.factor(findCols(classif))

mapSurface<-ggplot()+
  geom_polygon(data=word.df,aes(long,lat,group=group,fill=c
  geom_path(data=word.df, aes(x=long,y=lat,group=group),col
  theme_bw()+
```

```
geolocep     44 obs. of 1280 variables
intLabels    chr [1:5, 1] "0 - 0.5" "0.5 - 1.36" "1.36 - 2.86" "2.86 - 9.28" "9.2…
small        44 obs. of 3 variables
word.df      9925 obs. of 1297 variables
word.point   9925 obs. of 7 variables
Ws           44 obs. of 3 variables
Values
brks         num [1:6] 0 0.5 1.36 2.86 9.28 ...
classif      List of 2
codeIso      Factor w/ 45 levels "ARG","ARM","AUS",..: 15 1 2 3 4 6 5 7 9 10 ...
ggtheme      List of 6
```

```
+   coord_equal()+
+ #   annotate("text", label = "Source: adelaide.edu.au", x = 340
500, y = 5238100, size = 5)+
+   ggtheme
> print(mapSurface)
> mapSurface<-ggplot()+
+   geom_polygon(data=word.df,aes(long,lat,group=group,fill=class
))+
+   geom_path(data=word.df, aes(x=long,y=lat,group=group),color="
grey30")+
+   theme_bw()+
+   scale_fill_brewer(labels = intLabels)+
+   labs(fill = "World wine area (%)\n(Kmeans Intervalles)")+
+   coord_equal()+
+ #   annotate("text", label = "Source: adelaide.edu.au", x = 340
500, y = 5238100, size = 5)+
+   ggtheme
> print(mapSurface)
>
```

Data management - Sample folder structure

- Projectname
  - data (raw data - this folder can be read-only)
  - results
  - scripts (analysis scripts)
  - Publication_materials
  - documentation



Name

- alaska_bears_project
  - documentation
  - publication_materials
  - rawdata
  - results
  - scripts

- Other notes:
  - Include 'readme' files describing structure, process, etc
  - Use a system like Github to track changes and versions
  - Keep a copy of all folders locally and on a server. Where large datasets make this less practical, keep a small subset of the data with the scripts and results. Subset should be in exactly the same format as the larger dataset

Some common formats

- Shapefiles and geodatabases - a group of files that can store points, lines and polygons with geographic coordinates (both developed by ESRI-ArcGIS)
- CSV file, Google sheet, Excel Sheet with geospatial columns
- CSV without geospatial columns, but some address info that can be geocoded(convert a street address to Latitude/Longitude)
- Geojson - a version of Javascript Object Notation (json) specifically designed to store geographic data (open format / text-editor readable) see also https://geojson.io/
- KML (Keyhole Markup Language / Google Maps / Google Earth)
- Raster GIS imagery (jpg, tiff, png, etc)

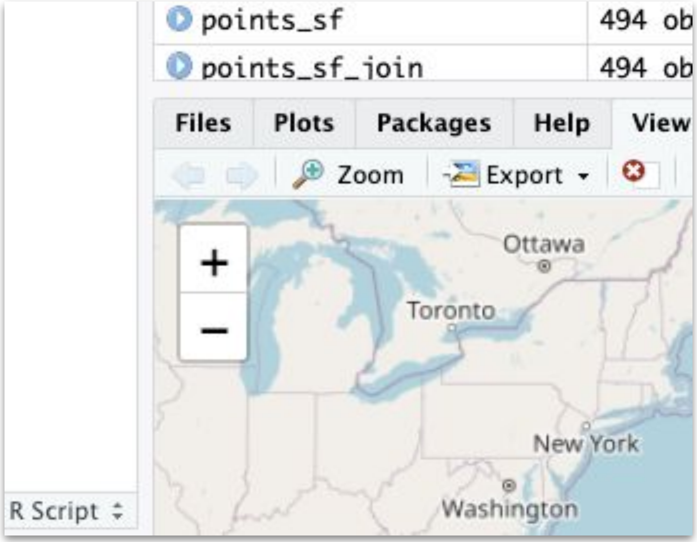
Some useful R packages for spatial data

See https://cran.r-project.org/web/views/Spatial.html

- ggplot2
- ggmap - map plotting package
- leaflet - base maps
- osmdata - open street map data, geocode an address, download map tiles
- sf - simple features
- sp
- tidyverse
- dplyr
- rnaturalearth
- rnaturalearthdata
- maps, tmap - thematic maps for R , tmaptools - read and process spatial data
- gapminder

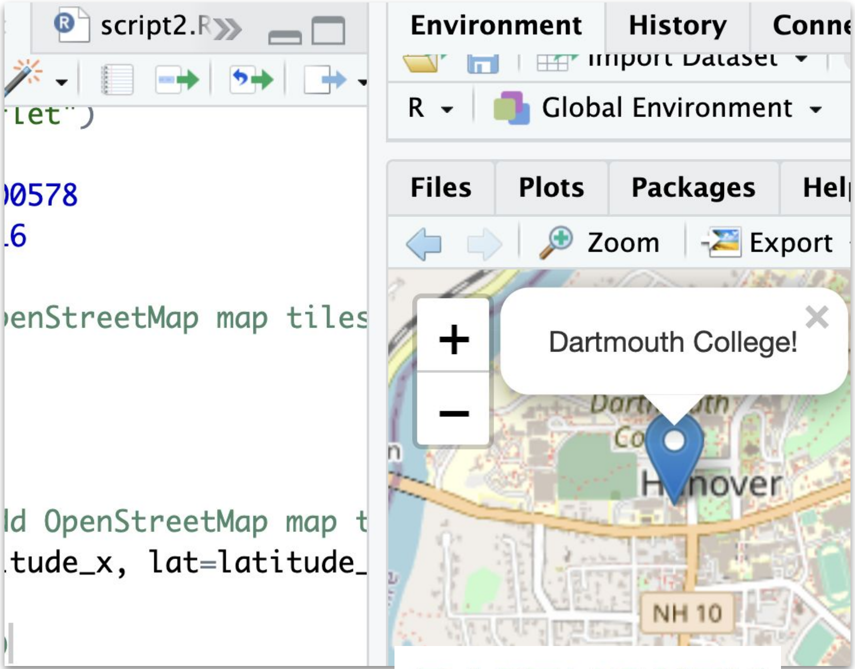
R code

```
install.packages("leaflet")
library(leaflet)
longitude_x <-   -72.2900578
latitude_y <-   43.703016
# add OpenStreetMap map tiles, right in R Studio!
m <- leaflet() %>%
  addTiles()
  # draw the map
m
```



R code

```
m <- leaflet() %>%
  addTiles() %>%  # add OpenStreetMap map tiles
  addMarkers(lng=longitude_x, lat=latitude_y, popup="Dartmouth College!")
m
```



```
let")

00578
.6

enStreetMap map tiles

ld OpenStreetMap map t
tude_x, lat=latitude_
```

Dartmouth College!

Reproducible Analysis Example

https://dartgo.org/rds-workshop

- Download and install R and RStudio
- Download both the "Bears Dataset" and "bears-csv" CSV
- Create a new folder on the desktop, call it bears_parks
- Inside this folder, create three folders: **data**, **results**, **scripts**
- Copy the CSV and the zip file to the **data** folder
- Open R Studio and create a new script (File > New file > R Script)

A Process Outline (human-readable/readme format)

A process outline and pseudo code for spatial data and analysis:

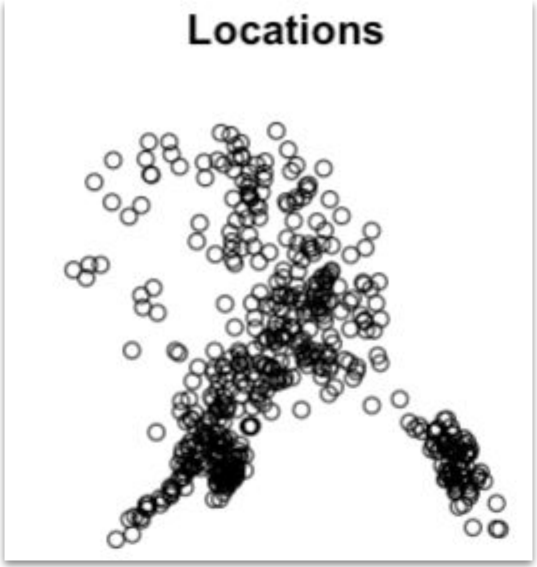
- Retrieve two datasets, one is a GIS 'shapefile' containing the boundaries of the US National Parks, second one is a CSV file of bear sightings with latitude and longitude locations
- Get the two datasets in to the same map projection and coordinate system, so that they will overlay properly in a GIS system or in R
- Use spatial analysis to find out if each bear is inside or outside of a park
- Report a basic statistic, the raw percentage of bears in parks, generate a map, and generate a new CSV file of the bears, with a field indicating the name of the park they were in, or 'null' if they were outside a park

Reproducible Spatial Analysis using R & R Studio

```
library(tidyverse)
library(dplyr)

# load spatial and mapping tools & libraries
library(sp)                     # For spatial objects and functionalities
library(sf)                     # For handling simple features
library(terra)                  # For working with raster and vector data
library(maps)                   # For plotting maps

setwd('~/Desktop/spatial_overlay/')
# pc users: setwd("C:/Users/username/desktop/spatial_overlay/")

# read in the CSV file using base R; we'll convert this later to spatial data
points <- read.csv('point-locations.csv')
```

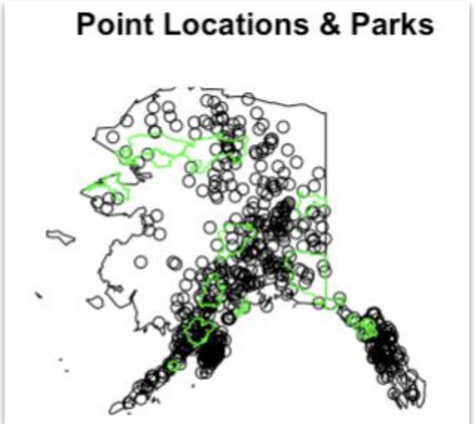
Spatial Analysis - loading points and polygons

```
# Convert the points data frame to a simple features object
# set the crs to  epsg code 4326.  See https://epsg.io/4326  for more info
points_sf <- sf::st_as_sf(points, coords = c('longitude', 'latitude'), crs = 4326)

# Plot point locations, just using the base plot function (not ggplot yet )
base::plot(sf::st_geometry(points_sf), main = "points - locations")

# add a rough outline of the map region using the 'maps' package
maps::map("world", region="usa", add=TRUE)
```



Spatial Data in R - loading points and polygons

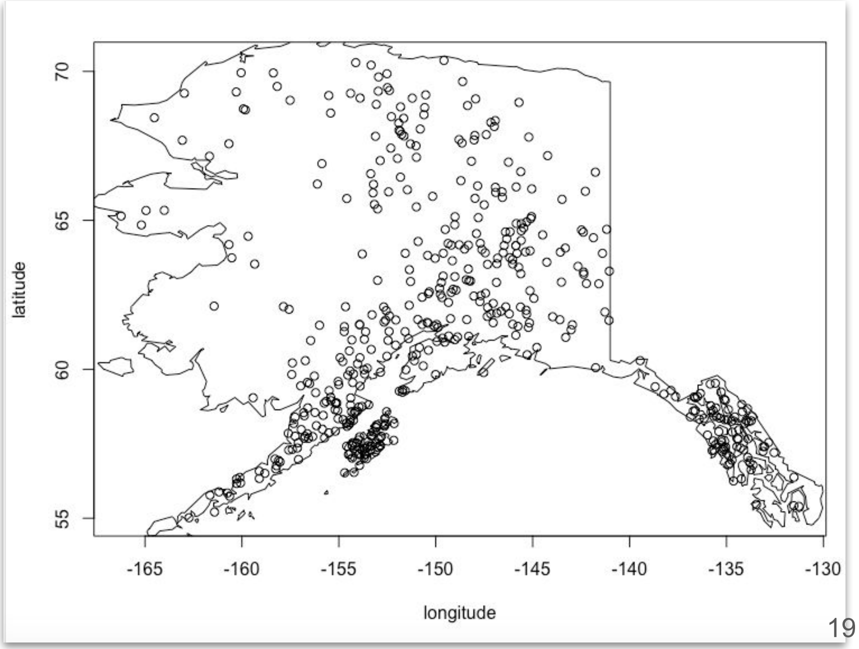
```
# read in the polygon data, using base R; we'll convert this later to spatial
data.  This is stored as a "shapefile" in zipped file

# Load a polygon shapefile into R using the 'terra' package's 'vect' function
polygons <- terra::vect('nationalparks_ak.shp')

# Convert polygons to an "sf" object  from SpatVector
polygons_sf <- sf::st_as_sf(polygons)

# Check and repair invalid geometries in polygons_sf
polygons_sf <- sf::st_make_valid(polygons_sf)

# Add polygons to the plot
plot(st_geometry(polygons_sf), border = "green", col = NA, add = TRUE)
```



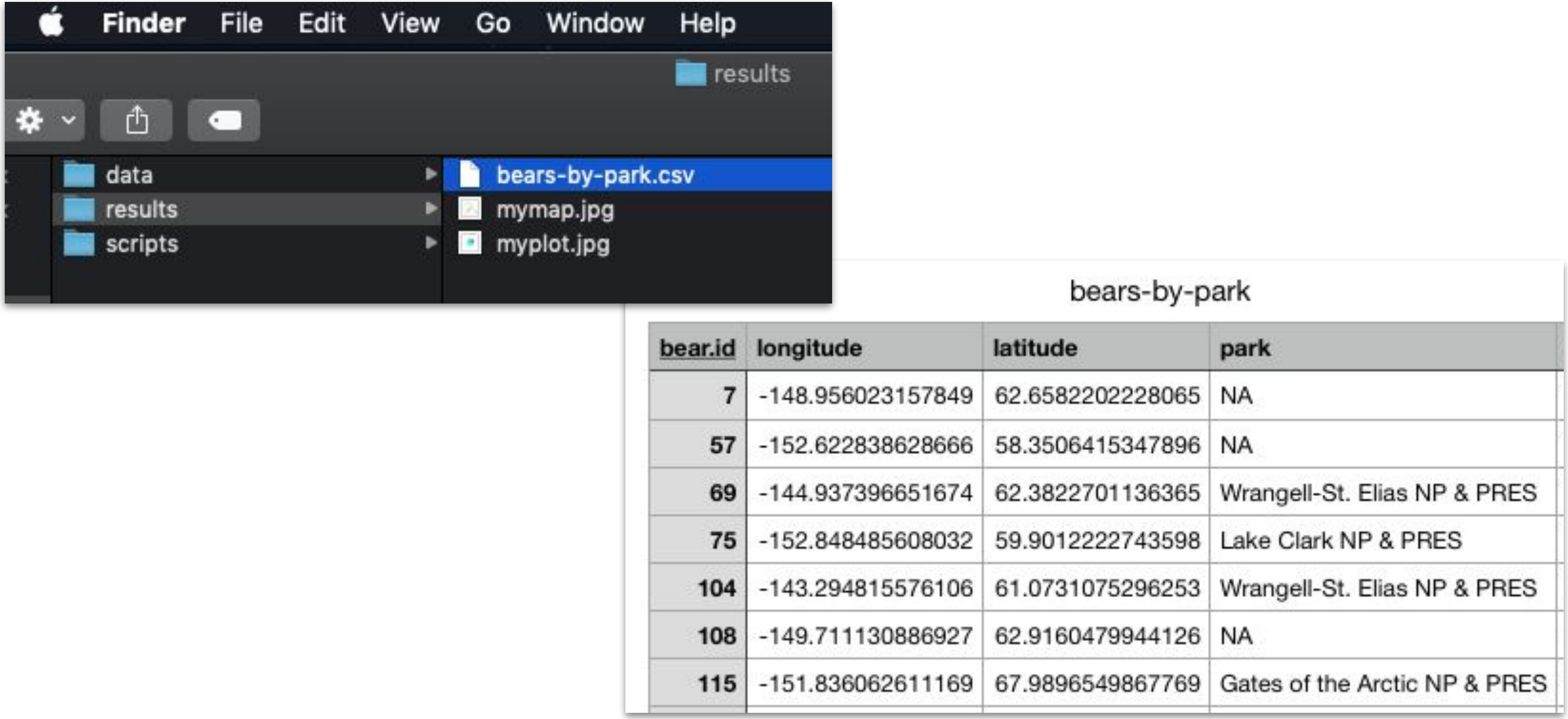
Making a map in R, display spatial data

If everything went well, the Plots window in R should now look like this, a map of Alaska with a bunch of point locations on it!

A little more than ten lines of code, and we have spatial data displayed in R Studio



Check on the Resulting CSV Output:



bears-by-park

bear.id	longitude	latitude	park
7	-148.956023157849	62.6582202228065	NA
57	-152.622838628666	58.3506415347896	NA
69	-144.937396651674	62.3822701136365	Wrangell-St. Elias NP & PRES
75	-152.848485608032	59.9012222743598	Lake Clark NP & PRES
104	-143.294815576106	61.0731075296253	Wrangell-St. Elias NP & PRES
108	-149.711130886927	62.9160479944126	NA
115	-151.836062611169	67.9896549867769	Gates of the Arctic NP & PRES

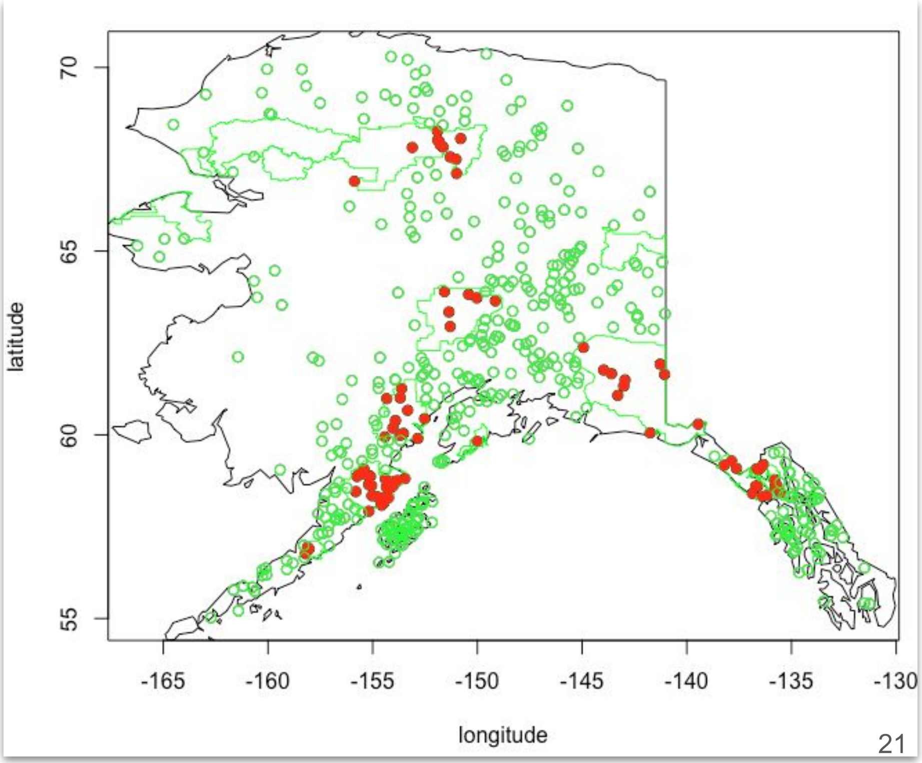
Spatial Analysis Visualization - Geographic Results

Map of *sightings* that are ***within*** a *park boundary*

Analysis layer shown, red circles inside, green circles outside

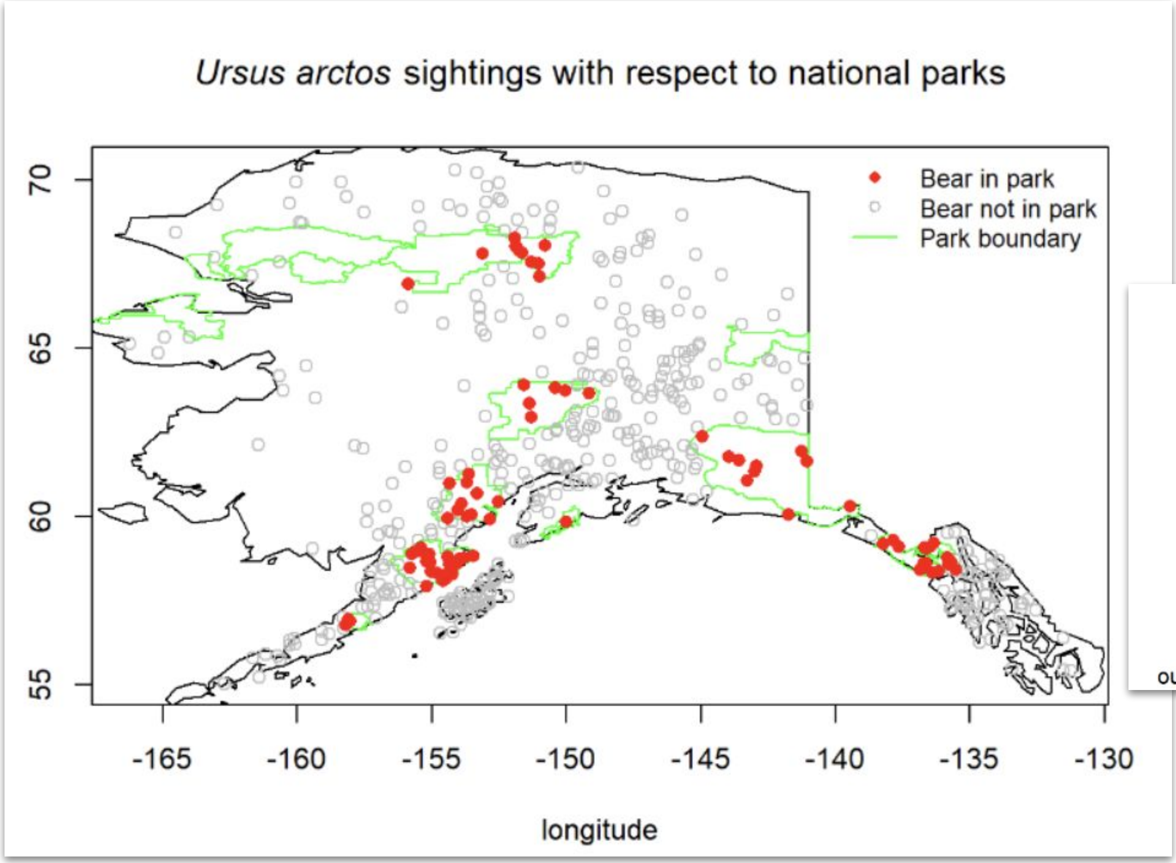
Original datasets are still intact

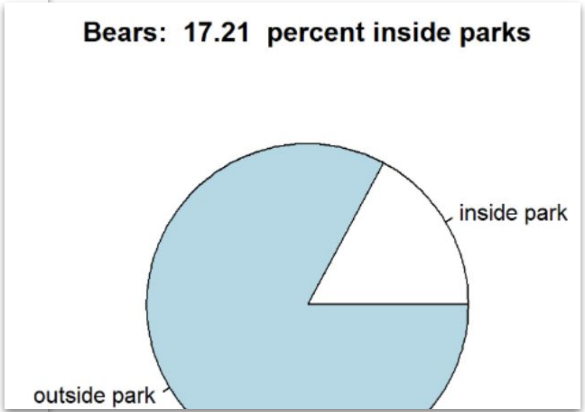
If a point landed within a polygon, the attributes of the polygon (park name, for instance) can be joined as a new column / attribute of the dataset



Add Legend and Title to Map

```
legend("topright", cex=0.85,
    c("Bear in park", "Bear not in park", "Park boundary"),
    pch=c(16, 1, NA), lty=c(NA, NA, 1),
    col=c("red", "grey", "green"), bty="n")
title(expression(paste(italic("Ursus arctos"),
    " sightings with respect to national parks")))
```


Ursus arctos sightings with respect to national parks
● Bear in park
○ Bear not in park
Park boundary
70
65
60
55
-165
-160
-155
-150
-145
-140
-135
-130
longitude


Bears: 17.21 percent inside parks
inside park
outside park

Questions?

Please provide us with feedback & constructive criticism, we use that to design future workshops

https://dartgo.org/feedback

As always, feel free to reach out anytime.

Thanks for attending our workshop!

Materials: https://dartgo.org/r-data-viz