Reproducible Research with Spatial Data

A part of the Reproducible Research Workshop Series

October 20, 2020

www.dartgo.org/RRADworkshops

A roadmap for this workshop

- Basics of Reproducible Research, applied to spatial data
- Why Reproducible?
- 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 https://rc.dartmouth.edu and click "contact us"

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 Reproducible?

- Saves time
- Saves money
- Prevents wasted efforts
- Increased scientific credibility
- Often required by granting agencies and organizations
- Allows researchers to innovate at a faster rate with fewer errors

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

Tools of the trade for reproducibility with spatial data

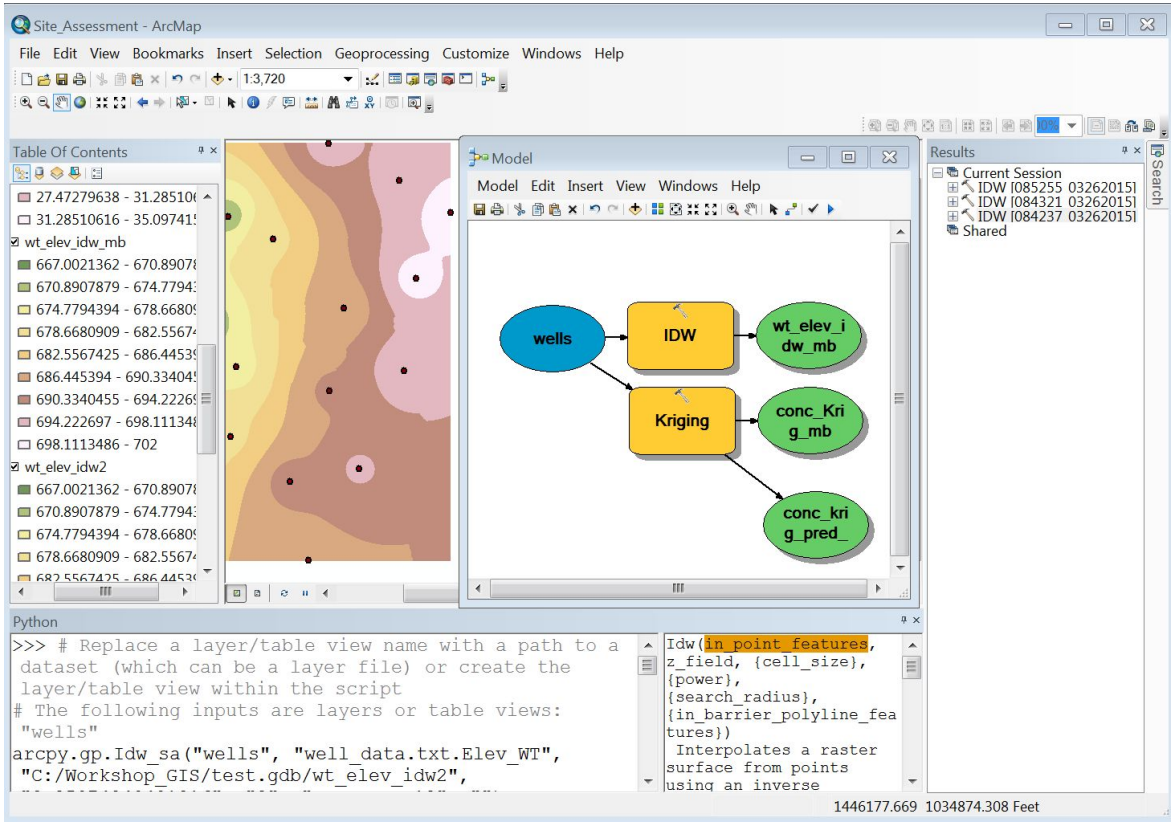
- Proprietary:
  - ArcGIS Desktop with Python Scripting
  - ArcGIS Pro with Python Scripting
  - MapInfo
- Open-Source:
  - R Project with spatial libraries
  - Python with spatial libraries
  - GDAL
  - QGIS
  - GRASS GIS
  - OSGEO https://www.osgeo.org/initiatives/geo-for-all/ or www.geoforall.org
  - https://www.osgeo.org/initiatives/geo-for-all/in-your-research/

ArcGIS Desktop with Modelbuilder, Python and ArcPY

ArcMap contains a rich point-and-click interface

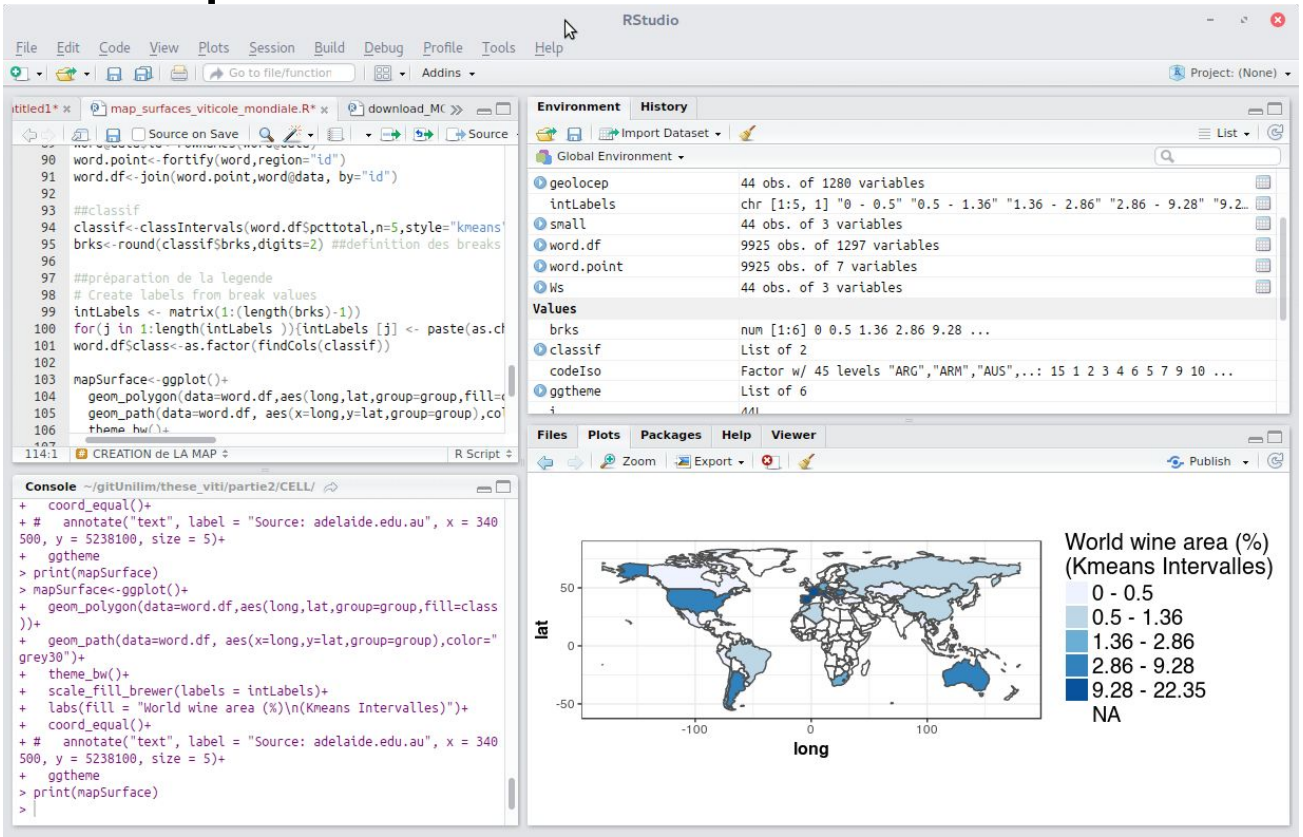
ArcMap also has Modelbuilder and the "Results" window, useful for graphical process development

And ArcMap supports Python scripting, right in the interface

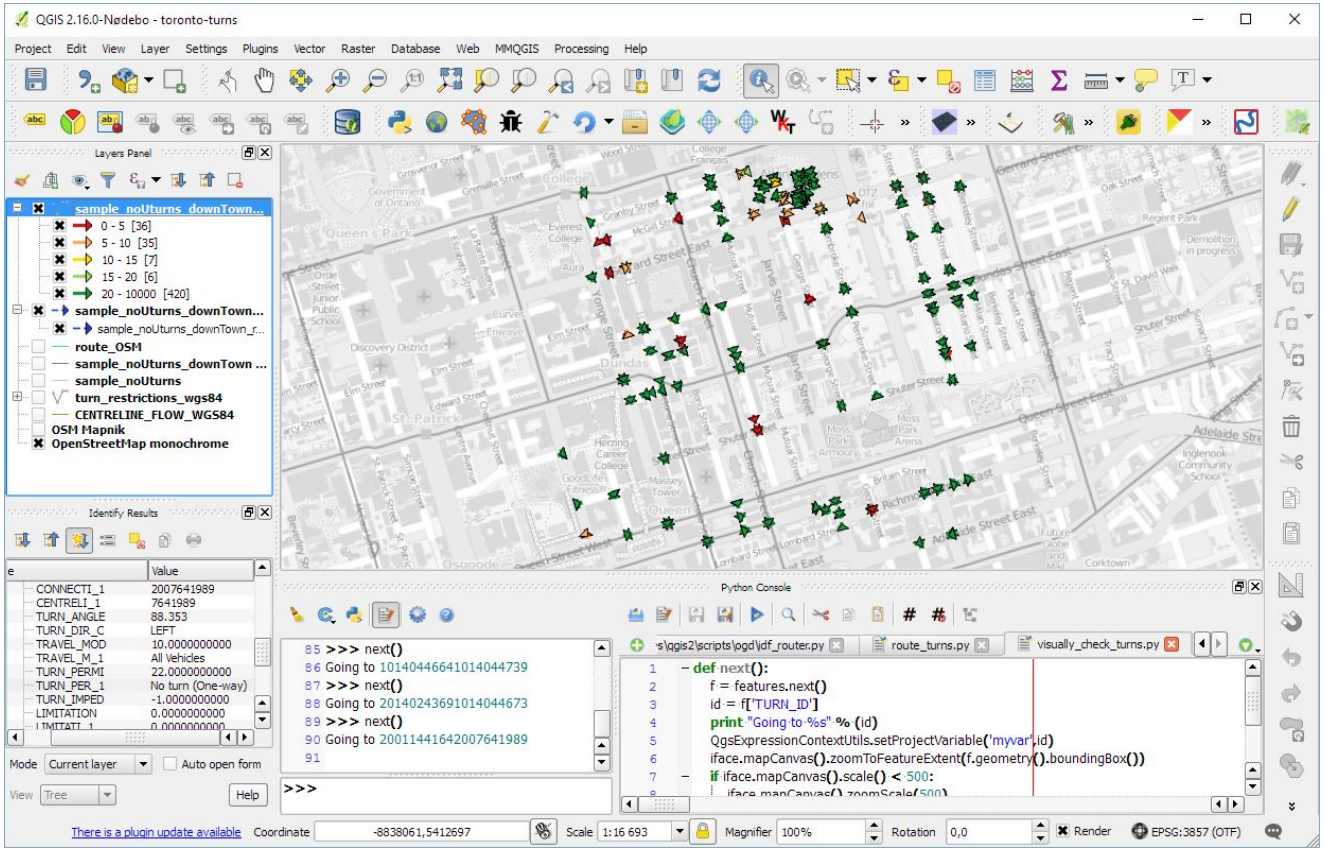


R with open-source spatial libraries

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

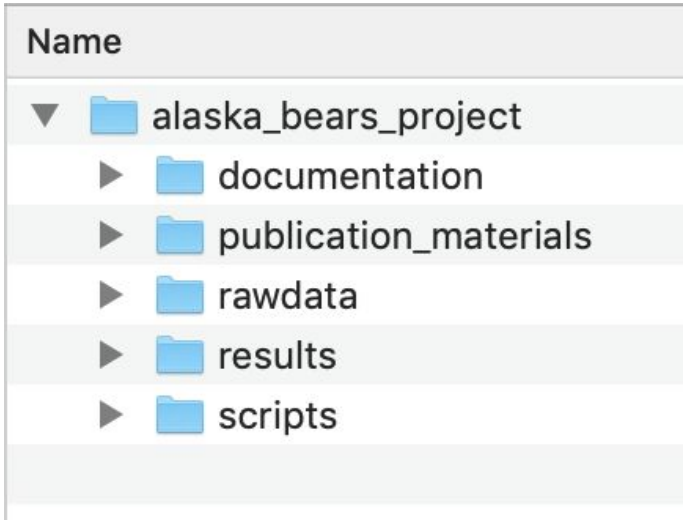


QGIS with Python Console



Data management - Sample folder structure

- Projectname
  - rawdata (this folder can be read-only)
  - results
  - scripts (analysis scripts)
  - publication_materials



- 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

Process Outline (human-readable/readme format)

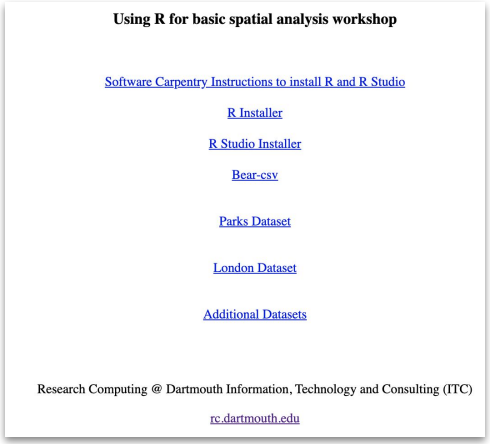
Process outline and pseudo code:

- 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
- Analyze the data to find out if each bear is inside or outside of a park
- Report 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 Analysis Example

http://dartgo.org/itcworkshop

- 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)



Using R for basic spatial analysis workshop

Software Carpentry Instructions to install R and R Studio

R Installer

R Studio Installer

Bear-csv

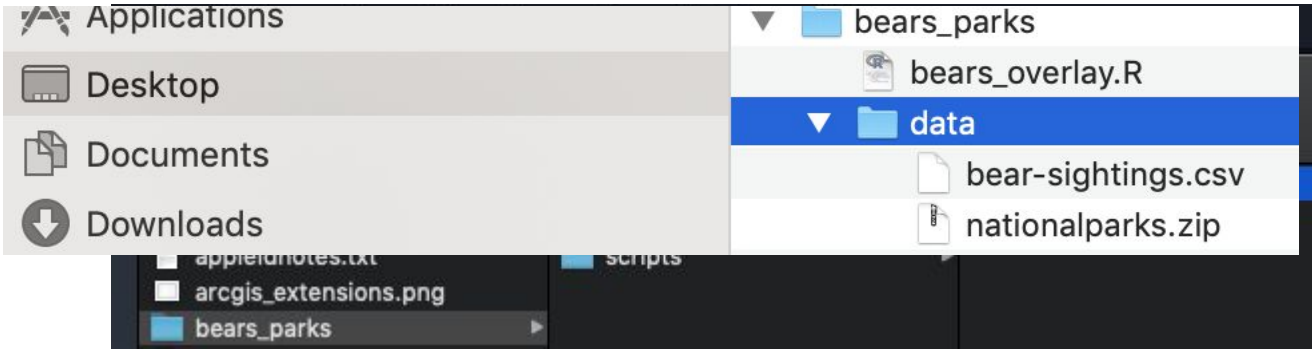
Parks Dataset

London Dataset

Additional Datasets

Research Computing @ Dartmouth Information, Technology and Consulting (ITC)

rc.dartmouth.edu



Some useful R packages for spatial data

ggplot2
ggmap - map plotting package
osmdata - open street map data, geocode an address, download map tiles
rgdal - R version of geospatial data abstraction library (gdal works in Python also)
- rgdal has tools like spatial overlay

sf - simple features
tidyverse
tmap - thematic maps for R
tmaptools - read and process spatial data
maps
maptools
sp

Reproducible Spatial Analysis using R & R Studio

```
getwd()     # tip, use Control Return key combination to run the line from a script
setwd('~/Desktop/bears_parks/')          # tip, always comment your code!

# pc users: setwd("C:/Users/f002d69.NAUSET/desktop/bears_parks/data")

# create a string variable for our results directory
resultsdir <- paste(getwd(),"/results", sep = "")

# built-in "unzip" function
unzip(zipfile = "data/nationalparks.zip", exdir = resultsdir)

# built-in read.csv function
bears <- read.csv('data/bear-sightings.csv')
```

Reproducible Spatial Analysis using R & R Studio

```
# use sp package's coordinates function to set the
#  coordinates for the bears csv, and convert it in
#  to a "formal class spatialpointsdataframe

install.packages("sp")
#   or, to avoid installing each time the script is run, install if needed:
if(!require("sp")) install.packages("sp")

library(sp)

# coordinates" function from the "sp" package
coordinates(bears) <- c('longitude','latitude')
```

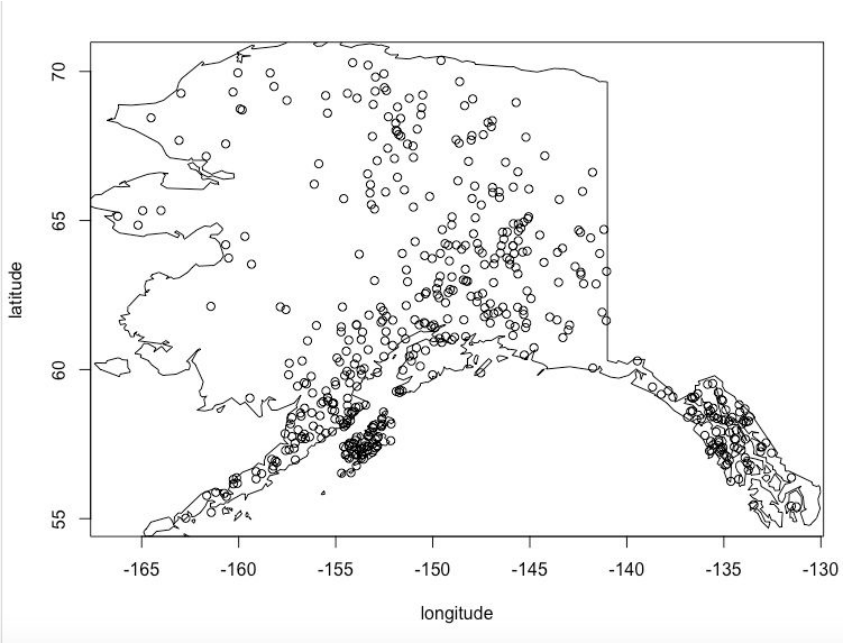
Using R with Spatial Data

```
# let's see if the "bear-sightings" csv has valid coordinates in it - do the coordinates land in Alaska?
# this line checks to see if a package is installed already
"maps" %in% rownames(installed.packages()) == TRUE

if(!require("maps")) install.packages("maps")
library(maps)
# plot the coordinates of the bears
plot(coordinates(bears))
# use the "maps" package to add a coarsely-drawn map layer for context
map("world", region="usa", add=TRUE)  # from the "maps" package
```

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



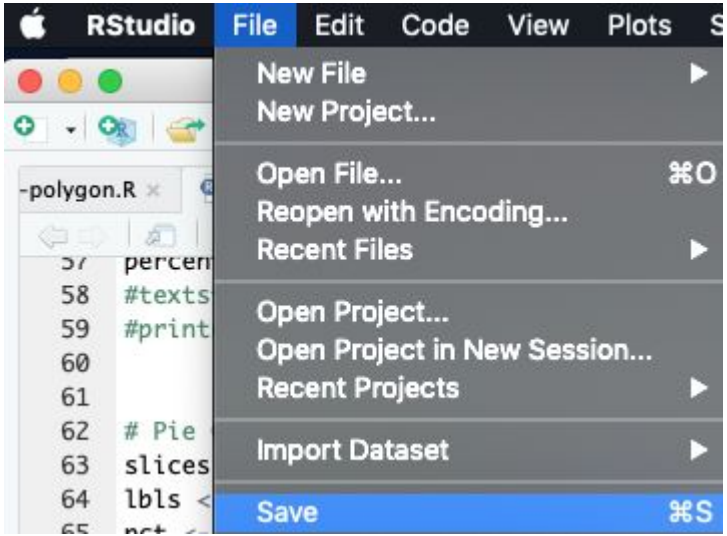
Saving an R Script

We've done some good preliminary work, lets save the script

File > Save >

save the file in desktop > bears_parks > scripts > bearsspatial_20201020.R

This could also be a point where you push a version out to a version-control system like GitLab, Github, SVN, etc



Spatial Analysis - I

```
"rgdal" %in% rownames(installed.packages()) == TRUE
library(rgdal)
# GDAL = geospatial data abstraction library. Open source!
# Check out their website at https://www.osgeo.org/ and https://www.gdal.org/

getwd()
setwd("/Users/stevegaughan/Desktop/bears_parks/results")
# OGR stands for Open Geographic Reference
parks <- readOGR('.', '10m_us_parks_area')
ourprojection <- proj4string(parks)
print(ourprojection)
```

Spatial Analysis - II

```
# use the sp package's proj4string to set the projection of

#  the new "bears" spatial data frame to the same projection as the parks dataset

proj4string(bears) <- proj4string(parks)

# the 'over' function

insidePark <- !is.na(over(bears, as(parks, "SpatialPolygons")))

# Spatial analysis goal #1 - get the fraction of bears inside a park!

mean(insidePark)
```

Generating Traditional Results - Sending Output to the Console

```
# use 'cat' to concatenate a string and send it to our console
# did we all get the same result?

cat("Percent inside parks: ", 100*mean(insidePark), ' percent ')

# let's create a visualization for output:

slices <- c(mean(insidePark), 1-mean(insidePark))

lbls <- c("Bears in the parks", "Bears outside the parks")

pct <- round(slices/sum(slices)*100,2)
```

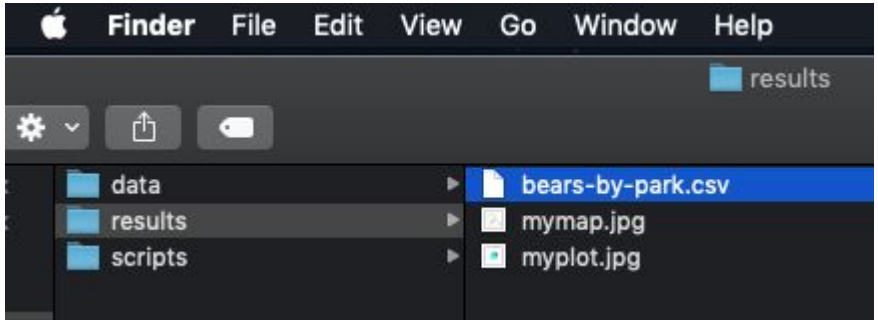
Generating Traditional Results - Creating a Chart, Saving Output Documents

```
lbls <- paste(lbls, pct)                # add percents to labels

lbls <- paste(lbls,"%",sep="")      # add % to labels

pie(slices,labels = lbls, col=rainbow(length(lbls)),

    main="Bear Sightings")

# let's save a copy of this great plot in to our 'results' folder

dev.copy(jpeg,'../results/myplot.jpg')

dev.off()     # dev.off tells R Studio to send the plot out rather than plot it
```

Generating Traditional Results - Save to CSV

```
# use 'over' again, this time with parks as a SpatialPolygonsDataFrame

# store the park name as an attribute of the bears data

bears$park <- over(bears, parks)$Unit_Name

# write a csv file with bear names and the name of the park (if found)

write.csv(bears, "../results/bears-by-park.csv", row.names=FALSE)
```

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

Generating Geographic Output

```
# hang in there, almost done!

library(maps)

plot(coordinates(bears))

map("world", region="usa", add=TRUE)  # from the "maps" package

plot(parks, border="green", add=TRUE)
```

Saving Geographic Output to a Document

```
# set the colors for the points inside and outside the park

points(bears[insidePark,],pch=16,col="red")

points(bears[!insidePark,], pch=1,col="green")

# send the plot to our results folder

dev.copy(jpeg,'../results/mymap.jpg')

dev.off()
```

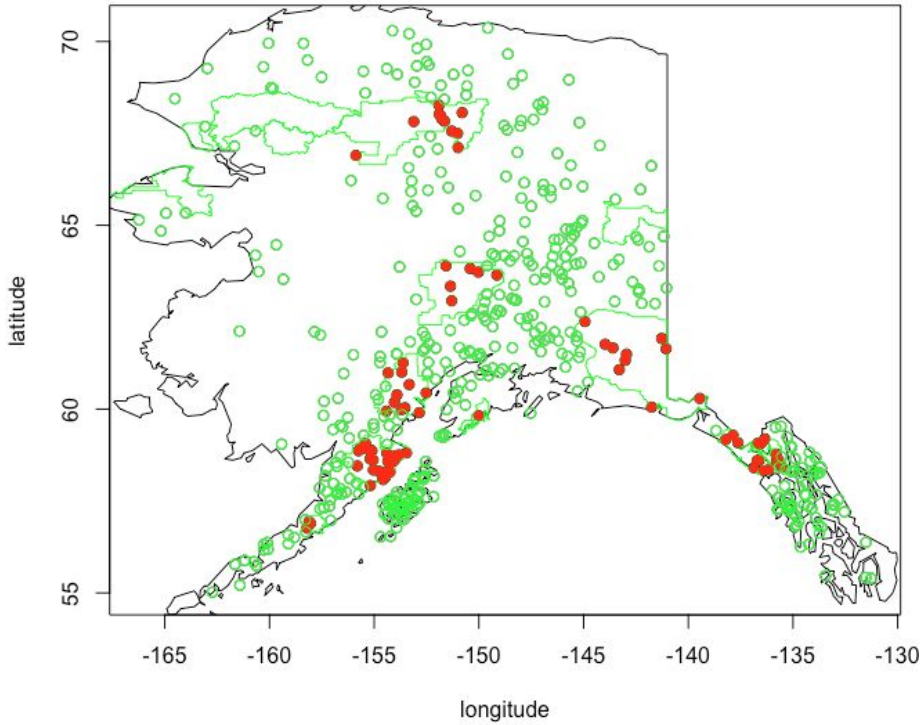
Spatial Analysis Visualization - Geographic Results

If all went well, map should look like this

Our analysis layer is shown

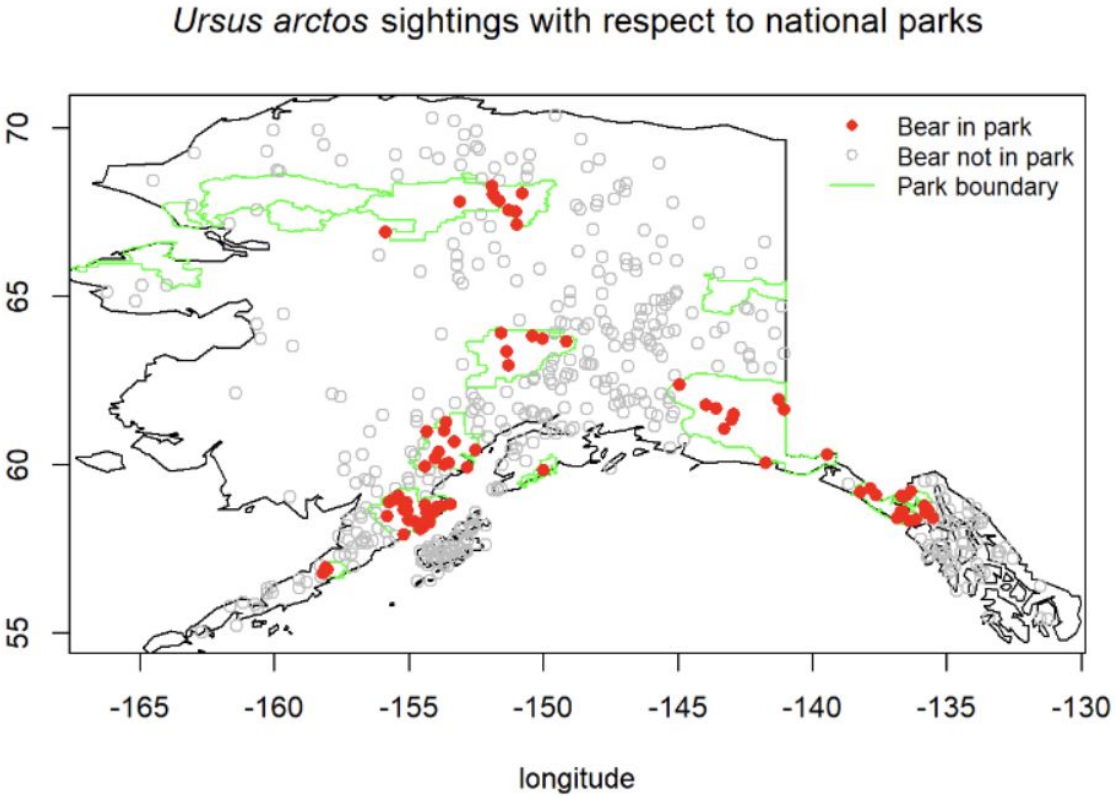
Our original datasets are still intact

Our research results are in a separate folder



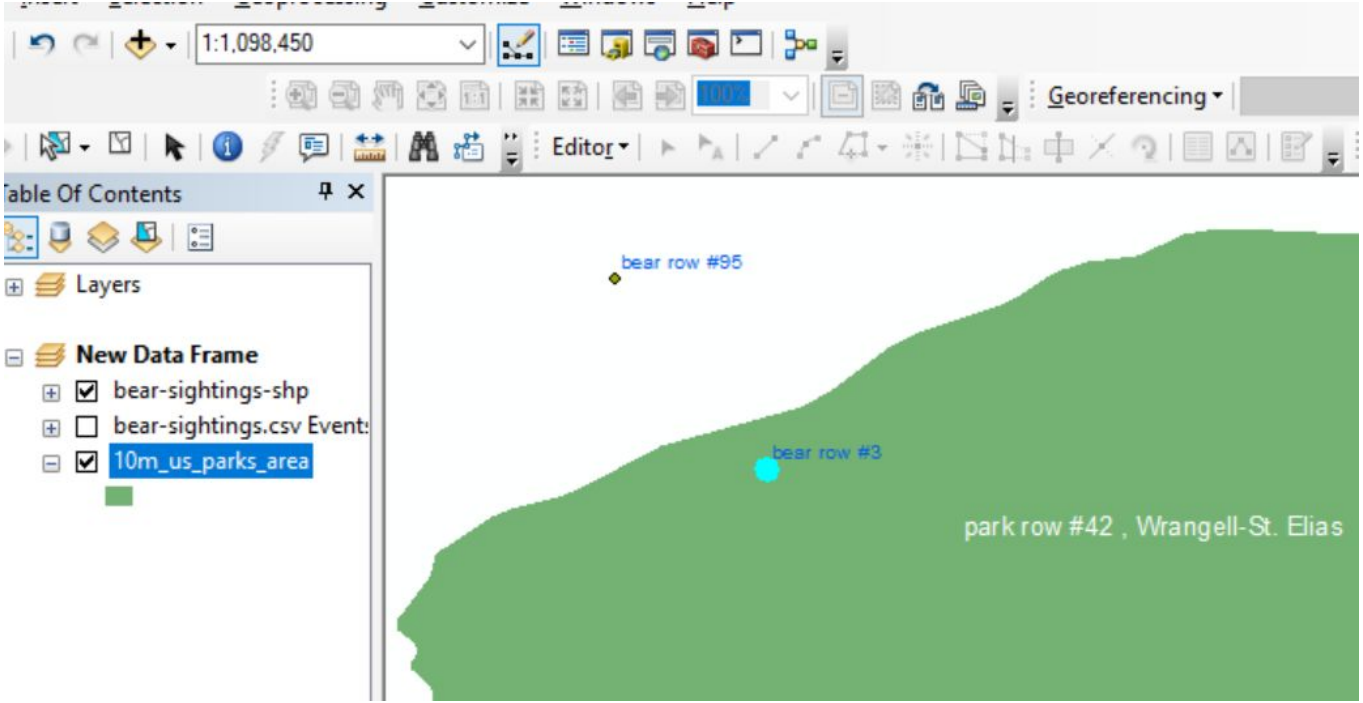
Optional, Add Legend and Title to Map

```
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

Dataset overlay in ArcMap

ArcMap



R > R Studio > R Markdown > HTML

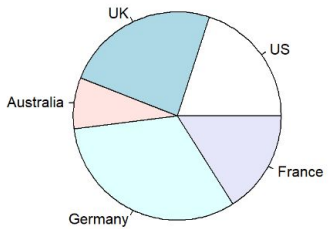
R Studio, along with R Markdown, can generate an HTML page with code, comments, tables of data, graphs, charts and even maps. Here is an example using some built in R datasets, and our "bears and parks" analysis

R Markdown

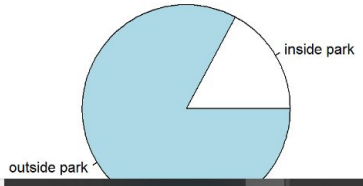
~/workshops/Rspatialscripts/ReproducibleResearchWithSpatialDataMarkdown.html

ReproducibleResearchWithSpatialDataMarkdown.html | Open in Browser | Find | Publish

Pie Chart of Countries



Bears: 17.21 percent inside parks



~/workshops/Rspatialscripts/ReproducibleResearchWithSpatialDataMarkdown.html

ReproducibleResearchWithSpatialDataMarkdown.html | Open in Browser | Find | Publish

```
##  Linking to sp version: 1.3-1
```

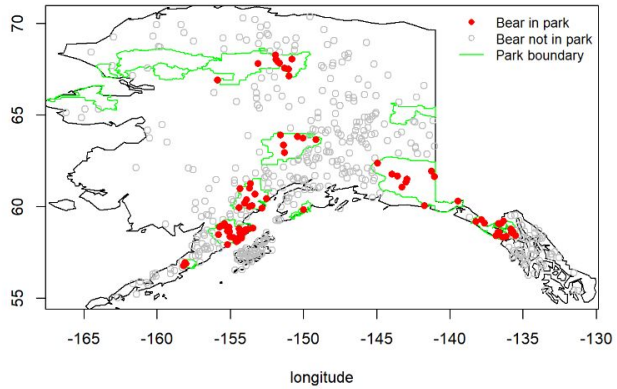
```
Loading required package: maps
```

```
Warning: package 'maps' was built under R version 3.5.2
```

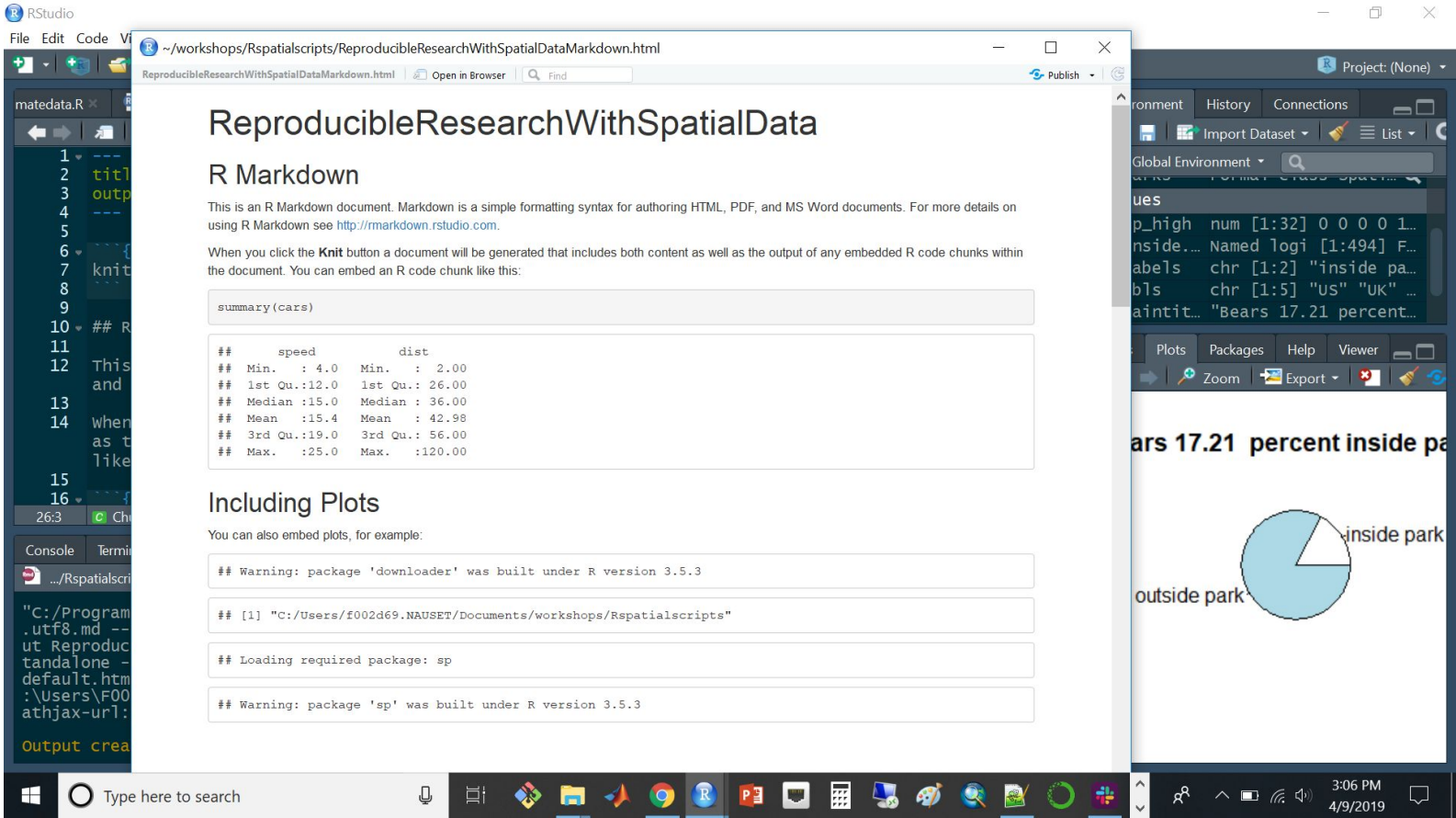
```
OGR data source with driver: ESRI Shapefile
Source: "C:\Users\f002d69.NAUSET\Documents\workshops\Rspatialscripts", layer: "10m_us_parks_area"
with 61 features
It has 8 fields
```

```
[1] 0.1720648
```

Ursus arctos sightings with respect to national parks



Markdown Document



Comparison of Results - Did it work?

Comparison and guts of the analysis

In R, we see the first row (bear row number) and the second row, park row number. So, R tells us that the bear row #3 is inside the park row #42

```
> print(proj4string(parks))
[1] "+proj=longlat +datum=WGS84 +no_defs +ellps=W
> print(over(bears,as(parks,"SpatialPolygons")))
  1   2   3   4   5   6   7   8   9  10  11  12
 NA  NA  42  38  42  NA  59  NA  NA  37  NA  NA
 26  27  28  29  30  31  32  33  34  35  36  37
 NA  NA  NA  NA  NA  38  NA  NA  59  43  NA  NA
```

Review

Spatial Analysis Reproducibility Tools:

- R with open source libraries
- GDAL
- ArcGIS Desktop with Modelbuilder, ArcTooolbox, Python, ArcPy
- Python with proprietary ArcPy library
- Python with open source libraries, GDAL, PySal

Resources

- R spatial view: https://cran.r-project.org/web/views/Spatial.html
- Spatial data science: https://rspatial.org/

Questions?

Thanks for attending our workshop!