Lab 4. Spatial Data in R.

For this lab we are going to explore spatial descriptive statistics in R. We will learn to:

**Part 1**

1. Bring vector spatial data (points, lines, polygons) into R.

2. Map these features.

3. Convert polygons to points.

4. Derive and map the spatial central tendencies of a data base.

**Part 2**

5. Summarize data by state and make a choropleth map.

6. Standardize a variable

**There are 110 points total within 12 questions**. Each question in the code is marked and underlined.

**Part 1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Step 1.** In order to read spatial data into R and turn them into Spatial\* family objects we require the

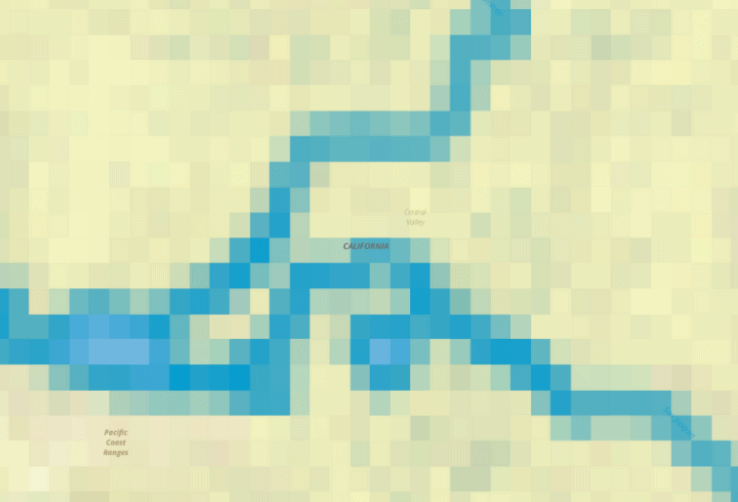
numerous spatial data packages. First, install the packages for spatial data manipulation. You will need to install packages:

* sf
* tidyverse
* dplyr
* terra

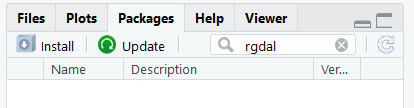
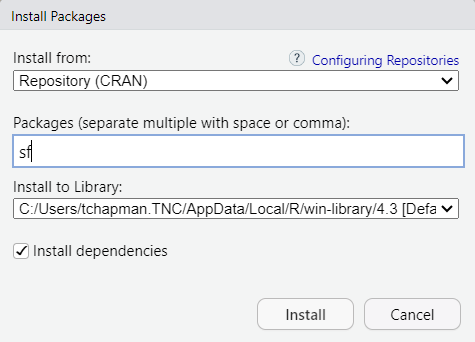
Go ahead and install them all now. Say yes to the pop-up box. These take some time. Add them to your code as library().The '**tidyverse**' is a set of packages that work together because they share common data representations and 'API' design. Tidyverse was built by scientist to streamline data wrangling and graphics. Dplyr is a package to help manipulate data.frames and is part of the tidyverse package series. We can use functions in dplyr to conduct data analyses on spatial data attributes, as well.

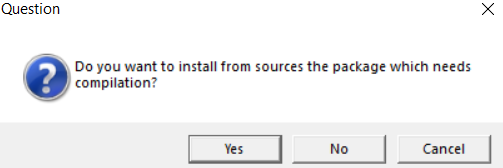
There are many spatial packages for R. We will use a few of them. Before we discuss how to import spatial data into R, let’s briefly introduce you to the two main types of data: vector and raster.

Vector data represents geographic data symbolized as points, lines, or polygons. Raster data represents geographic data as a matrix of cells that each contain an attribute value (even if that value is ‘no data’). Each pixel has a value. In vector data, not each location needs a value. Raster cells are all the same size (called a resolution). Vector polylines or polygons can be a variety of sizes and be in the same shapefile. Vector image on the left and a raster image on the left.



To work with vector data in R, we will use the Sf package. This package creates a ‘sf’ (simple features) object having a spatial object and a data frame. You will use the class() function routinely to determine what sort of object in R you have created. This will help you understand what is being created and what you can do with your object.



There is a lot to learn about the sf package and spatial data. Here is a nice summary blog: <https://joshuamrosenberg.com/post/2020/06/08/basic-sf-functionality-for-analyzing-and-plotting-geospatial-data/>

Open the lab 4 script in R.

You will fill in the R script and save it as “Lab 4 Lastname” in your lab 4 folder.

This is what you will turn in.

All text highlighted in gray in this word document relates to information or code in you R script. Once you install packages, you just use library() function to call them. You only need to install them once.

## load packages

library(sf)

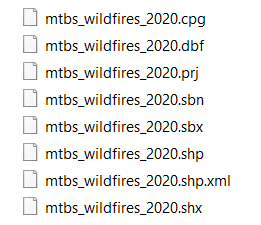
library (tidyverse)

library(terra)

library(ggplot2)

library(units)

In this lab we will explore polygons of wildfires in the US that I downloaded from Monitoring Trends in Burn Severity ( <https://www.mtbs.gov/direct-download>). I have filtered the shapefile to only include wildfires in 2020. A shapefile comes with six to eight associated files.



They should all be in one directory and that is what R expects. Place all your lab 4 data in your lab 4 folder within GEOG 3023 within a lab 4 data folder. Download the wildfires and make sure you put them in your working directory inside a data folder. Success in coding requires data management and that starts with data organization and knowing where everything is getting placed.

**Step 2** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

When reading in a shapefile with the sf package.

You need to know the name of shapefile. Our lab 4 shapefile is named ‘mtbs\_wildfires\_2020’.

You need to know the name and location of the folder that contains all the shapefile parts.

I placed my shapefile in a ‘data’ within my working directory. (e.g., ‘C:\\Users\\tchapman.TNC\\Desktop\\2022 Fall GEOG stats\\Labs\\Lab 4\\data’).

Remember you have to change the format of the slashes for R to read it.

Here we go.

## set working directory -change this to your working directory.

setwd('C:/Users/tchapman.TNC/Desktop/2022 Fall GEOG stats/Labs/Lab 4/data')

Note – if you are working on a mac, this will appear differently.

## read in the wildfire data

fires <-st\_read("mtbs\_wildfires\_2020.shp")

## confirm the spatial object

class(fires)

Note that you have two objects: sf and a dataframe. The dataframe is the table with all the information that relates to each polygon in the spatial data.

We can create a simple plot of the polygons with the plot command. Since this can take a bit of time to plot all the polygons, we are going to map only the first 100 polygons of the data.

## use base R plot

plot(fires[1:100,])

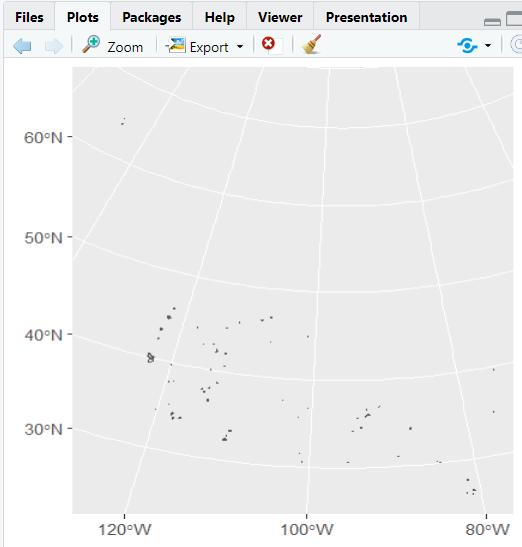
When using ggplot to plot the polygons, you can pass to ggplot the sf geometry of the shapes. Ggplot will automatically know that the geom\_sf contains the geographic information. geom\_sf is the geometry of the data.

## use ggplot2

ggplot(fires[1:100,]) +

geom\_sf()

Use the Zoom on the plot to better see the polygons.



Step 3\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We will calculate the area of each fire polygon in hectares.

st\_area(fires)

## the console will print out the area of each polygon. At the top of the console print out (scroll up), you will see the units for the area (Units: [m^2])

## Put the units in hectares. 1 ha = 100x100m

st\_area(fires)/10000

Now let’s add a field to our wildfire database that is equal to the area of each polygon and then check our work. Since the sf object has both a spatial object and a dataframe, you can add a field with the $ sign.

## add a field to the dataframe equal to the area in ha.

fires$area <- myArea

#check your work

head(fires)

When you check your work, what are the units of the area field? Let’s fix this.

We can use a function in a package called units – which was added as a package when we called the sf package. There are dependencies in packages so that you both install and download several packages when you install one package.

To make sure that we are using a function from the units package, we will use :: notation. This tells R which package to use for running a function. If two packages both have functions called filter, we would want to specify package1:: filter() or package2 ::filter().

##Create a field in hectares and add it to fires

myArea <- units::set\_units(st\_area(fires), hectare)

fires$area <- myArea

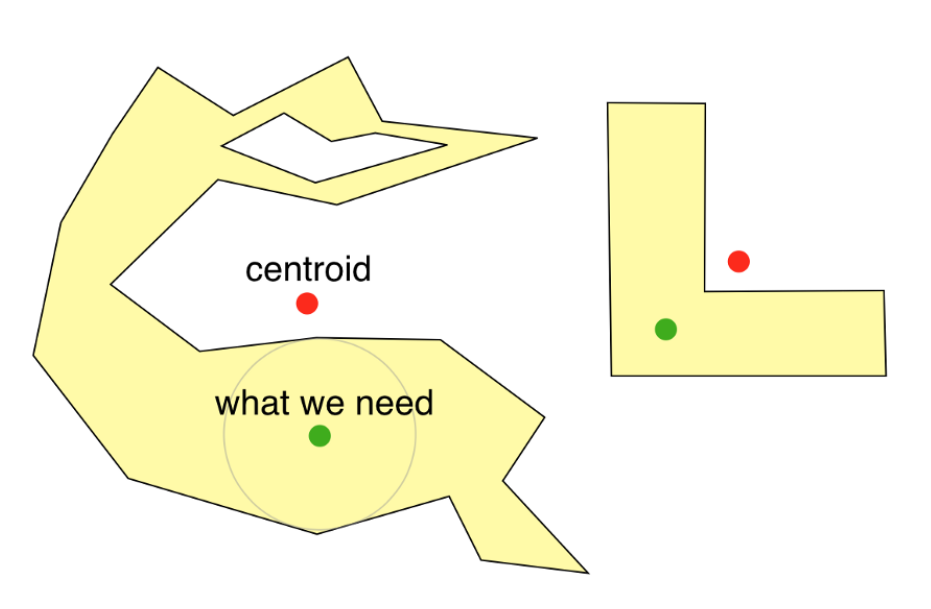
#check your work again

head(fires)

Step 4\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Let’s convert the polygon data to points so we can calculate some descriptive statistics on the data.

Now we can run the function to convert polygons to points. Centroids are the geometric center of a shape, which may not always fall inside the shape.



Point on surface will always put the point inside or on the shape.

## convert polygons to points. Try points on surface.

firePoint <- st\_point\_on\_surface(fires)

## check the dimensions

dim(firePoint)

Now that we have points in stead of polygons, we can more easily map the fires. Answer the question in the lab.

Step 5\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Before moving forward, we can check the coordinate systems and projections of our data.

Coordinate systems enable geographic datasets to use common locations for integration. A coordinate system is a reference system used to represent the locations of geographic features, imagery, and observations such as GPS locations within a common geographic framework. A global or spherical coordinate system such as latitude–longitude. These are often referred to as *geographic coordinate systems.*

A projected coordinate system based on a map projection such as transverse Mercator, Albers equal area, or Robinson, all of which (along with numerous other map projection models) provide various mechanisms to project maps of the earth's spherical surface onto a two-dimensional Cartesian coordinate plane. Projected coordinate systems are sometimes referred to as *map projections*.

The units of the coordinates will be in the appropriate unit of measurement for whatever coordinate reference system the data are in.

## Get the projection of the firePoints.

st\_crs(firePoints)

Explore the data. What are the attribute column names?

We are going to calculate the mean center of the 2020 wildfires in the US. To do this we need to use our descriptive statistics on the X coordinates (the longitude) and the Y coordinates (the latitude).

We will use a sf function called st\_coordinates. We will explore the output of st\_coordinates and get it into a format where we can run statistics.

##get the coordinates

st\_coordinates(firePoints)

##explore the class of the object

class(st\_coordinates(firePoints))

dim(st\_coordinates(firePoints))

Scroll up to the top of the console print of st\_coordinates. See how there is an X column and a Y column.

The X column can be called by using st\_coordinates(firePoints)[,1].

The Y column can be called by using st\_coordinates(firePoints)[,2].

Now you will be asked to calculate the mean of each and assign it to an object.

Step 6 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Now we will map the mean center of wildfires.

You have created MCx and MCy – the means of the X coordinates and Y coordinates respectively.

We need to create a dataframe in order to create an sf object.

## Create a dataframe by combining the X and Y mean centers.

(meanCenter <- data.frame(cbind(MCx, MCy)))

Now we can turn this dataframe into a sf object.

## Create a dataframe by combining the X and Y mean centers. cbind = column bind. You are joining together two columns of data.

(meanCenter <- data.frame(cbind(MCx, MCy)))

## Create sf object

(meanCenter\_sf <- st\_as\_sf(meanCenter,coords=c("MCx", "MCy"), crs = st\_crs(fires)))

##Note - we give the function st\_as\_sf the dataframe called meanCenter and we tell it that the coords are in the columns named "MCx" and "MCy".

##Remember when we wrap our line of code in (), it will print the result in the console. This is very useful when you want to see what you are doing.

This is how you can take coordinates and make a sf object. Since we want to map this point, we will also need to make fields in the point’s table that show the x and y coordinates.

#Create fields with X and Y coordinates

meanCenter\_sf$MCx <- MCx

meanCenter\_sf$MCy <- MCy

## map the coordinates of the fires and the Mean Center.

ggplot(firePoints) +

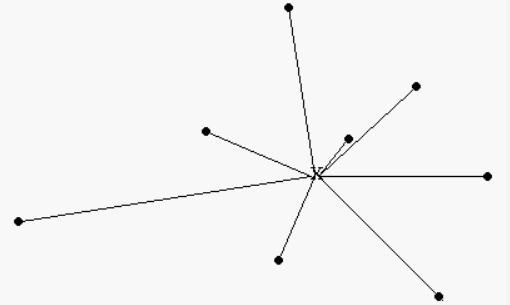
geom\_sf()+ geom\_point(data = meanCenter\_sf,aes(x = MCx, y = MCy), size = 7, fill="red", shape = 23)

##http://www.sthda.com/english/wiki/r-plot-pch-symbols-the-different-point-shapes-available-in-r

**Step 7** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We will now calculate the central feature. The central feature is the point with the shortest distance to all other points. Unlike the mean center, the central feature is an actual point in the database.

We will link together a few steps to calculate the central feature. First, calculate the distance (Euclidian) between each point and every other point. This will make a huge table since we have 820 points. To see how this works, let’s first look the first ten points in the database. The distance between a point and itself will be zero. Each column is the distance between that point and each row is a different point. So row 10 and column 10 has a value of zero since the distance between a point and itself is 0.



##Calculate the distance between ten points to get a sense of what the st\_distance function does.

(distSubset <- st\_distance(firePoints[1:10,]))

Look at the table. Now we can take the average for each row or for each col and we would have the average distance that each point is from every other point. We can use a base R function called colMeans.

##Now Calculate the distance between all points

dist <- st\_distance(firePoints)

## We can take the mean of each column to determine the average distance for each point and all other points.

(distMeans <- colMeans(dist))

Now we want to know the central feature which has the shortest distance to each point (e.g., this point would have the minimum average distance in this table).

## If we then take the fire point with the minimum value, we have the point that is the median center.

(distMin <- min(colMeans(dist)))

Now we know the minimum distance. We need to know *which point* contains this minimum value.

## Find the firePoint with the minimum distance

which(distMeans==distMin)

Now we know which row number in fire points has the minimum distance to all other points.

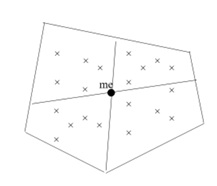
You will be asked to map this central feature.

**Step 8**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Now we are going to map the median center as our final step in exploring spatial central tendencies.

The median center is the location that minimized all Euclidian distances to that point. It is not an existing point in the database (like the central feature). It is a location turned into a point – like the mean center – that is created by minimizing the distances to all points. There is a fairly straight forward way to create the median center.

Plot the points on a map. Draw a line North-South leaving an equal number of points on each side (left/right) Draw a line East-West leaving an equal number of points on each side (top/bottom) The intersection of both lines is the Median Center. It can be used to find a suitable location for something that needs to be centrally located. The Median Center will gravitate towards an area with the most features. The Median Center is good for finding the most accessible location.



We will need to find the median value for x and Y coordinates. This will result in what is a ‘line’ at the midpoint of the x values and a ‘line’ at the midpoint of the Y values. We don’t need to actually map the lines, since we can just use the coordinates.

Since you know how to access the coordinates (st\_coordinates) of a sf object. You can figure this one out on your own. Answer the question in the R script.

**PART 2** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

In part 2, we will look more in depth at the size of wildfires and which states have the most area burned.

Let’s start by always exploring the variable of interest. Area. We calculated this area in the first part of the lab. Answer some question sin the lab about the distribution of the variable area.

If you forgot that firePoints had a variable named ‘area’, you can look at the fields in firePoints using head(firePoints).

We can take our mapping to the next level by now mapping the attribute “area” – our column with the area data.

## Create a proportional dot map map the coordinates of the fires.

## for the aes() function, you need to add the coordinates themselves and then scale them by the Area\_ha attribute.

## We can then add a legend with the scaled symbols.

## See how I added a title and labelled the x and y axes.

head(firePoints)

#Note that in the below graph, we had to convert the area in meters to just a number using as.numeric(). This function converts the vector to numbers.

ggplot() + geom\_point(data = firePoints, aes(x=st\_coordinates(firePoints)[,1], y=st\_coordinates(firePoints)[,2],size=as.numeric(area))) +

scale\_size\_continuous(name="area")+ ggtitle("Area of US Wildfires in 2020") + xlab("Longitude (m)") + ylab("Latitude (m)")

Answer the question in the R script.

**Step 9**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The second topic that we must begin to understand in R is how to summarize and aggregate data and standardize data. For this next section we will:

* 1. calculate the area of wildfires in each US state and map the result, and
  2. standardize the area of wildfires in each state by state area.

First we must make sure that both the states and firePoints are in the same projection.

Answer the question in the lab.

Now we know that we need to project states to the same projection as firePoints. Having your spatial data in the same projection ensures that your spatial analyses are correct.

##reproject using st\_transform. You need to provide the crs of firePoints

states <- st\_transform(states, crs = st\_crs(firePoints))

**Step 11**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Now we will sum the area of fire in each state. We will use the aggregate () function in R. The aggregate function splits data into subsets or groups and then computes summary statistics on those groups. The ‘by’ argument in the function is the group by which you want to compute the statistics and the ‘FUN’ argument is the statistics function you want to compute.

Since this is spatial data, if we group by ‘states’, R will use each polygon as a unique ID. The function we want to do is SUM. We will sum the area of fire in each state.

## Use the aggregate function. The FUN is the function. You can do many functions here.

(fireAreaSum<- aggregate(x = firePoints["areaFire"], by = states, FUN = sum))

## check the result

head(fireAreaSum)

dim(fireAreaSum)

## the result is a spatial polygon data frame.

class(fireAreaSum)

## Now we can join this new field back to states by making a field in states equal to our new areaFire calculation

states$areaFire <- fireAreaSum$areaFire

## check your work

head(states)

Now we will standardize wildfire Area by total state area. Since out states are different sizes, a state may have the most fire Area since it is the largest state. We want to control for state size so that our map better represents the wildfire area in each state. We could take this a step further and standardize by vegetation types (e.g., total area of forests).

Answer the questions in the lab.

----------------------------------------------------------------

Congratulations! You have finished lab 4.