GIS in R: intro to vector format spatial data - points, lines and polygons

Learning Objectives

After completing this tutorial, you will be able to:

- Be able to list and describe the attributes of 3 types of vector data: point, lines and polygons.
- Be able to open a shapefile in R using the `readOGR()` function.
- Be able to access the `crs` of a vector dataset in R.
- Be able to view the attributes for a shapefile layer imported into R.
- Plot one or more vector objects in `R` using `plot()`.

What you need

You will need a computer with internet access to complete this lesson and the data for week 4 of the course.

Download Week 4 Data (~500 MB){:data-proofer-ignore=" .btn }

About Vector Data

Vector data are composed of discrete geometric locations (x,y values) known as **vertices** that define the "shape" of the spatial object. The organization of the vertices, determines the type of vector that we are working with: point, line or polygon.

```
<a href="{{ site.baseurl }}/images/course-materials/earth-analytics/week-4/pnt_line_poly.png">
<img src="{{ site.baseurl }}/images/course-materials/earth-analytics/week-4/pnt_line_poly.png" alt="poi
<figcaption> There are 3 types of vector objects: points, lines or
polygons. Each object type has a different structure.
Image Source: Colin Williams (NEON)
</figcaption>
```

- **Points:** Each individual point is defined by a single x, y coordinate. There can be many points in a vector point file. Examples of point data include: sampling locations, the location of individual trees or the location of plots.
- **Lines:** Lines are composed of many (at least 2) vertices, or points, that are connected. For instance, a road or a stream may be represented by a line. This line is composed of a series of segments, each "bend" in the road or stream represents a vertex that has defined `x, y` location.
- **Polygons:** A polygon consists of 3 or more vertices that are connected and "closed". Thus the outlines of plot boundaries, lakes, oceans, and states or countries are often represented by polygons. Occasionally, a polygon can have a hole in the middle of it (like a doughnut), this is something to be aware of but not an issue we will deal with in this tutorial.

Data Tip: Sometimes, boundary layers such as states and countries, are stored as lines rather than polygons. However, these boundaries, when represented as a line, will not create a closed object with a defined "area" that can be "filled". {: .notice}

Shapefiles: Points, Lines, and Polygons

Geospatial data in vector format are often stored in a `shapefile` format. Because the structure of points, lines, and polygons are different, each individual shapefile can only contain one vector type (all points, all lines or all polygons). You will not find a mixture of point, line and polygon objects in a single shapefile.

Objects stored in a shapefile often have a set of associated `attributes` that describe the data. For example, a line shapefile that contains the locations of streams, might contain the associated stream name, stream "order" and other information about each stream line object.

- More about shapefiles can found on Wikipedia.

Import Shapefiles

We will use the `rgdal` package to work with vector data in `R`. Notice that the `sp` package automatically loads when `rgdal` is loaded. The `sp` or spatial package in R, supports lots of different types of spatial classes and objects. We will also load the `raster` package so we can explore raster and vector spatial metadata using similar commands.

```
# work with spatial data; sp package will load with rgdal.
library(rgdal)
# for metadata/attributes- vectors or rasters
library(raster)

# set working directory to earth-analytics dir
# setwd("pathToDirHere")
```

The shapefiles that we will import are:

- A polygon shapefile representing our California field site boundary,
- A line shapefile representing roads, and
- A point shapefile representing the field plots were vegetation was measured.

The first shapefile that we will open contains the study plot locations. To import shapefiles we use the `R` function `readOGR()`.

`readOGR()` requires two components:

1. The directory where our shapefile lives: `data/week4/D17-California/SJER/vector_data/`
2. The name of the shapefile (without the extension): `SJER_plot_centroids`

Let's import our field plot locations.

```
# Import a point shapefile: readOGR("path", "fileName")
# no extension needed as readOGR only imports shapefiles

sjer_plot_locations <- readOGR("data/week4/california/SJER/vector_data",
                       layer="SJER_plot_centroids")
## OGR data source with driver: ESRI Shapefile
## Source: "data/week4/california/SJER/vector_data", layer: "SJER_plot_centroids"
## with 18 features
## It has 5 fields
```

Data Tip: The acronym, OGR, refers to the OpenGIS Simple Features Reference Implementation. Learn more about OGR. {: .notice}

Shapefile Metadata & Attributes

When we import the `SJER_plot_centroids` shapefile layer into `R` (as our `sjer_crop_extent` object), the `readOGR()` function automatically stores information about the data. We are particularly interested in the geospatial **metadata**, describing the format, `CRS`, `extent`, and other components of the vector data, and the **attributes** which describe properties associated with each individual vector object.

Spatial Metadata

Key metadata for all shapefiles include:

1. **Object Type:** the class of the imported object.
2. **Coordinate Reference System (CRS):** the projection of the data.
3. **Extent:** the spatial extent (geographic area that the shapefile covers) of the shapefile. Note that the spatial extent for a shapefile represents the extent for ALL spatial objects in the shapefile.

We can view shapefile metadata using the `class`, `crs` and `extent` methods:

```
# view just the class for the shapefile
class(sjer_plot_locations)
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"

# view just the crs for the shapefile
crs(sjer_plot_locations)
## CRS arguments:
##  +proj=utm +zone=11 +datum=WGS84 +units=m +no_defs +ellps=WGS84
## +towgs84=0,0,0

# view just the extent for the shapefile
extent(sjer_plot_locations)
## class       : Extent
## xmin        : 254738.6
## xmax        : 258497.1
## ymin        : 4107527
## ymax        : 4112168

# view all metadata at same time
sjer_plot_locations
## class       : SpatialPointsDataFrame
## features    : 18
## extent      : 254738.6, 258497.1, 4107527, 4112168  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=11 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
## variables   : 5
## names       :  Plot_ID,  Point, northing,  easting, plot_type
## min values  : SJER1068, center,  4107527, 254738.6,     grass
## max values  :  SJER952, center,  4112168, 258497.1,     trees
```

Our `sjer_plot_locations` object is a polygon of class `SpatialPointsDataFrame`, in the CRS **UTM zone 11N**. The CRS is critical to interpreting the object `extent` values as it specifies units.

```
<a href="{{ site.baseurl }}/images/course-materials/earth-analytics/week-4/spatial_extent.png">
<img src="{{ site.baseurl }}/images/course-materials/earth-analytics/week-4/spatial_extent.png" alt="the
<figcaption>The spatial extent of a shapefile or R spatial object represents
the geographic "edge" or location that is the furthest north, south east and
west. Thus is represents the overall geographic coverage of the spatial object.
Image Source: National Ecological Observatory Network (NEON)
</figcaption>
```

Spatial Data Attributes

Each object in a shapefile has one or more attributes associated with it. Shapefile attributes are similar to fields or columns in a spreadsheet. Each row in the spreadsheet has a set of columns associated with it that describe the row element. In the case of a shapefile, each row represents a spatial object - for example, a road, represented as a line in a line shapefile, will have one "row" of attributes associated with it. These attributes can include different types of information that describe objects stored within a shapefile. Thus, our road, may have a name, length, number of lanes, speed limit, type of road and other attributes stored with it.

```
<a href="{{ site.baseurl }}/images/course-materials/earth-analytics/week-4/attribute_table.png">
<img src="{{ site.baseurl }}/images/course-materials/earth-analytics/week-4/attribute_table.png" alt="A
associated attributes that describe or characterize the feature.
Attribute data are stored in a separate *.dbf file. "></a>
<figcaption>Each spatial feature in an R spatial object has the same set of
associated attributes that describe or characterize the feature.
Attribute data are stored in a separate *.dbf file. Attribute data can be
compared to a spreadsheet. Each row in a spreadsheet represents one feature
in the spatial object.
Image Source: National Ecological Observatory Network (NEON)
</figcaption>
```

We view the attributes of a `SpatialPointsDataFrame` using `objectName@data` (e.g., `sjer_plot_locations@data`).

```
# view attributes
sjer_plot_locations@data
##     Plot_ID  Point northing  easting plot_type
## 1  SJER1068 center  4111568 255852.4     trees
## 2   SJER112 center  4111299 257407.0     trees
## 3   SJER116 center  4110820 256838.8     grass
## 4   SJER117 center  4108752 256176.9     trees
## 5   SJER120 center  4110476 255968.4     grass
## 6   SJER128 center  4111389 257078.9     trees
## 7   SJER192 center  4111071 256683.4     grass
## 8   SJER272 center  4112168 256717.5     trees
## 9  SJER2796 center  4111534 256034.4      soil
## 10 SJER3239 center  4109857 258497.1      soil
## 11   SJER36 center  4110162 258277.8     trees
## 12  SJER361 center  4107527 256961.8     grass
## 13   SJER37 center  4107579 256148.2     trees
## 14    SJER4 center  4109767 257228.3     trees
## 15    SJER8 center  4110249 254738.6     trees
## 16  SJER824 center  4110048 256185.6      soil
## 17  SJER916 center  4109617 257460.5      soil
## 18  SJER952 center  4110759 255871.2     grass
```

We the @data called the attributes of our shapefile. What format are these attributes stored in?

```
# view structure of attributes
str(sjer_plot_locations@data)
## 'data.frame':    18 obs. of  5 variables:
##  $ Plot_ID  : chr  "SJER1068" "SJER112" "SJER116" "SJER117" ...
##  $ Point    : chr  "center" "center" "center" "center" ...
##  $ northing : num  4111568 4111299 4110820 4108752 4110476 ...
##  $ easting  : num  255852 257407 256839 256177 255968 ...
##  $ plot_type: chr  "trees" "trees" "grass" "trees" ...
```

```
# view attribute
sjer_plot_locations$plot_type
##  [1] "trees" "trees" "grass" "trees" "grass" "trees" "grass" "trees"
##  [9] "soil"  "soil"  "trees" "grass" "trees" "trees" "trees" "soil"
## [17] "soil"  "grass"
```

The attribute data for our shapefile are stored as a data.frame! We've worked with data.frames in the past few weeks. You can access individual attribute values in the same way you access them when working with time series data using the `$` sign.

Metadata & Attribute Summary

We can view a metadata & attribute summary of each shapefile by entering the name of the `R` object in the console. Note that the metadata output includes the **class**, the number of **features**, the **extent**, and the **coordinate reference system** (`crs`) of the `R` object. The last two lines of `summary` show a preview of the `R` object **attributes**.

```
# view a summary of metadata & attributes associated with the spatial object
summary(sjer_plot_locations)
## Object of class SpatialPointsDataFrame
## Coordinates:
##                 min       max
## coords.x1  254738.6  258497.1
## coords.x2 4107527.1 4112167.8
## Is projected: TRUE
## proj4string :
## [+proj=utm +zone=11 +datum=WGS84 +units=m +no_defs +ellps=WGS84
## +towgs84=0,0,0]
## Number of points: 18
## Data attributes:
##    Plot_ID             Point              northing          easting
##  Length:18          Length:18          Min.   :4107527   Min.   :254739
##  Class :character   Class :character   1st Qu.:4109790   1st Qu.:256063
##  Mode  :character   Mode  :character   Median :4110363   Median :256700
##                                        Mean   :4110258   Mean   :256674
##                                        3rd Qu.:4111242   3rd Qu.:257191
##                                        Max.   :4112168   Max.   :258497
##   plot_type
##  Length:18
##  Class :character
##  Mode  :character
##
##
##
```

Plot a Shapefile

Next, let's visualize the data in our `R` `spatialpointsdataframe` object using `plot()`.

```
# create a plot of the shapefile
# 'pch' sets the symbol
# 'col' sets point symbol color
```

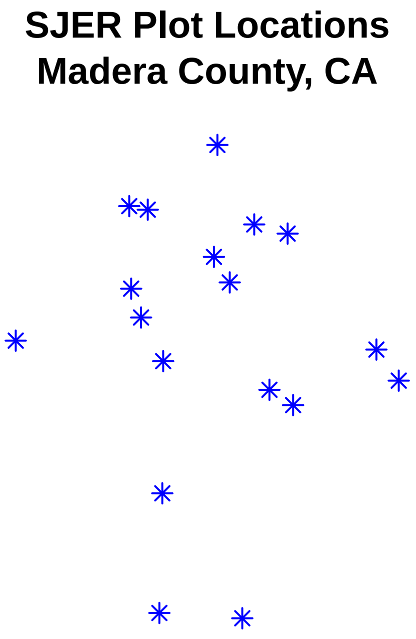

Figure 1: SJER plot locations.

```r
plot(sjer_plot_locations, col="blue",
     pch=8,
     main="SJER Plot Locations\nMadera County, CA")
```

Optional challenge: Import Line & Polygon Shapefiles

Using the steps above, import the `data/week4/california/madera-county-roads/tl_2013_06039_roads` and `data/week4/california/SJER/vector_data/SJER_crop.shp` shapefiles into `R`. Call the roads object `sjer_roads` and the crop layer `sjer_crop_extent`.

Answer the following questions:

1. What type of `R` spatial object is created when you import each layer?
2. What is the `CRS` and `extent` for each object?
3. Do the files contain, points, lines or polygons?
4. How many spatial objects are in each file?

Plot Multiple Shapefiles

The `plot()` function can be used to plot spatial objects. Use the following arguments to add a title to your plot and to layer several spatial objects on top of each other in your plot.

- `add = TRUE`: overlay a shapefile or raster on top the existing plot. This argument mimics layers in a typical GIS application like QGIS.
- `main=""`: add a title to the plot. To add a line break to your title, use `\n` where the line break should occur.

```r
# Plot multiple shapefiles
plot(sjer_crop_extent, col = "lightgreen",
     main="SJER Field Site Plot Locations - California")
```

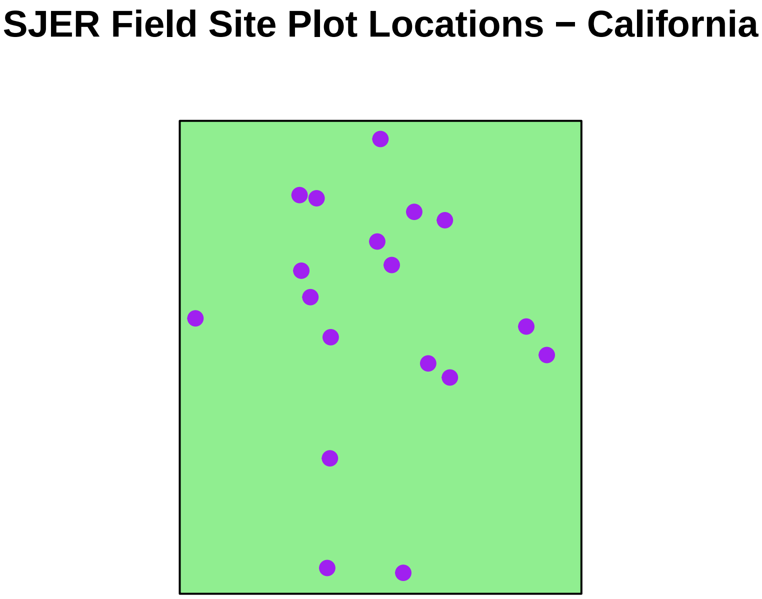

Figure 2: plot of sjer plots layered on top of the crop extent.

```r
plot(sjer_roads, add = TRUE)

# Use the pch element to adjust the symbology of the points
plot(sjer_plot_locations,
  add  = TRUE,
  pch = 19,
  col = "purple")
```

Optional challenge: Import & plot roads shapefile

- Import the `/data/week4/california/madera-county-roads/tl_2010_05143_roads.shp` shapefile in `R`. Plot the roads layer.
- Next, plot the roads layer on top of the SJER crop extent layer. What happens?
- Check the CRS of both layers. What do you notice?

Additional resources: Spatial plots in R

For more on parameter options in the base `R` `plot()` function, check out these resources:

- Parameter methods in `R`.
- Color names in `R`
- Shapefile Metadata & Attributes in R