Handling Spatial Projection & CRS

Overview

Teaching: 40 min
Exercises: 20 min
Questions

- What do I do when vector data don't line up?

Objectives

- Plot vector objects with different CRSs in the same plot.

Things You'll Need To Complete This Episode

See the lesson homepage for detailed information about the software, data, and other prerequisites you will need to work through the examples in this episode.
In an earlier episode we learned how to handle a situation where you have two different files with raster data in different projections. Now we will apply those same principles to working with vector data. We will create a base map of our study site using United States state and country boundary information accessed from the United States Census Bureau. We will learn how to map vector data that are in different CRSs and thus don't line up on a map.

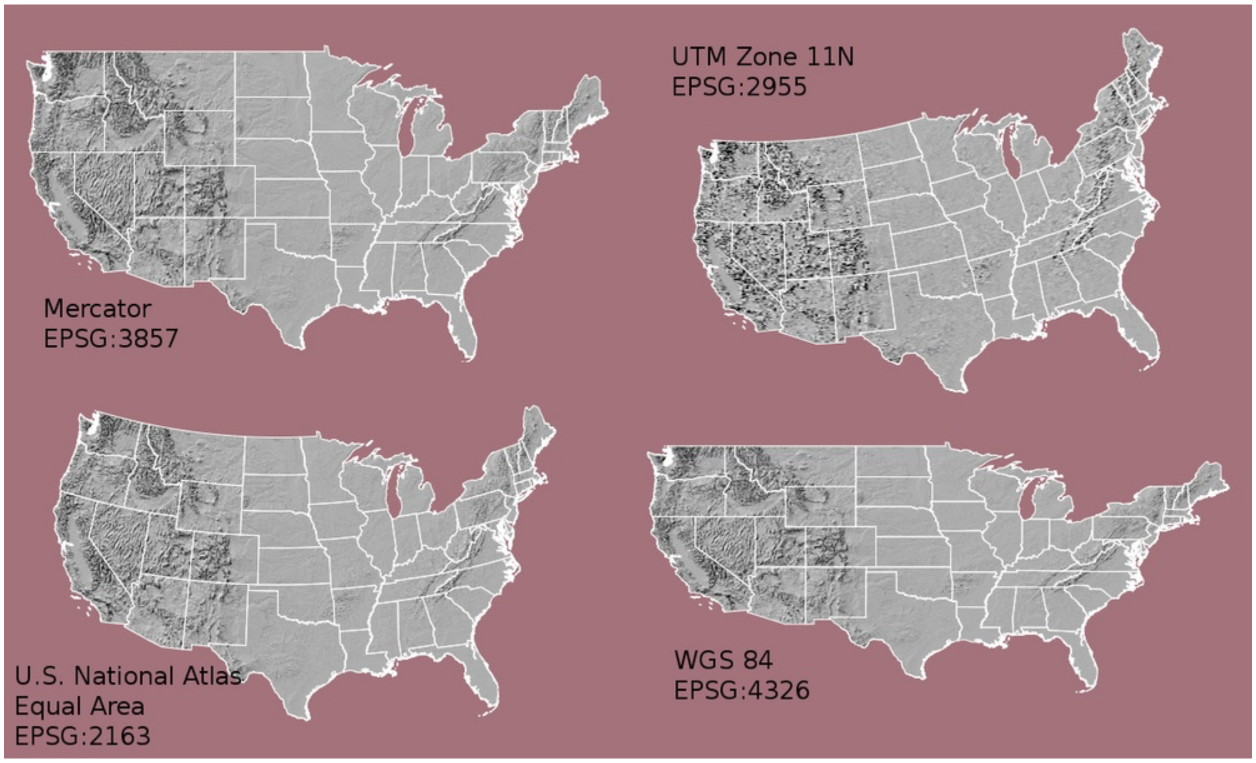
We will continue to work with the three shapefiles that we loaded in the Open and Plot Shapefiles in R episode.

Working With Spatial Data From Different Sources

We often need to gather spatial datasets from different sources and/or data that cover different spatial extents. These data are often in different Coordinate Reference Systems (CRSs).

Some reasons for data being in different CRSs include:

1. The data are stored in a particular CRS convention used by the data provider (for example, a government agency).
2. The data are stored in a particular CRS that is customized to a region. For instance, many states in the US prefer to use a State Plane projection customized for that state.



Notice the differences in shape associated with each different projection. These differences are a direct result of the calculations used to “flatten” the data onto a 2-dimensional map. Often data are stored purposefully in a particular projection that optimizes the relative shape and size of surrounding geographic boundaries (states, counties, countries, etc).

In this episode we will learn how to identify and manage spatial data in different projections. We will learn how to reproject the data so that they are in the same projection to support plotting / mapping. Note that these skills are also required for any geoprocessing / spatial analysis. Data need to be in the same CRS to ensure accurate results.

We will continue to use the `sf` and `raster` packages in this episode.

Import US Boundaries - Census Data

There are many good sources of boundary base layers that we can use to create a basemap. Some R packages even have these base layers built in to support quick and efficient mapping. In this episode, we will use boundary layers for the contiguous United States, provided by the United States Census Bureau. It is useful to have shapefiles to work with because we can add additional attributes to them if need be - for project specific mapping.

Read US Boundary File

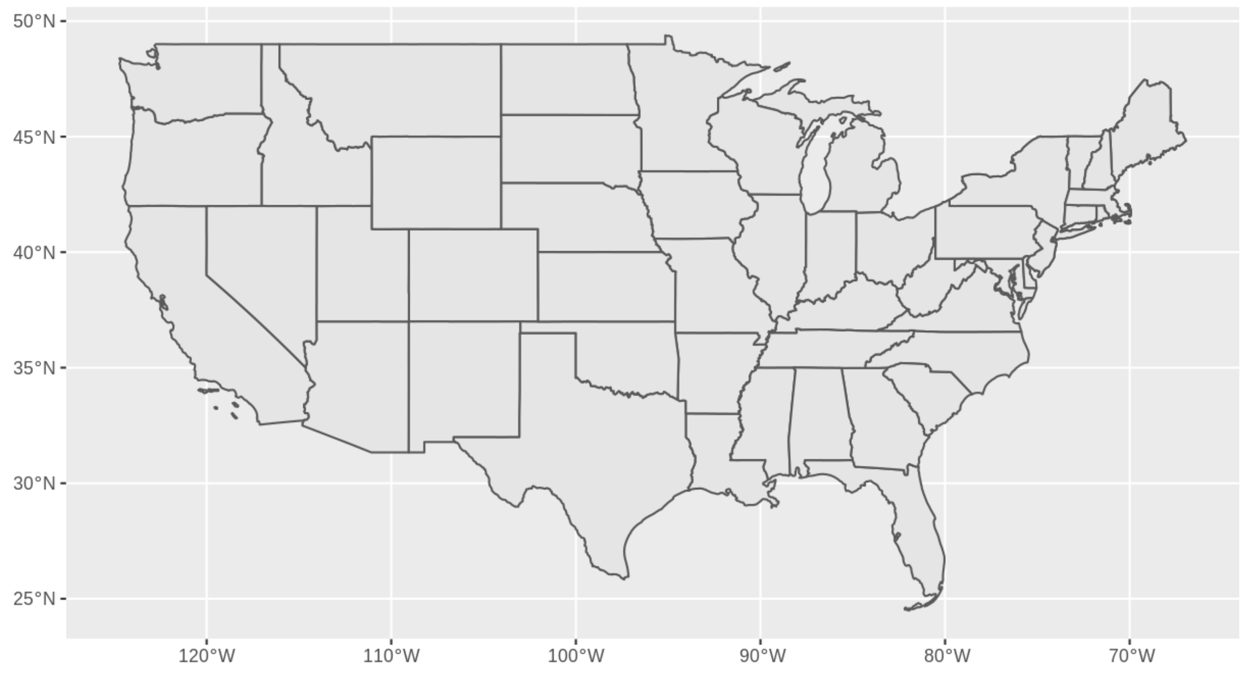
We will use the `st_read()` function to import the `/US-Boundary-Layers/US-State-Boundaries-Census-2014` layer into R. This layer contains the boundaries of all contiguous states in the U.S. Please note that these data have been modified and reprojected from the original data downloaded from the Census website to support the learning goals of this episode.

```
state_boundary_US <- st_read("data/NEON-DS-Site-Layout-Files/US-Boundary-Laye
rs/US-State-Boundaries-Census-2014.shp")
Reading layer `US-State-Boundaries-Census-2014' from data source
  `/home/runner/work/r-raster-vector-geospatial/r-raster-vector-geospatial/_e
pisodes_rmd/data/NEON-DS-Site-Layout-Files/US-Boundary-Layers/US-State-Bounda
ries-Census-2014.shp'
  using driver `ESRI Shapefile'
Simple feature collection with 58 features and 10 fields
Geometry type: MULTIPOLYGON
Dimension:     XYZ
Bounding box:  xmin: -124.7258 ymin: 24.49813 xmax: -66.9499 ymax: 49.38436
z_range:       zmin: 0 zmax: 0
Geodetic CRS:  WGS 84
```

Next, let’s plot the U.S. states data:

```
ggplot() +
  geom_sf(data = state_boundary_US) +
  ggtitle("Map of Contiguous US State Boundaries") +
  coord_sf()
```

Map of Contiguous US State Boundaries



U.S. Boundary Layer

We can add a boundary layer of the United States to our map - to make it look nicer. We will import`NEON-DS-Site-Layout-Files/US-Boundary-Layers/US-Boundary-Dissolved-States`.

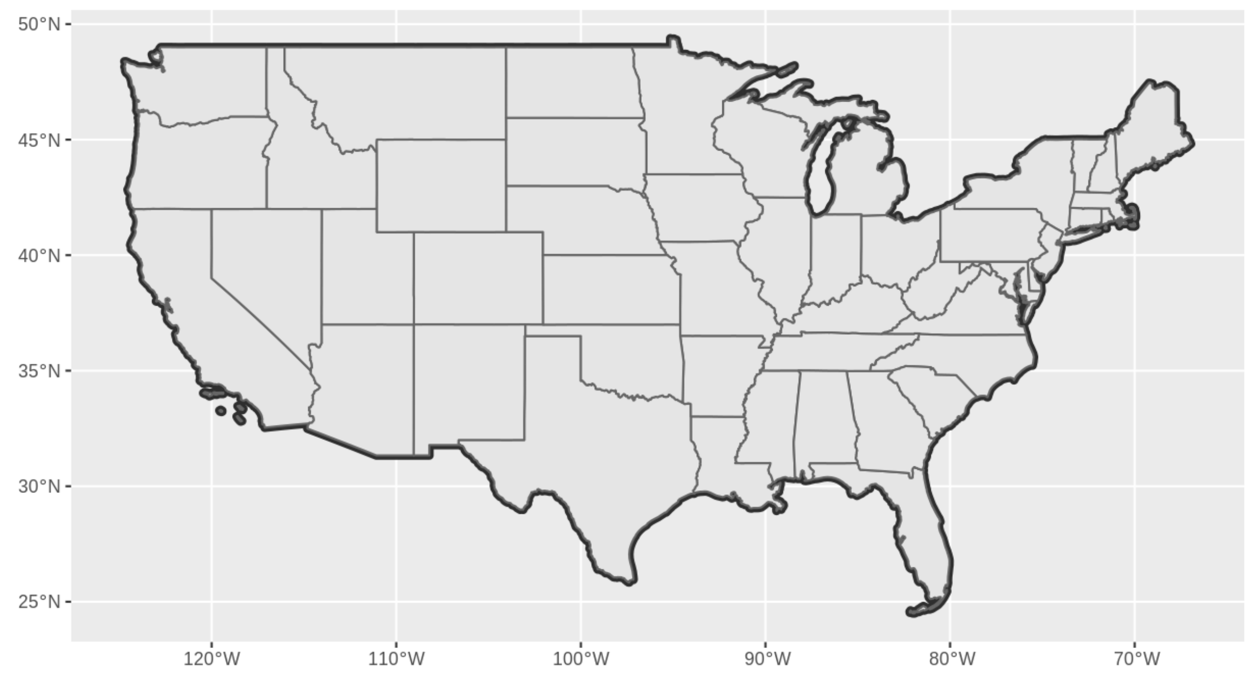
```
country_boundary_US <- st_read("data/NEON-DS-Site-Layout-Files/US-Boundary-La
yers/US-Boundary-Dissolved-States.shp")
Reading layer `US-Boundary-Dissolved-States' from data source
```

```
  `/home/runner/work/r-raster-vector-geospatial/r-raster-vector-geospatial/_e
pisodes_rmd/data/NEON-DS-Site-Layout-Files/US-Boundary-Layers/US-Boundary-Dis
solved-States.shp'
  using driver `ESRI Shapefile'
Simple feature collection with 1 feature and 9 fields
Geometry type: MULTIPOLYGON
Dimension:     XYZ
Bounding box:  xmin: -124.7258 ymin: 24.49813 xmax: -66.9499 ymax: 49.38436
z_range:       zmin: 0 zmax: 0
Geodetic CRS:  WGS 84
```

If we specify a thicker line width using `size = 2` for the border layer, it will make our map pop! We will also manually set the colors of the state boundaries and country boundaries.

```
ggplot() +
  geom_sf(data = country_boundary_US, color = "gray18", size = 2) +
  geom_sf(data = state_boundary_US, color = "gray40") +
  ggtitle("Map of Contiguous US State Boundaries") +
  coord_sf()
```

Map of Contiguous US State Boundaries



Next, let's add the location of a flux tower where our study area is. As we are adding these layers, take note of the CRS of each object. First let's look at the CRS of our tower location object:

```
st_crs(point_HARV)
Coordinate Reference System:
  User input: WGS 84 / UTM zone 18N
  wkt:
PROJCRS["WGS 84 / UTM zone 18N",
    BASEGEOGCRS["WGS 84",
        DATUM["World Geodetic System 1984",
            ELLIPSOID["WGS 84",6378137,298.257223563,
                LENGTHUNIT["metre",1]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4326]],
    CONVERSION["UTM zone 18N",
        METHOD["Transverse Mercator",
            ID["EPSG",9807]],
        PARAMETER["Latitude of natural origin",0,
            ANGLEUNIT["Degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",-75,
            ANGLEUNIT["Degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["Scale factor at natural origin",0.9996,
            SCALEUNIT["unity",1],
            ID["EPSG",8805]],
        PARAMETER["False easting",500000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",0,
            LENGTHUNIT["metre",1],
            ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["(E)",east,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["(N)",north,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    ID["EPSG",32618]]
```

Our project string for `DSM_HARV` specifies the UTM projection as follows:

`+proj=utm +zone=18 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0`

- **proj=utm:** the projection is UTM, UTM has several zones.
- **zone=18:** the zone is 18
- **datum=WGS84:** the datum WGS84 (the datum refers to the 0,0 reference for the coordinate system used in the projection)
- **units=m:** the units for the coordinates are in METERS.
- **ellps=WGS84:** the ellipsoid (how the earth's roundness is calculated) for the data is WGS84

Note that the `zone` is unique to the UTM projection. Not all CRSs will have a zone.

Let's check the CRS of our state and country boundary objects:

```
st_crs(state_boundary_US)
Coordinate Reference System:
  User input: WGS 84
  wkt:
GEOGCRS["WGS 84",
    DATUM["World Geodetic System 1984",
        ELLIPSOID["WGS 84",6378137,298.257223563,
            LENGTHUNIT["metre",1]]],
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
    CS[ellipsoidal,2],
        AXIS["latitude",north,
            ORDER[1],
            ANGLEUNIT["degree",0.0174532925199433]],
        AXIS["longitude",east,
            ORDER[2],
            ANGLEUNIT["degree",0.0174532925199433]],
    ID["EPSG",4326]]
st_crs(country_boundary_US)
Coordinate Reference System:
  User input: WGS 84
  wkt:
GEOGCRS["WGS 84",
    DATUM["World Geodetic System 1984",
        ELLIPSOID["WGS 84",6378137,298.257223563,
            LENGTHUNIT["metre",1]]],
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
    CS[ellipsoidal,2],
        AXIS["latitude",north,
            ORDER[1],
            ANGLEUNIT["degree",0.0174532925199433]],
        AXIS["longitude",east,
            ORDER[2],
            ANGLEUNIT["degree",0.0174532925199433]],
    ID["EPSG",4326]]
```
Our project string for `state_boundary_US` and `country_boundary_US` specifies the lat/long projection as follows:

`+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0`

- **proj=longlat:** the data are in a geographic (latitude and longitude) coordinate system
- **datum=WGS84:** the datum WGS84 (the datum refers to the 0,0 reference for the coordinate system used in the projection)
- **ellps=WGS84:** the ellipsoid (how the earth's roundness is calculated) is WGS84

Note that there are no specified units above. This is because this geographic coordinate reference system is in latitude and longitude which is most often recorded in decimal degrees.

Data Tip

the last portion of each `proj4` string is `+towgs84=0,0,0` . This is a conversion factor that is used if a datum conversion is required. We will not deal with datums in this episode series.

CRS Units - View Object Extent

Next, let's view the extent or spatial coverage for the `point_HARV` spatial object compared to the `state_boundary_US` object.

First we'll look at the extent for our study site:

```
st_bbox(point_HARV)
     xmin      ymin      xmax      ymax
 732183.2 4713265.0  732183.2 4713265.0
```

And then the extent for the state boundary data.

```
st_bbox(state_boundary_US)
      xmin       ymin       xmax       ymax 
-124.72584   24.49813  -66.94989   49.38436 
```

Note the difference in the units for each object. The extent for `state_boundary_US` is in latitude and longitude which yields smaller numbers representing decimal degree units. Our tower location point is in UTM, is represented in meters.

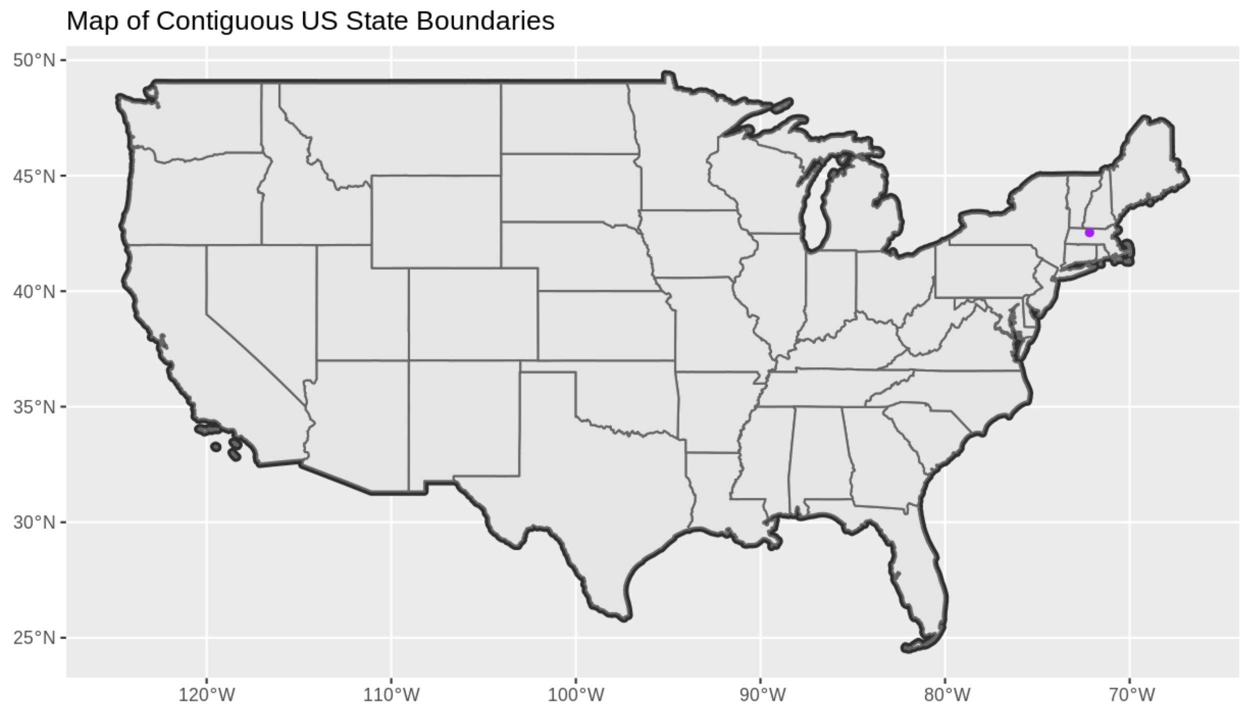
Proj4 & CRS Resources

- Official PROJ library documentation
- More information on the proj4 format.
- A fairly comprehensive list of CRSs by format.
- To view a list of datum conversion factors type: `projInfo(type = "datum")` into the R console.

Reproject Vector Data or No?

We saw in an earlier episode that when working with raster data in different CRSs, we needed to convert all objects to the same CRS. We can do the same thing with our vector data - however, we don't need to! When using the `ggplot2` package, `ggplot` automatically converts all objects to the same CRS before plotting. This means we can plot our three data sets together without doing any conversion:

```
ggplot() +
  geom_sf(data = country_boundary_US, size = 2, color = "gray18") +
  geom_sf(data = state_boundary_US, color = "gray40") +
  geom_sf(data = point_HARV, shape = 19, color = "purple") +
  ggtitle("Map of Contiguous US State Boundaries") +
  coord_sf()
```



Challenge - Plot Multiple Layers of Spatial Data

Create a map of the North Eastern United States as follows:

1. Import and plot `Boundary-US-State-NEast.shp`. Adjust line width as necessary.
2. Layer the Fisher Tower (in the NEON Harvard Forest site) point location `point_HARV` onto the plot.
3. Add a title.
4. Add a legend that shows both the state boundary (as a line) and the Tower location point.

Answers

Key Points

- `ggplot2` automatically converts all objects in a plot to the same CRS.
- Still be aware of the CRS and extent for each object.