GIS in R: how to reproject vector data in different coordinate reference systems (crs) in R

Learning Objectives

After completing this tutorial, you will be able to:

- Describe atleast 2 reasons that a data provider may chose to store a dataset in a particular CRS.
- Reproject a vector dataset to another CRS in R.
- Identify the CRS of a spatial dataset in R.

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 3 Data (~250 MB){:data-proofer-ignore=" .btn }

Working With Spatial Data From Different Sources

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

Some reasons for data being in different CRSs include:

1. The data are stored in a particular CRS convention used by the data provider which might be a federal agency, or a state planning office.
2. The data are stored in a particular CRS that is customized to a region. For instance, many states prefer to use a **State Plane** projection customized for that state.

```
<a href="https://source.opennews.org/media/cache/b9/4f/b94f663c79024f0048ae7b4f88060cb5.jpg">
<img src="https://source.opennews.org/media/cache/b9/4f/b94f663c79024f0048ae7b4f88060cb5.jpg">
</a>

<figcaption>Maps of the United States using data in different projections.
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).
Source: opennews.org</figcaption>
```

Check out this short video highlighting how map projections can make continents seems proportionally larger or smaller than they actually are!

In this tutorial 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 use the `rgdal` and `raster` libraries in this tutorial.

```
# load spatial data packages
library(rgdal)
library(raster)

# set working directory to data folder
# setwd("pathToDirHere")
```

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 tutorial, we will use boundary layers for the 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 `readOGR()` function to import the `/usa-boundary-layers/US-State-Boundaries-Census-2014` layer into R. This layer contains the boundaries of all continental 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 tutorial.

```
# Import the shapefile data into R
state_boundary_us <- readOGR("data/week4/usa-boundary-layers",
          "US-State-Boundaries-Census-2014")
## OGR data source with driver: ESRI Shapefile
## Source: "data/week4/usa-boundary-layers", layer: "US-State-Boundaries-Census-2014"
## with 58 features
## It has 10 fields
## Integer64 fields read as strings:  ALAND AWATER
## Warning in readOGR("data/week4/usa-boundary-layers", "US-State-Boundaries-
## Census-2014"): Z-dimension discarded

# view data structure
class(state_boundary_us)
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

Note: the Z-dimension warning is normal. The `readOGR()` function doesn't import z (vertical dimension or height) data by default. This is because not all shapefiles contain z dimension data. More on readOGR

Next, let's plot the U.S. states data.

```
# view column names
plot(state_boundary_us,
     main="Map of Continental US State Boundaries\n US Census Bureau Data")
```

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 `data/week4/usa-boundary-layers/US-Boundary-Dissolved-States`. If we specify a thicker line width

Map of Continental US State Boundaries
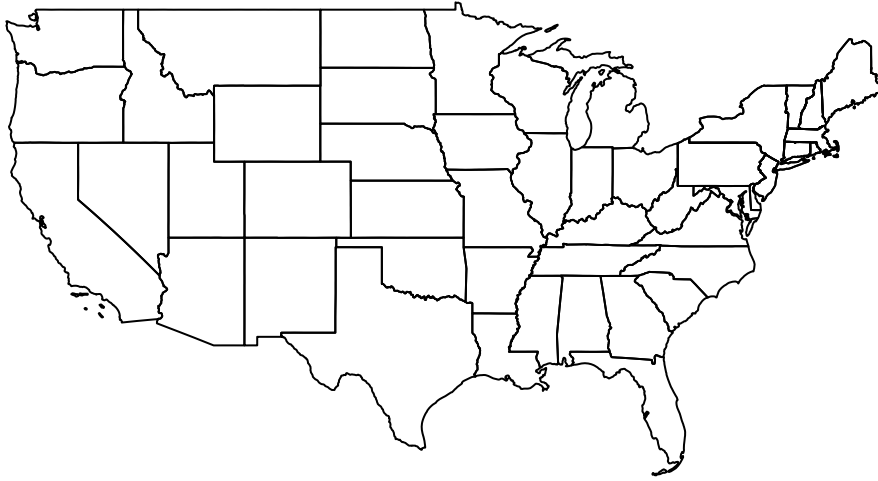## US Census Bureau Data

Figure 1: Plot of the continental united states.

using `lwd=4` for the border layer, it will make our map pop!

```r
# Read the .csv file
country_boundary_us <- readOGR("data/week4/usa-boundary-layers",
          "US-Boundary-Dissolved-States")
## OGR data source with driver: ESRI Shapefile
## Source: "data/week4/usa-boundary-layers", layer: "US-Boundary-Dissolved-States"
## with 1 features
## It has 9 fields
## Integer64 fields read as strings:  ALAND AWATER
## Warning in readOGR("data/week4/usa-boundary-layers", "US-Boundary-
## Dissolved-States"): Z-dimension discarded

# look at the data structure
class(country_boundary_us)
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"

# view column names
plot(state_boundary_us,
    main="Map of Continental US State Boundaries\n US Census Bureau Data",
    border="gray40")

# view column names
plot(country_boundary_us,
    lwd=4,
    border="gray18",
    add = TRUE)
```

Next, let's add the location of our study area sites. As we are adding these layers, take note of the class of each object. We will use AOI to represent "Area of Interest" in our data.

Map of Continental US State Boundaries
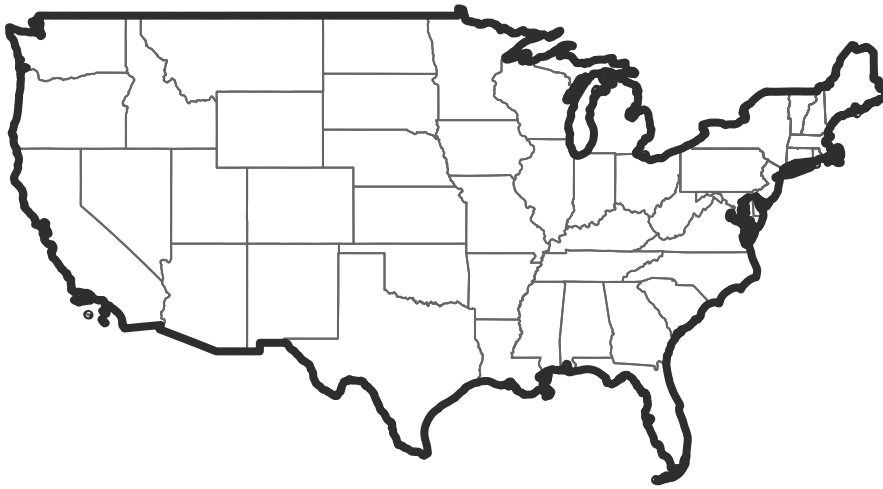**US Census Bureau Data**

Figure 2: Plot of the US overlayed with states and a boundary.

```
# Import a point shapefile
sjer_aoi <- readOGR("data/week4/california/SJER/vector_data",
                    "SJER_crop")
## OGR data source with driver: ESRI Shapefile
## Source: "data/week4/california/SJER/vector_data", layer: "SJER_crop"
## with 1 features
## It has 1 fields
class(sjer_aoi)
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"

# plot point - looks ok?
plot(sjer_aoi,
     pch = 19,
     col = "purple",
     main="San Joachin Experimental Range AOI")
```

Our SJER AOI layer plots nicely. Let's next add it as a layer on top of the U.S. states and boundary layers in our basemap plot.

```
# plot state boundaries
plot(state_boundary_us,
     main="Map of Continental US State Boundaries \n with SJER AOI",
     border="gray40")

# add US border outline
plot(country_boundary_us,
     lwd=4,
     border="gray18",
     add = TRUE)
```

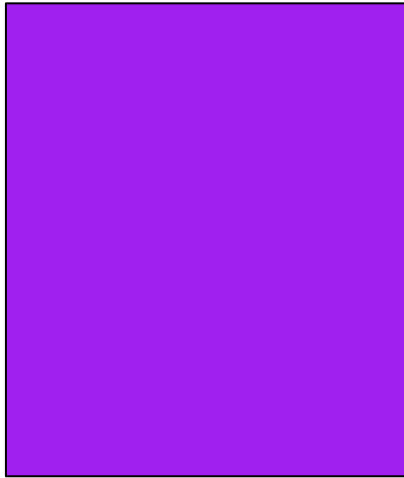## San Joachin Experimental Range AOI

Figure 3: plot aoi

```
# add AOI boundary
plot(sjer_aoi,
     pch = 19,
     col = "purple",
     add = TRUE)
```

What do you notice about the resultant plot? Do you see the AOI boundary in the California area? Is something wrong?

Let's check out the CRS (`crs()`) of both datasets to see if we can identify any issues that might cause the point location to not plot properly on top of our U.S. boundary layers.

```
# view CRS of our site data
crs(sjer_aoi)
## CRS arguments:
##  +proj=utm +zone=11 +datum=WGS84 +units=m +no_defs +ellps=WGS84
## +towgs84=0,0,0

# view crs of census data
crs(state_boundary_us)
## CRS arguments:
##  +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
crs(country_boundary_us)
## CRS arguments:
##  +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
```

It looks like our data are in different CRS. We can tell this by looking at the CRS strings in `proj4` format.

Understanding CRS in Proj4 Format

The CRS for our data are given to us by R in `proj4` format. Let's break down the pieces of `proj4` string. The string contains all of the individual CRS elements that R or another GIS might need. Each element is

5

Map of Continental US State Boundaries
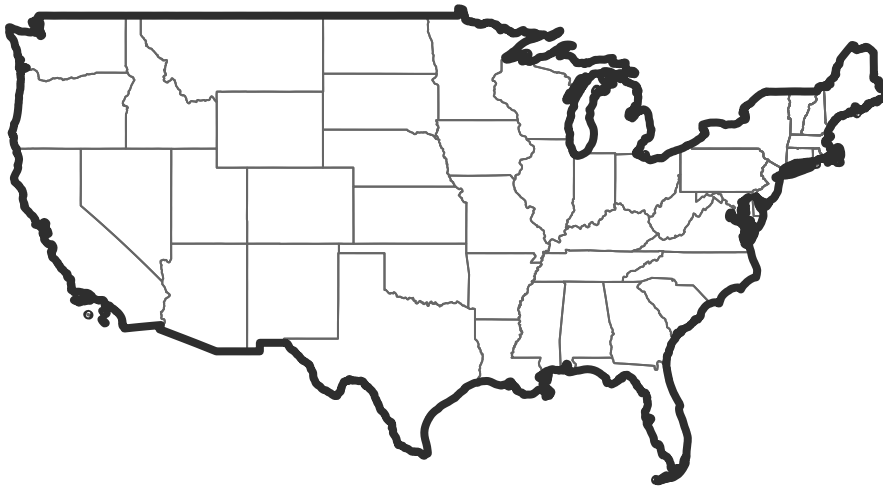# with SJER AOI

Figure 4: plot states

specified with a + sign, similar to how a `.csv` file is delimited or broken up by a `,`. After each + we see the CRS element being defined. For example projection (`proj=`) and datum (`datum=`).

UTM Proj4 String

Our project string for `sjer_aoi` 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 CRS will have a zone.

Geographic (lat / long) Proj4 String

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 tutorial series. {: .notice}

CRS Units - View Object Extent

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

```
# extent & crs for AOI
extent(sjer_aoi)
## class       : Extent
## xmin        : 254570.6
## xmax        : 258867.4
## ymin        : 4107303
## ymax        : 4112362
crs(sjer_aoi)
## CRS arguments:
##  +proj=utm +zone=11 +datum=WGS84 +units=m +no_defs +ellps=WGS84
## +towgs84=0,0,0

# extent & crs for object in geographic
extent(state_boundary_us)
## class       : Extent
## xmin        : -124.7258
## xmax        : -66.94989
## ymin        : 24.49813
## ymax        : 49.38436
crs(state_boundary_us)
## CRS arguments:
##  +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
```

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 AOI boundary point is in UTM, is represented in meters.

Proj4 & CRS Resources

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

Reproject Vector Data

Now we know our data are in different CRS. To address this, we have to modify or **reproject** the data so they are all in the **same** CRS. We can use `spTransform()` function to reproject our data. When we reproject the data, we specify the CRS that we wish to transform our data to. This CRS contains the datum, units and other information that `R` needs to **reproject** our data.

The `spTransform()` function requires two inputs:

1. the name of the object that you wish to transform
2. the CRS that you wish to transform that object too. In this case we can use the `crs()` of the `state_boundary_us` object as follows: `crs(state_boundary_us)`

Data Tip: `spTransform()` will only work if your original spatial object has a CRS assigned to it AND if that CRS is the correct CRS! {: .notice}

Next, let's reproject our point layer into the geographic - latitude and longitude `WGS84` coordinate reference system (CRS).

```
# reproject data
sjer_aoi_WGS84 <- spTransform(sjer_aoi,
                              crs(state_boundary_us))

# what is the CRS of the new object
crs(sjer_aoi_WGS84)
## CRS arguments:
##  +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
# does the extent look like decimal degrees?
extent(sjer_aoi_WGS84)
## class      : Extent
## xmin       : -119.7626
## xmax       : -119.7127
## ymin       : 37.0799
## ymax       : 37.12657
```

Once our data are reprojected, we can try to plot again.

```
# plot state boundaries
plot(state_boundary_us,
     main="Map of Continental US State Boundaries\n With SJER AOI",
     border="gray40")

# add US border outline
plot(country_boundary_us,
     lwd=4,
     border="gray18",
     add = TRUE)

# add AOI
plot(sjer_aoi_WGS84,
     pch = 19,
     col = "purple",
     add = TRUE)
```

But now, the aoi is a polygon and it's too small to see on the map. Let's convert the polygon to a polygon CENTROID and plot yet again.

```
# get coordinate center of the polygon
aoi_centroid <- coordinates(sjer_aoi_WGS84)

# plot state boundaries
plot(state_boundary_us,
     main="Map of Continental US State Boundaries\n With SJER AOI",
     border="gray40")

# add US border outline
```

Map of Continental US State Boundaries
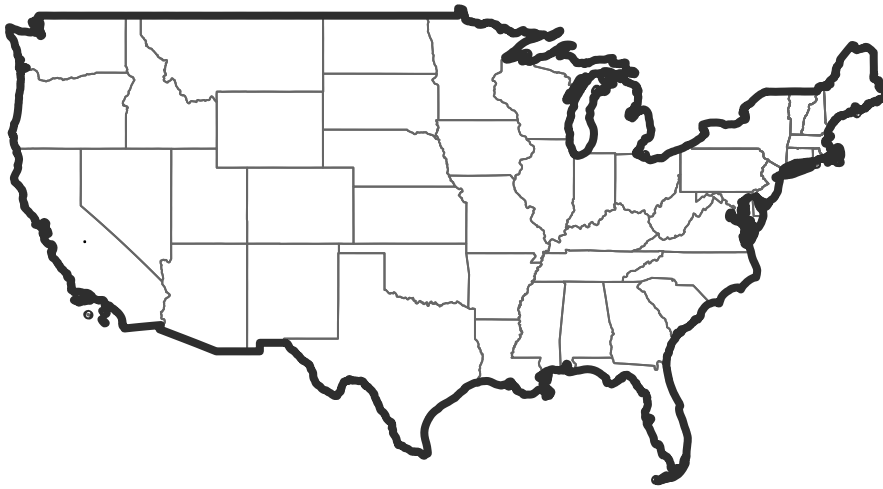# With SJER AOI

Figure 5: US Map with SJER AOI Location

```
plot(country_boundary_us,
     lwd=4,
     border="gray18",
     add = TRUE)

# add point location of the centroid to the plot
points(aoi_centroid, pch=8, col="magenta", cex=1.5)
```

Reprojecting our data ensured that things line up on our map! It will also allow us to perform any required geoprocessing (spatial calculations / transformations) on our data.

Homework plot 1: Crop, Reproject, Plot data

Create a map of our SJER study area as follows:

1. Import the `madera-county-roads/tl_2013_06039_roads.shp` layer located in your week4 data download. Adjust line width as necessary.
2. Create a map that shows the roads layer, study site locations and the sjer_aoi boundary.
3. Add a **title** to your plot.
4. Add a **legend** to your plot that shows both the roads and the plot locations.
5. Plot the roads by road type and add each type to the legend. HINT: use the metadata included in your data download to figure out what each type of road represents ("C", "S", etc.). Use the homework lesson on custom legends to help build the legend.

IMPORTANT: be sure that all of the data are within the same EXTENT and crs of the sjer_aoi layer. This means that you may have to CROP and reproject your data prior to plotting it!

```
## RStudioGD
##         2
```
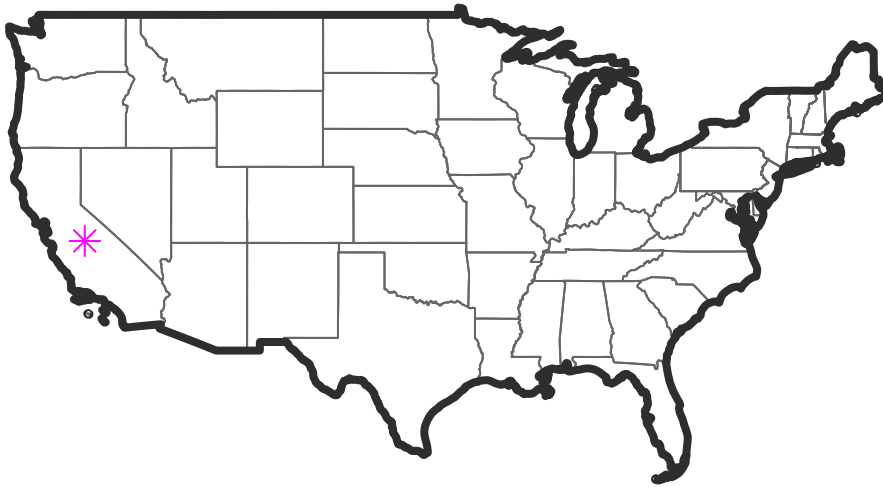
Map of Continental US State Boundaries
With SJER AOI

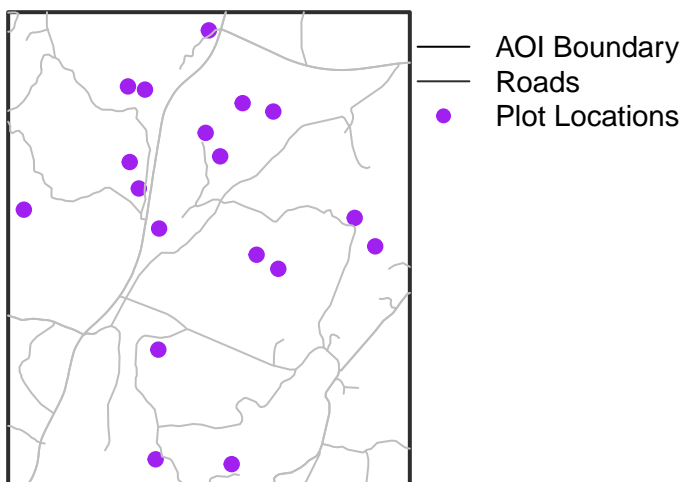Figure 6: figure out AOI polygon centroid.

SJER Area of Interest (AOI)

— AOI Boundary
— Roads
● Plot Locations

Figure 7: challenge plot