Lab 1: Accessing, exporting, and mapping spatial data in R and QGIS

In this lab we will be working between the Spatial Packages in R and QGIS to Access, Manage, and Export spatial data for analysis, projection, visualization, and presentation. As mentioned in class, there are times when researchers/practitioners will want to work in R and others when they will want to work in QGIS. Both have their own strengths and weaknesses so being able to move files between the two is very practical.

To begin, download the sat_verbal.csv data file and the lab1_Script.R script file. The sat_verbal.csv file contains data pertaining to SAT verbal test scores at the state level and the lab1_Script.R script file contains the R code used in this lab.

Open R….. and the working lab1_Script.R file.

First, we want to set our working directory

```
setwd()
oldpar<-par()
```

Projections

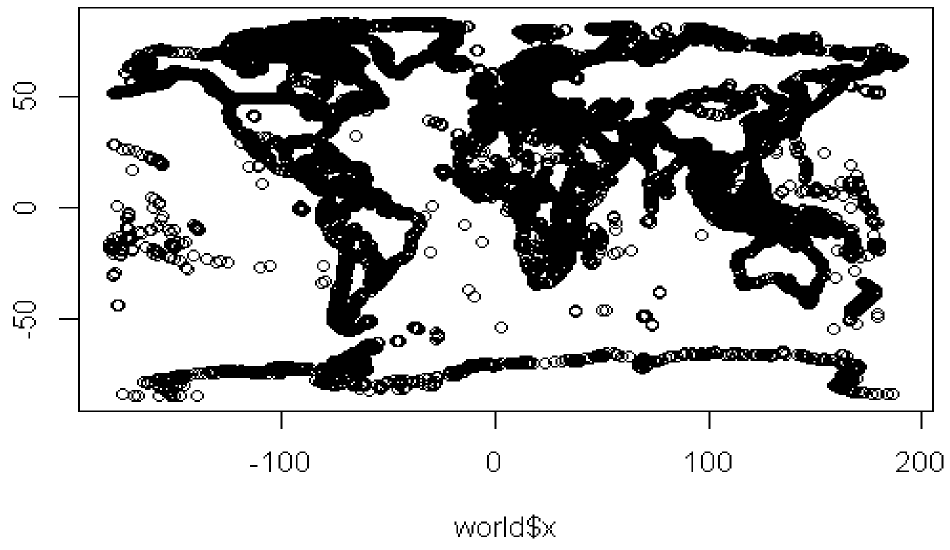
All spatial data need to be projected to be reliably examined spatially. There are a number of projections, with the default projection being long/lat. Using the following code, we will project data that are available as part of the "maps" package.

First, be sure the "maps" package is loaded and installed.

```
library(maps)
```

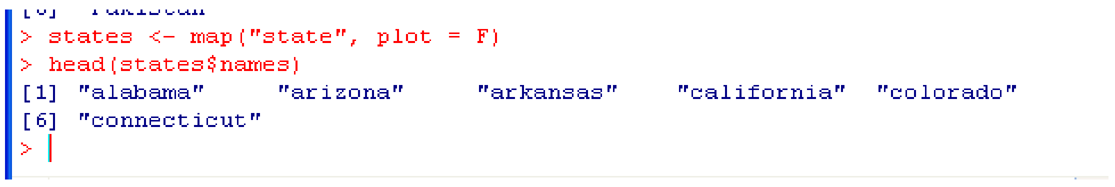
Once the package is loaded, assign the world boundary file to the object world and examine the units of analysis using the following code to display the names of the nations in the file.

```
world <- map("world")
str(world)
head(world$names)
plot(world)
```

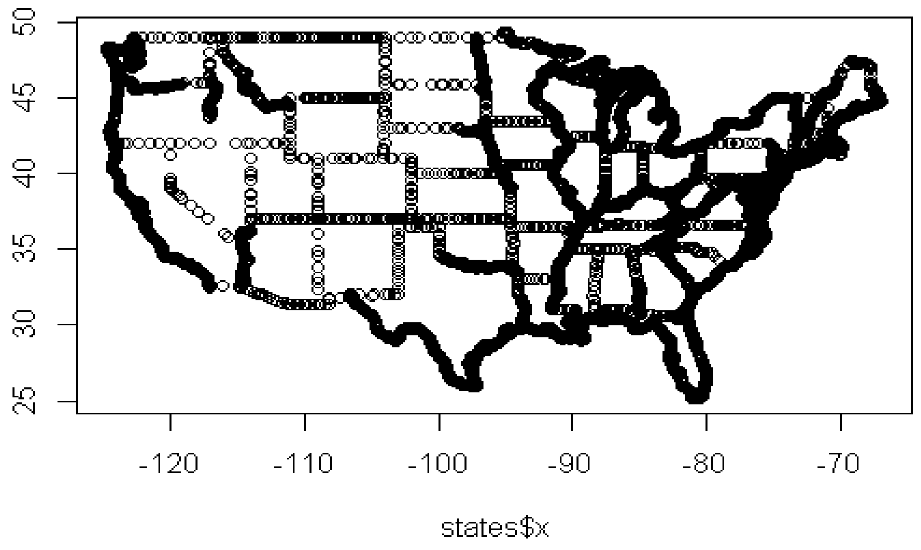


Next, assign the state boundary file to the object states and display the names of states using the following code.

states <- map("state")
str(states)
head(states$names)
plot(states)



```
> states <- map("state", plot = F)
> head(states$names)
[1] "alabama"     "arizona"     "arkansas"    "california"  "colorado"
[6] "connecticut"
>
```



Next, convert the object to a SpatialLines object using the “maptools” objects.

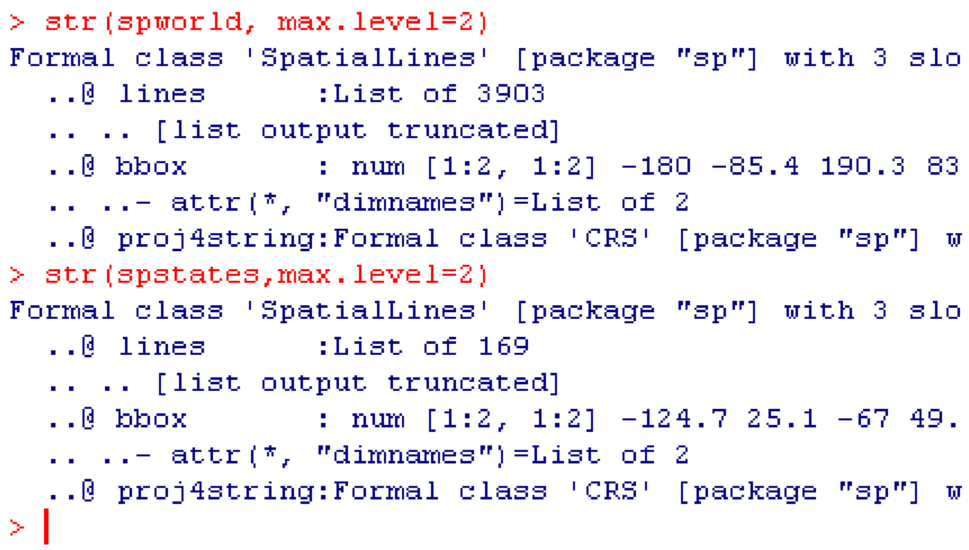
First, load and install “maptools”

```
library(maptools)
```

Next, use the map2SpatialLines command to create a SpatialLines object with a generic long/lat projection.

```
spworld <- map2SpatialLines(world, proj4string = CRS("+proj=longlat"))
spstates <- map2SpatialLines(states, proj4string = CRS("+proj=longlat"))
```

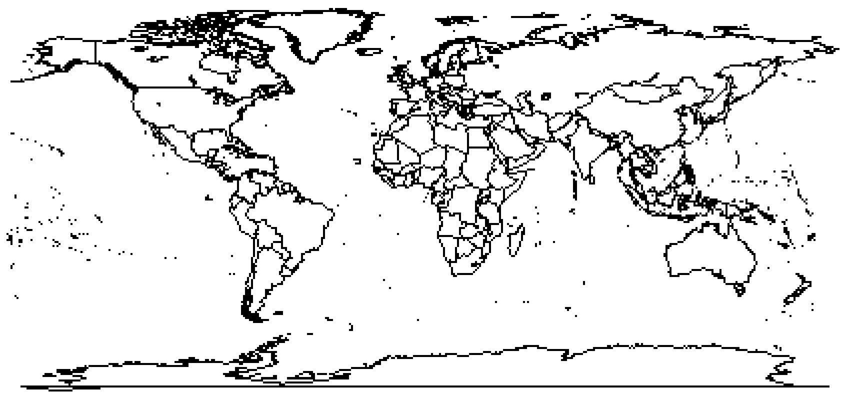
```
str(spworld, max.level=2)
str(sptates,max.level=2)
```



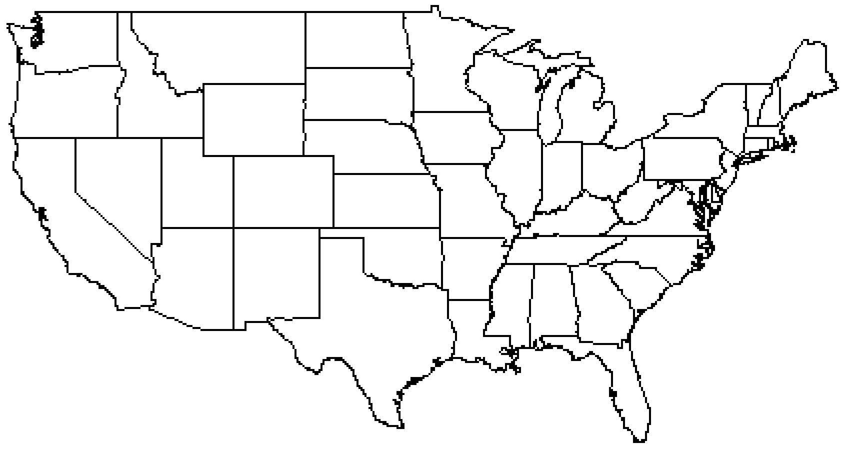
```
> str(spworld, max.level=2)
Formal class 'SpatialLines' [package "sp"] with 3 slo
  ..@ lines      :List of 3903
  .. .. [list output truncated]
  ..@ bbox       : num [1:2, 1:2] -180 -85.4 190.3 83
  .. ..- attr(*, "dimnames")=List of 2
  ..@ proj4string:Formal class 'CRS' [package "sp"] w
> str(spstates,max.level=2)
Formal class 'SpatialLines' [package "sp"] with 3 slo
  ..@ lines      :List of 169
  .. .. [list output truncated]
  ..@ bbox       : num [1:2, 1:2] -124.7 25.1 -67 49.
  .. ..- attr(*, "dimnames")=List of 2
  ..@ proj4string:Formal class 'CRS' [package "sp"] w
> 
```

To visualize the spatial line objects use the following command.

```
plot(spworld)
```



```
plot(spstates)
```



We can also project the data using the “rgdal” package to examine differences between the visual display of the generic long/lat projection and a more aesthetically pleasing Lambert Azmuthal Equal Area (laea) projection.

Load “rgdal”

```
library(rgdal)
```

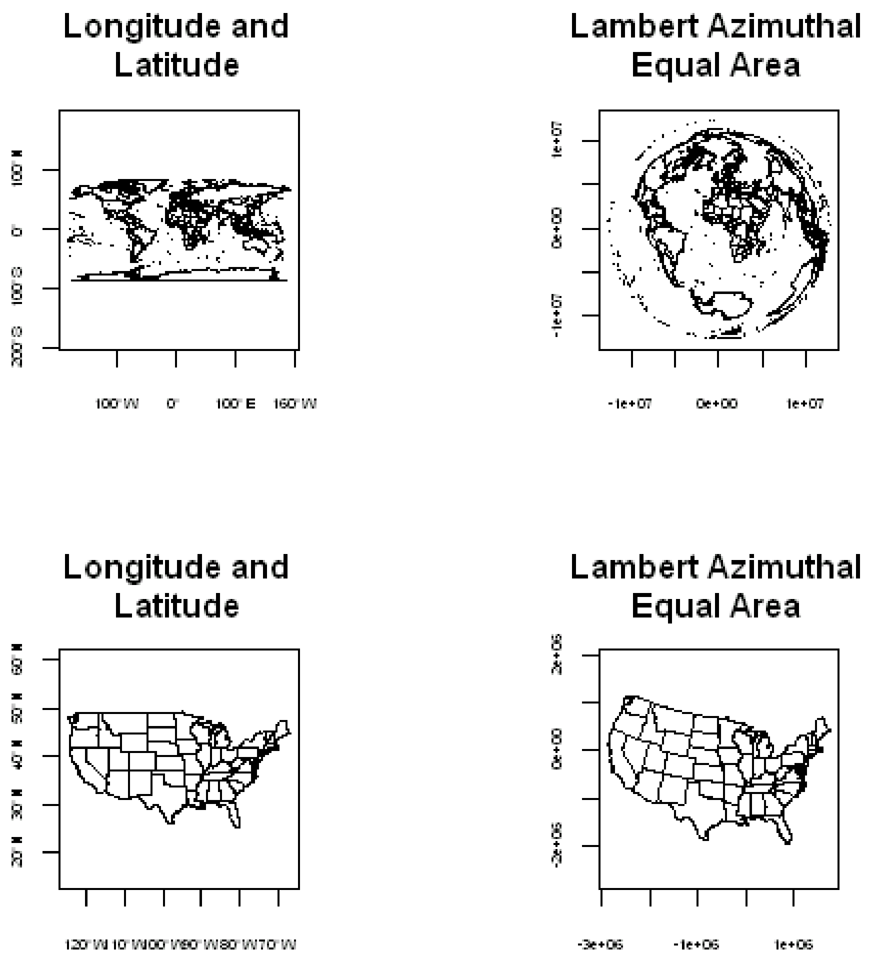
Use the spTransform command to set the laea objects for world and state using the following code….(see http://spatialreference.org/)

```
world.laea <- spTransform(spworld, CRS("+proj=laea +lat_0=0 +lon_0=0"))
states.laea <- spTransform(spstates, CRS("+proj=laea +lat_0=43.0758 +lon_0=-
89.3976"))
```

Now that we have the world and the states objects projected by both long/lat and laea, use the following code to display the objects in a 2 by 2 frame.

```
par(mfrow = c(2, 2), pty = "s", cex.axis = 0.5)
plot(spworld, axes = T)
title(main = "Longitude and\nLatitude")
plot(world.laea, axes = T)
title(main = "Lambert Azimuthal\nEqual Area")
plot(spstates, axes = T)
title(main = "Longitude and\nLatitude")
plot(states.laea, axes = T)
title(main = "Lambert Azimuthal\nEqual Area")
```

The following output shows the difference between a generic long/lat projection and a conformal projection.



Spatial Referencing

Next, we work through linking data to spatial objects in R.

First, restore the default graphic parameters…

par(oldpar)

Next, create map.states object by calling the states boundaries and using the following code.

map.states <- map("state", plot = T, fill = T, res=0)

List the state names and set state names as ID field in the object using the following code.

list.names.states <- strsplit(map.states$names,":")
head(list.names.states, n=63)
View(list.names.states)

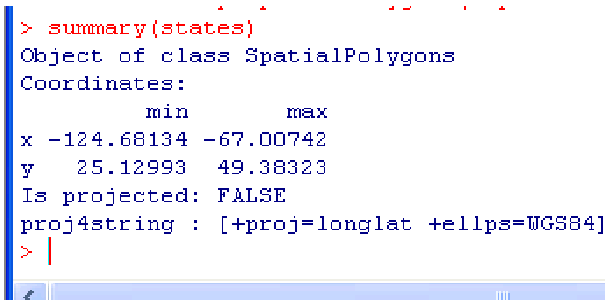
map.IDs <- sapply(list.names.states, function(x) x[1])
head(map.IDs, n=63)

Use the maps2SpatailPolygons command to convert the map.states object to Spatial Polygons with IDs set to map.IDs using the following code.

polystates <- map2SpatialPolygons(map.states, IDs = map.IDs,proj4string = CRS("+proj=longlat"))

Examine the Spatial Polygons object to see bounding box and projection information.

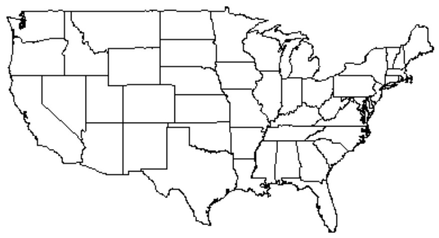
summary(polystates)



```
> summary(states)
Object of class SpatialPolygons
Coordinates:
         min       max
x -124.68134 -67.00742
y   25.12993  49.38323
Is projected: FALSE
proj4string : [+proj=longlat +ellps=WGS84]
> 
```

Plot the long/lat projected states object.

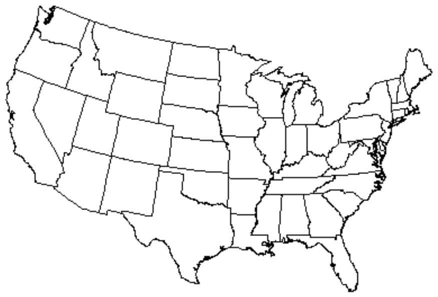
```
plot(polystates)
```



Next, project the states object using the following laea projection command and plot the projected states file.

```
states.laea <- spTransform(polystates, CRS("+proj=laea +lat_0=43.0758 +lon_0=-
89.3976"))

plot(states.laea)
```



Next, set, and list, the spatial polygon IDs for states.laea using the following code:

```
sp.IDs <- sapply(slot(states.laea, "polygons"), function(x) slot(x,"ID"))
head(sp.IDs, n=50)
```

Now that the IDs are set as the state names, lets read in the state level SAT scores from the .csv file provided to you through CourseWorks. Set the data as the object "sat" and list the data using the following code.

```
sat_verbal<- read.csv("sat_verbal.csv", stringsAsFactors = F,row.names = 1)
head(sat_verbal, n=50)
```

Create the SpatialPolygonsDataFrame by linking states and sat to create states.sat… then summarize the SpatialPolygonsDataFrame using the following code:

```
states.verbal <- SpatialPolygonsDataFrame(polystates,sat_verbal)
summary(states.verbal)
```

We can project the states.sat data frame by using the same spTransform code from above and then we can plot the new "states.sat.laea" SpatialPolygonsDataFrame.

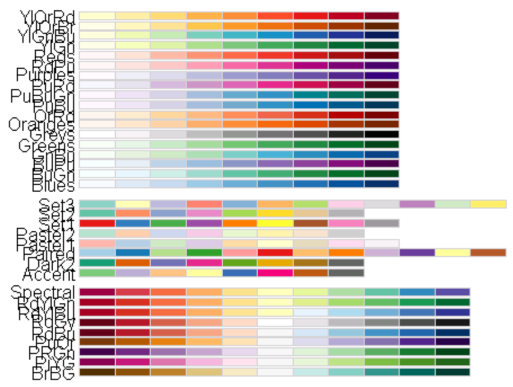
```
states.verbal.laea <- spTransform(states.verbal, CRS("+proj=laea +lat_0=43.0758
+lon_0=-89.3976"))
```

```
plot(states.verbal.laea)
summary(states.verbal.laea)
```

Given that our data are now spatial, we are interested in visualizing the geographic distribution of the data.

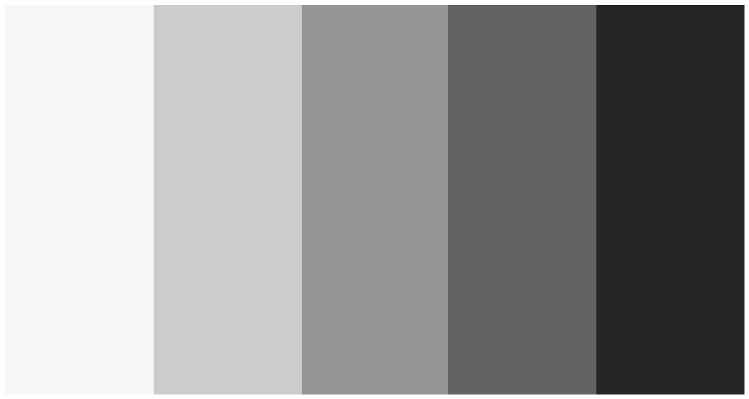
To display the data, we need to install and load the color brewer package. Use the following code to do so and to see the available color ramps.

```
library(RColorBrewer)
display.brewer.all()
```



To make it a little easier to decipher… you can select specific number of classes and color schemes. Here is a grey scheme with 5 classes.

```
display.brewer.pal(5, "Greys")
```



Greys (sequential)

You can see that there are a number of different color ramps that could be used to represent variations in the data on SAT scores.

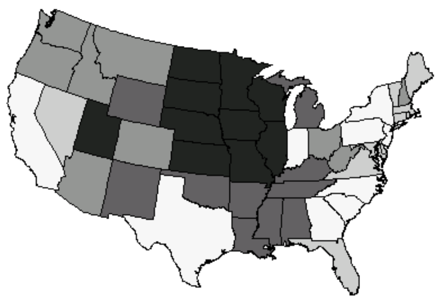
To set class categories install and load the “classInt” package

```
library(classInt)
```

1) Using the following code, we will visualize the distribution of SAT verbal scores broken down into quantiles and displayed in a grey color ramp.

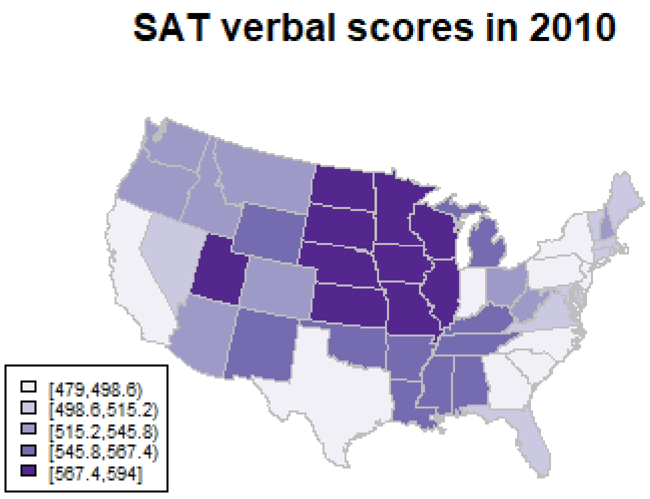
```
plotvar <- states.verbal.laea$verbal
nclr <- 5
plotclr <- brewer.pal(nclr, "Greys")
plotclr
class <- classIntervals(plotvar, nclr, style = "quantile")
class
```

```
colcode <- findColours(class, plotclr, digits = 3)
colcode
plot(states.verbal.laea, col = colcode)
```



2) This can be adjusted for ease of interpretation, by adding color, legends, and titles using the following code.

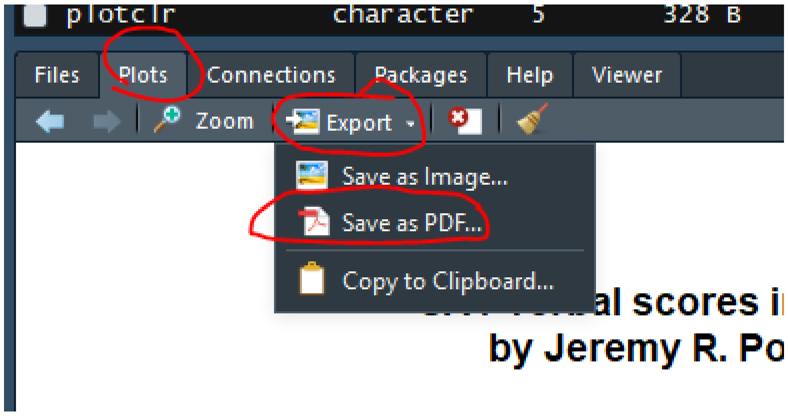
```
plotclr <- brewer.pal(nclr, "Purples")
class <- classIntervals(plotvar, nclr, style = "quantile")
colcode <- findColours(class, plotclr, digits = 3)
plot(states.verbal.laea, col = colcode, border = "grey",axes = F)
title(main = "SAT verbal scores in 2010")
legend("bottomleft", legend = names(attr(colcode,"table")),
      fill = attr(colcode, "palette"), cex=0.55)
```



Ultimately, we see that the R graphics output are not the most aesthetically pleasing so use the following code to export the SptatialPolygonsDataFrame to a .shp file that can be brought directly into QGIS for map production.

```
writeOGR(states.verbal.laea, dsn = “working_directory", layer = "sat_verbal", driver = "ESRI Shapefile")
```

In your working folder, you should now have the sat.shp, sat.dbf, and sat.shx files which can be read into QGIS for further analysis…. Or you simply wanted to produce this plot, you can export the plot as a PDF file using the export menu.



For you lab1 Assignment, read in the “sat_math.csv” file and create a map displaying math scores with monochromatic color ramp (not grey or purple), a title (descriptive and with your name), a legend, and data source notation. This is to be submitted as you Lab1 deliverable to Course Works.