# Spatial Data in R

ASRI 2021

6/17/2021

## Spatial data structures

### Raster Data

- pixelated data where each pixel associated with a specific location.
- associated with:
  - an **extent** (geographic area that the raster covers),
  - a **resolution** (area covered by each pixel), and
  - a **coordinate reference system (CRS)** (describes how location on a 2-D map is related to location on 3-D Earth)

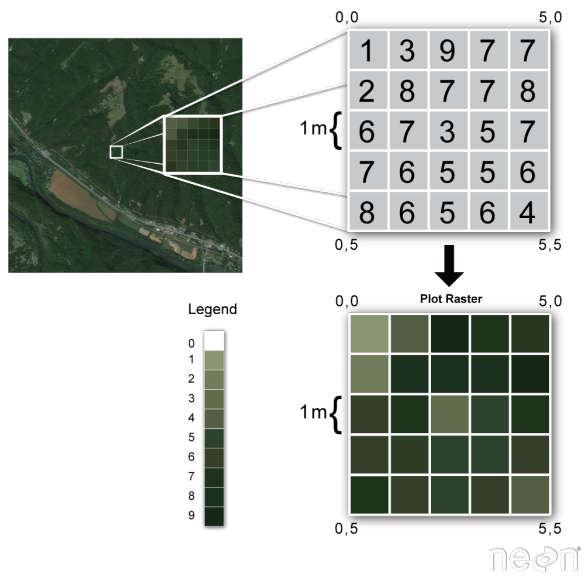



Figure 1: Image borrowed from Data Carpentry Introduction to Geospatial Concepts lesson

In R, we can explore raster files with the help of several useful packages like `raster`. To illustrate, let's download a raster of land use in Arkansas, available (here).

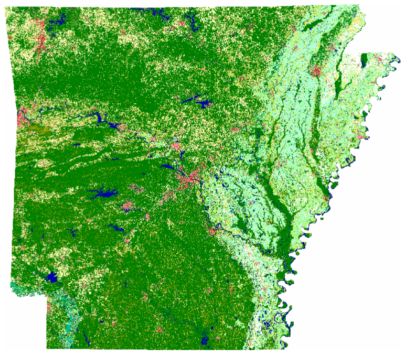
```
library(raster)
```

```
## Loading required package: sp
```

```
### edit this line to point to place where raster in GeoTIFF file format is stored
ar_land_use <- raster('LULC_SUMMER_CAST2004/LULC_SUMMER_CAST2004.tif')

### plot the raster
```

```r
plot(ar_land_use)
```



```r
### take a look at the ar_land_use object
ar_land_use
```

```
## class      : RasterLayer
## dimensions : 13698, 15709, 215181882  (nrow, ncol, ncell)
## resolution : 28.5, 28.5  (x, y)
## extent     : 355081.5, 802788, 3652560, 4042953  (xmin, xmax, ymin, ymax)
## crs        : +proj=utm +zone=15 +datum=NAD83 +units=m +no_defs
## source     : LULC_SUMMER_CAST2004.tif
## names      : LULC_SUMMER_CAST2004
## values     : 0, 210  (min, max)
```

```r
### raster manipulation example
ar_soy_rice <- calc(ar_land_use,
                # substitute values for all pixels where rice/soybeans are not grown to 'NA'
                fun=function(x){x[x != 202 & x != 201] <- NA; return(x)} )

ar_soy_rice
```

```
## class      : RasterLayer
## dimensions : 13698, 15709, 215181882  (nrow, ncol, ncell)
## resolution : 28.5, 28.5  (x, y)
## extent     : 355081.5, 802788, 3652560, 4042953  (xmin, xmax, ymin, ymax)
## crs        : +proj=utm +zone=15 +datum=NAD83 +units=m +no_defs
## source     : r_tmp_2021-11-12_150414_17133_16124.grd
## names      : layer
## values     : 201, 202  (min, max)
```

## Vector Data

- points, lines and polygons
- often stored as shapefiles (.shp)

```r
library(sf)
```

```
## Linking to GEOS 3.8.1, GDAL 3.1.4, PROJ 6.3.1
```

```r
library(maps)  # contains maps of US states with state
               # and county borders that can be retrieved as sf objects
states <- st_as_sf(map("state", plot = FALSE, fill = TRUE))
head(states)
```

```
## Simple feature collection with 6 features and 1 field
```

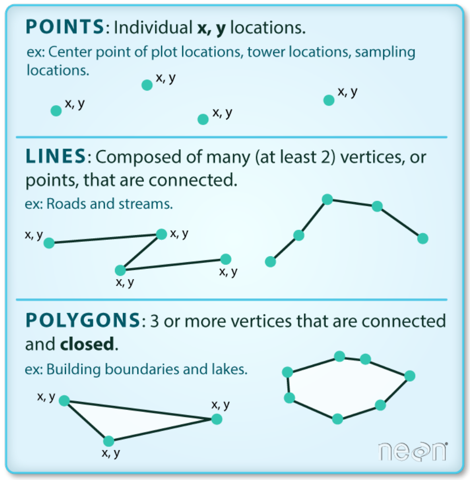



Figure 2: Image borrowed from Data Carpentry Introduction to Geospatial Concepts lesson

```
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -124.3834 ymin: 30.24071 xmax: -71.78015 ymax: 42.04937
## Geodetic CRS:  WGS 84
##            ID                           geom
## 1     alabama MULTIPOLYGON (((-87.46201 3...
## 2     arizona MULTIPOLYGON (((-114.6374 3...
## 3    arkansas MULTIPOLYGON (((-94.05103 3...
## 4  california MULTIPOLYGON (((-120.006 42...
## 5    colorado MULTIPOLYGON (((-102.0552 4...
## 6 connecticut MULTIPOLYGON (((-73.49902 4...
```

```
counties <- st_as_sf(map("county", plot = FALSE, fill = TRUE))
counties <- subset(counties, grepl("arkansas", counties$ID))

### counties is a simple feature object of type 'MULTIPOLYGON'
str(counties)
```

```
## Classes 'sf' and 'data.frame':   75 obs. of  2 variables:
##  $ ID  : chr  "arkansas,arkansas" "arkansas,ashley" "arkansas,baxter" "arkansas,benton" ...
##  $ geom:sfc_MULTIPOLYGON of length 75; first list element: List of 1
##   ..$ :List of 1
##   .. ..$ : num [1:94, 1:2] -91.4 -91.3 -91.3 -91.3 -91.3 ...
##   ..- attr(*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"
##  - attr(*, "sf_column")= chr "geom"
##  - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA
##   ..- attr(*, "names")= chr "ID"
```

```
### however, it has a different CRS than the raster we were using before
crs(counties)
```

```
## CRS arguments: +proj=longlat +datum=WGS84 +no_defs
```

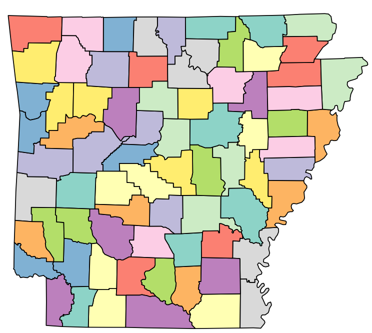
```
### if we want to plot on the same map, we will need to reproject to the same CRS
ar_counties_utm <- st_transform(counties, crs(ar_soy_rice))
crs(ar_counties_utm)
```

```
## CRS arguments:
##  +proj=utm +zone=15 +datum=NAD83 +units=m +no_defs
```

```r
crs(ar_soy_rice) # now they match, we're good!
```

```
## CRS arguments:
##  +proj=utm +zone=15 +datum=NAD83 +units=m +no_defs
```

```r
### take a quick look at the polygon data
plot(ar_counties_utm)
```




## Putting it all together: A map of rice and soybean growing areas in Arkansas in 2004

After the first exercises, you should now have the following:

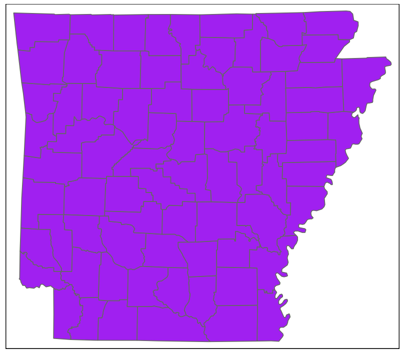
1. A raster file `ar_soy_rice`containing pixels coded as either 'NA' or 202 where land use is classified as 'rice' in the map we downloaded from the AR GIS office
2. A vector file `counties` containing coordinates for polygons of Arkansas county outlines.

Now, we will build a pretty map, with layers of county polygons and a layer of land use pixels. There are many many ways to make maps in R; for most applications I love using `tmap`, which allows you to easily customize maps with a modular, very tidyverse-style syntax. The basic structure involves two tmap elements:

```r
library(tmap)
library(tmaptools)

### a simple tmap
tm_shape(ar_counties_utm) + # first, specify the shape object
  tm_polygons()             # second, specify the aesthetic layer(s) for each shape object

### of course, each layer can be customized in many ways
tm_shape(ar_counties_utm) +
  tm_polygons(col = "purple")
```



```
### see e.g.
?tm_borders
```

Let's include the raster file of rice-growing areas we made earlier. To do this, we will 'layer' each shape element, so that the raster file is layered on top of the county polygon file

```
### a more complicated map
tm_shape(ar_counties_utm) +
  tm_polygons(fill = "grey70") +
tm_shape(ar_soy_rice) +
  tm_raster(palette = "viridis")
```

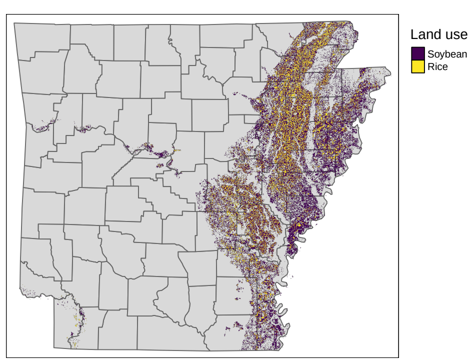
```
## stars_proxy object shown at 1071 by 934 cells.
```

```
### our response (soy/rice) is a categorical variable,
### though it is encoded as a number, hence the continuous legend

### let's add a manual legend instead
ar_map <-
tm_shape(ar_counties_utm) +
  tm_polygons() +
tm_shape(ar_soy_rice) +
  tm_raster(palette = "viridis", legend.show = F) +
tm_add_legend("fill",
              col = viridis::viridis(5)[c(1,5)], # specifying first and fifth color from a vector of 5
              labels = c("Soybean","Rice"),
              title = "Land use") +
tm_layout(legend.outside = T)

ar_map
```

```
## stars_proxy object shown at 1071 by 934 cells.
```



```
### your tmap can be saved like so:
#tmap_save(ar_map, file = "AR_land_use.pdf")
```

## Going further

1. See if you can add a scale bar with `tm_scale_bar` element. Adjust the default scale bar to have fewer breaks and not cover the southeast part of the map
2. Rearrange the layers, and/or use `tm_borders` to create a map where the county lines are visible on top of the raster layer.
3. Add a layer of points on top of the map showing your favorite cit(ies) in Arkansas. See 4f from our PEER workshop tutorial for a quick overview of how to transform a data frame of point coordinates to a spatial object.