1. Create Maps With R Geospatial Classes and Graphics Tools (*Making Maps*)
2. Read and write ESRI Shape Files (*ESRI*)
3. Display T Spatial Objects with Google Maps and Google Earth (*KML*)
4. Read and Display Data from GPS Devices Using R (*GPX*)
5. Overlay Points on Satellite Image / Extract Pixel Values (*Raster*)

Create Maps With R Geospatial Classes and Graphics Tools
Part 1

Create Maps With R Geospatial Classes and Graphics Tools

- R includes a rich set of plotting functions that can be applied to spatial data.
- This example demonstrates how to generate publication-quality maps using these functions, which in many cases can eliminate the need to use dedicated GIS software.

Overview

Here are the typical steps required to produce a map:

- Acquire and read the relevant data layers into R.
- If necessary, transform the data layers into a common spatial reference system, e.g. using the `spTransform` function in the rgdal package to reproject vector data.
- For simplicity, the examples below use data that all share the same coordinate system.

- If necessary, convert the R objects into whatever type is required by the desired plotting function.
- This is unlikely to be necessary unless using one of the R spatial packages that has not adopted the core spatial classes defined in sp, as illustrated by the PBSmapping plotting code below.
- Plot the spatial data layers in the appropriate order and using desired sizing, color, etc.
- This often requires an iterative process of trial-and-error, but ultimately yields code that can saved and used to regenerate the same map or to create another map with different data inputs.

Example 1: Thematic map with points, lines, and polygons

- This example produces a simple thematic map showing the location of major dams in the western United States.
- The data layers are available from the USGS as ESRI shapefiles, a commonly used file format for storing vector geospatial data.
- The map includes point, line, and polygon data layers along with dam labels, a legend, and optional coordinates along the axes.

Output maps and R code

Using sp with base R graphics:

- The **sp** package provides the basis for many other spatially oriented R packages, as it defines a set of classes that have become the de facto standard spatial data types in R.

- However, **sp** also provides some user-level functionality in its own right, including spatial extensions to commonly used R base graphics functions such as plot, points, and lines.
- This makes map production a fairly straightforward task for users who are familiar with conventional R graphics functions.

Using PBSmapping

- The **PBSmapping** package, developed by a team of fisheries scientists, can also be used for generating maps.
- However, note that PBSmapping uses its own custom-defined spatial data types that are optimized to work with various specialized package functions.

- This makes it harder to take advantage of functions defined in the numerous packages that are built on **sp**, although as illustrated in the code used in this example, the maptools package does provide functions that convert between the different formats.

Example 2: Raster base map with point and polygon overlays

- The second sample map displays a satellite image of the Olympic Peninsula region of Washington state along with the outlines of the corresponding counties and the centroid point for each county.
- This example demonstrates how to construct a map consisting of multiple, spatially coinciding data layers, in which the base layer is a grid or raster image and the remaining layers are point, line, and/or polygon vector layers.
- As above, all of the data layers share the same map projection.

Here are the three base layers:

- Raster base (Raster Grid: Satellite Image)
- Polygons base (Polygons: Counties)
- Points base (Points: County Centroids)

Output maps and R code

Using sp with lattice graphics:

- The sp package extends plotting functionality of the lattice package as well as that of the base R graphics system as illustrated above.
- Although lattice functions tend to have more complicated syntax designed to handle the details of creating highly customizable multi-paneled figures (which is not relevant here), for plotting spatial data they also provide niceties such as easy insertion of scale bars and north arrows.
- In addition, the spplot function can accommodate raster data, although we'll see how the raster package lets us use the base plot function for this purpose too.

Using sp and raster with base R graphics:

- Although the **sp** package can work with raster data, the newer raster package offers considerably more capabilities, including an extension of the base plot function for producing raster maps upon which other figure elements can be superimposed.
- Again, this makes map production a fairly straightforward task for users who are familiar with conventional `R` graphics functions.
- Also note that the code below produces the output map approximately 10 times faster than the code using `spplot` above.