**Lab 1: R for GIS – fundamentals**

**Install R packages: rgdal, raster, GISTools. In R consoles, please enter the following commands:**

install.packages('rgdal')

install.packages('raster')

install.packages("GISTools")

**Set working directory, revise the command if you are using an USB drive (e.g., H drive)**

setwd("C:/Geog4984/lab1")

**Import R libraries**

library(raster)

library(rgdal)

library(GISTools)

**1.1 Working with shapefile**

Read a shapefile using R.

In lab1 folder please find [y2010\_51\_pophu.shp]. R uses readOGR() function to read a shapefile. Note there are several other options in R for accessing shapefile. readOGR() is only one of those functions.

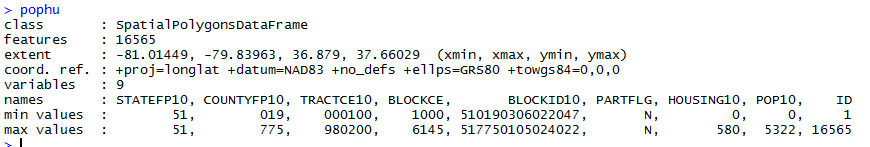
pophu<-readOGR('.',layer='y2010\_51\_pophu')

* ‘.’ tells R program that the shapefile is located in the current directory.
* Layer=’ ’ specifies the name of the shapefile.
* R reads the shapefile as a SpatialPolygonsDataFrame, named as pophu here.

**Some basic information from the SpatialPolygonsDataFrame**

In R console, enter pophu, you’ll see information about features, coordinate reference system, variables, and some summary statistics. For example, POP10 (census 2010) variable has minimum value of 0 and maximum value of 5322.

pophu



**Plot shapefile is easy, just use plot() function.**

plot(pophu)

**Mapping specific attribute can be done using choropleth() function. The following command shows how you map POP10 and HOUSING10 fields:**

choropleth(pophu,pophu$POP10)

choropleth(pophu,pophu$HOUSING10)

**Obtain summary statistics: sum, mean, etc.**

Total population: sum(pophu$POP10)

Average population:mean(pophu$POP10)

**Select a subset of pophu layer based on attributes (similar to select by attributes tool in ArcMap). The following command selects features using POP10>100 criteria and save results to pophu\_sub layer.**

pophu\_sub <- pophu[pophu$POP10>100,]

plot(pophu\_sub)

**Try the following command and plot the result. Here we select features using POP10>100 AND HOUSING10 > 100.**

pophu\_sub <- pophu[pophu$POP10>100&pophu$HOUSING10>100,]

plot(pophu\_sub)

**Lab Questions:**

**Question 1: read another shapefile [counties.shp] using readOGR() function and complete the following table.**

|  |  |
| --- | --- |
| counties.shp | NA |
| #features |  |
| #variables |  |
| coordinate reference system |  |

**Question 2: complete the following table**

|  |  |  |
| --- | --- | --- |
|  | POP2000 | POP2003 |
| TOTAL Population |  |  |
| Mean population |  |  |

**Question 3: Use R to select Montgomery and Pulaski county and plot the two-county layer. Attach your figure here.**

**1.2 Working with raster file**

**R reads raster file using raster() function. The resultant variable keeps all original raster properties, including pixel values, resolution, projection information, etc.**

nlcd<-raster('nlcd.tif')

**After importing the nlcd.tif file, you can plot it using plot() function. You can also type nlcd in R console and see file properties.**

plot(nlcd)

nlcd

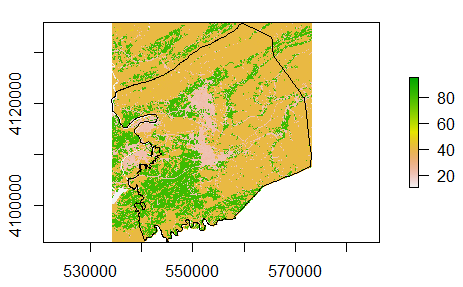
**Clip raster using Montgomery county boundary. First you need to develop the Montgomery county layer. After this step, a crop() function can be used for clipping raster.**

Montgomery <- counties[counties$NAME=='Montgomery',]

nlcdsub <- crop(nlcd, extent(Montgomery))

plot(nlcdsub)

plot(Montgomery,add=TRUE)



**As you can see from the above figure, we need another step to mask out all pixels lie beyond the county polygon. R has a mask() function for this task.**

nlcdsub <- mask(nlcdsub, Montgomery)

plot(nlcdsub)

**Use length() function to report total pixel numbers for a specific land cover type. The following example provides water pixel counts. Water is coded as 11 in the original raster attribute table.**

length(nlcdsub[nlcdsub==11])

**Lab question4: please complete the following table**

|  |  |  |
| --- | --- | --- |
|  | **Total pixel counts** | **Total area in km2** |
| **Deciduous forest (code=41)** |  |  |
| **Evergreen forest (code=42)** |  |  |
| **Mixed forest (code=43)** |  |  |

**Reclassify Raster in R. Reclassify is rather easy in R. Please see the following example where we relabel all forest classes to 1 and the other pixels to 0.**

forest <- nlcdsub

forest[forest>=41&forest<=43]=1

forest[forest!=1]=0

plot(forest)

**Lab question 5: please generate a map showing all urban pixels (urban class code = 21, 22,23,24)**