GIS in R Command Cheat Sheet

September 1, 2015

Installation of Relevant Packages

Packages:

- `sp`: tools for spatial data of all types
- `raster`: extra tools for very large raster datasets
- `rgdal`: tools for reading and writing files in different formats

Installation:
Update R to version > 3.1.
On Windows:

- `install.packages(c(‘‘sp’’,‘‘raster’’))`
- `install.packages(‘‘rgdal’’)`

On OSX:

- `install.packages(c(‘‘sp’’,‘‘raster’’))`
- Download and install GDAL Complete
- Download rgdal package.
- Open .dmg file and place `rgdal_0.9-1.tgz` on desktop.
- Run `install.packages("~/Desktop/rgdal_0.9-1.tgz",repos=NULL)`

Vector Data

Creating Spatial Objects From Scratch

Points:
Points: `SpatialPoints([matrix of coordinates])`

- Note: if latitude and longitude coordinates, must be ordered longitude (x-coordinate), latitude (y-coordinate)

Points with DF: `SpatialPointsDataFrame([Spatial Points Obj], [DataFrame])`

Polygons:
Polygon (basic shapes): `Polygon([matrix of coordinates of vertices])`
Polygons (single “observations” potentially consisting of several basic shapes):
`Polygons([list of Polygon Objs], [names for Polygons])`
Collection of “observations”: `SpatialPolygons([list of Polygons Objs], [names for Polygons])`

- *SpatialPolygons are the R analogue of a shapefile*

Spatial Polygons with DF: `SpatialPolygonsDataFrame([SpatialPolygons Obj, DataFrame])`

Loading Spatial Objects from Files

GPS Coordinates in Table:

1. Use `read.csv()` to import DataFrame with lat long coordinates.
2. `coordinates([DataFrame]) <- c([name of column with long],[name of column with lat])`
    - Note reverse ordering: longitude (x-coordinate), then latitude (y-coordinate).

Vector-Based Files:

```
data <- readOGR(dsn=[path to FOLDER holding data], layer=[name of shapefile in folder])
```

- Note: do not include extension (like `.shp` in `layer` argument)

Interrogating Spatial Objects

Summaries:

Quick summary: `summary([spatial_object])`
Longer summary of contents: `str([spatial_object])`
Full list of contents: `attributes([spatial_object])`
Check if projected: `is.projected([spatial_object])`

Extract Attributes:

Bounding Box: `bbox([spatial_object])`
Get full projection info: `proj4string([spatial_object])`
Get associated coordinates: `coordinates([spatial_object])`

Managing Projections

Projection code database

Assigning projection by EPSG code: `proj4string([spatial object]) <-CRS("+init=EPSG:4326")`

Raster Data

Creating Spatial Objects From Scratch

New Grid Topology:

```
newtopology <- GridTopology(cellcentre.offset=[center of South-West cell],
                            cellsize=[x and y size of each cell],
                            cells.dim=[number of cells in x and y])
```

Example: `newtopology <- GridTopology(cellcentre.offset=c(0,0), cellsize=c(1,1), cells.dim=c(5,5))`

Add Data:

```
SpGdf <- SpatialGridDataFrame([grid topology object], [DataFrame])
```

- DataFrame must have as many rows as topology has cells. Values in DataFrame are associated with cells in order starting in top left cell, moving across row left to row, then moving down one row and repeating, ending in bottom right cell.

Odds and Ends

Open-Source GIS Software Acronyms:

GIS tools in R are based on a set of tools developed by the open-source community and which underlie a great many GIS tools beside those available in R, including tools in Python and several stand-alone applications (like QGIS). As a result, there are a number of acronyms you're likely to find if you start googling GIS tools – here's a quick guide to them.

- **OSGeo**: Open-Source Geospatial Foundation; the group the manages the ecosystem of open-source GIS software.
- **GDAL**: Geospatial Data Abstraction Library. Once upon a time, OSGeo published two sets of tools – OGR for working with vector data, and GDAL for working with raster data. In recent years, however, these tools have converged, so GDAL *usually* used to refer to the full library created by OSGeo. In R, however, the older meanings often still apply, which is why `readGDAL()` is for reading raster data and `readOGR()` is for reading vector data.

- **OGR**: OpenGIS Simple Features Reference (I think?). At one time OGR was the set of tools published by OSGeo for manipulating vector data. OGR is now officially a part of GDAL (which is why it comes in the `rgdal` library).
- **proj4**: proj4 is standard format for describing projections.
- **GRASS**: An OSGeo platform designed to unify the GDAL tools in a graphical user interface.
- **QGIS**: An open-source program designed specifically to be an alternative to ArcGIS based on the GDAL library.

Want to know more? Check out the OSGeo FAQs!