GIS in R Command Cheat Sheet

September 12, 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])`

Lines:

Line (single geometric line): `Line([matrix of coordinates of vertices])`
Lines (single “observations” potentially consisting of several basic lines, like a river):
`Lines([list of Line Objs], [names for Line objs])`
SpatialLines (collection of “observations”, like shapefile):
`SpatialLines([list of Lines Objs], [names for Lines objs])`
Spatial Lines with DF: `SpatialLinesDataFrame([SpatialLines Obj, DataFrame])`

Polygons:

Polygon (one geometric shape defined by a single enclosing line):
`Polygon([matrix of coordinates of vertices])`
Polygons (single “observations” potentially consisting of several basic shapes):

```
      Polygons([list of Polygon Objs], [names for Polygon objs])
```

SpatialPolygons (collection of "observations", like shapefile):

```
      SpatialPolygons([list of Polygons Objs], [names for Polygons objs])
```

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

Extract Attributes:

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

Managing Projections

Projection code database

Assigning projection by EPSG code: `proj4string([Spatial obj]) <-CRS("+init=EPSG:4326")`
Get projection from Spatial obj: `proj4string([Spatial obj])`
Re-project:

```
  newProjection <- CRS("projection string goes here")
  spTransform([Spatial object],newProjection)
```

Raster Data

Creating Rasters From Scratch

Grid Topology (the skeleton):

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

SpatialGridDataFrame (skeleton + data):

```
  SpGdf <- SpatialGridDataFrame([GridTopology obj], [DataFrame])
```

- Each DataFrame *column* becomes different variable.
- Length of columns should match total number of cells in GridTopology obj
- DataFrame entries associated with cells in order, with top left cell as *1*, increasing left to right, then top to bottom, ending with bottom right cell.

Loading Spatial Objects from Files

```
dem <- readGDAL("file name.fileextension")
```

- Pass the entire filename – path, filename, and extension – unlike in `readOGR()`.

Interrogating Spatial Objects

Summaries:

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

Extract Attributes:

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

Managing Projections

Projection code database

Assigning projection by EPSG code: `proj4string([Spatial obj]) <-CRS("+init=EPSG:4326")`
Get projection from Spatial obj: `proj4string([Spatial obj])`
Re-project:

```
newProjection <- CRS("projection string goes here")
spTransform([Spatial object],newProjection)
```