R Spatial Reference Card

Robin

10/07/2014

R has a suite of powerful and effective spatial data processing tools. It is the purpose of this reference card to summarise the most important and frequently used packages and functions. Some of the commands use real datasets from the 'Creating-maps-in-R' GitHub repository. Unzip this repo and run the commands from the folder's root directory to see the commands in action, or use R's help functions. Using the example of the `help` command, type: `?help` (for information on a specific function), `??help` (to search for word in R's documentation) and `example(help)` (for pre-made examples of that function).

Key packages

Additional packages must be installed (using `install.packages('packageName')`) on top of R's default packages to use R's spatial capabilities. Packages are loaded via `library(packageName)`. Some of the most important and frequently used spatial packages are:

- **sp**: basis of most other spatial R packages - provides Spatial* classes
- **rgdal**: R's interface to the GDAL data import library
- **rgeos**: provides a number of spatial functions
- **maptools**: tools for handling spatial objects
- **raster**: provides functions for raster data

There are hundreds of additional R packages described on the Comprehensive R Archive Network's (CRAN) website under the *spatial view*.

R's Spatial* classes

To work with spatial data, R uses *classes* that are more complex that the default data classes in R's base package; *S3* `vector`, `list` and `data.frame` s. Spatial object classes can be identified with the function `class(objectName)`. Each contains a number of different data *slots*: sub-classes of specific types. Some important spatial classes provided by the `sp` package include:

- `SpatialPoints`: point data, each point represented by an x and y coordinate
- `SpatialLines`: line data, composed of lists of the `Lines` sub-class or *slot*
- `SpatialPolygons`: polygon data, with `Polygons` and `Polygon` *slots*
- `SpatialPixels`: a spatial class for raster data comprised of pixels

It is important to note that each of the above classes cannot contain non-spatial attribute data. First they must be converted into a class of the same name but suffixed with `DataFrame`. Thus `spPdf <- SpatialPointsDataFrame(spP, df)` will create a new object containing spatial and non-spatial information.

The attribute data of the spPdf object can be accessed in this instance by `spPdf@data`: the `@data` notation refers to the `data` slot of the new object. More fundamental spatial classes, which are in fact subsets of the `Spatial*` type classes listed above, include `bbox` (the object's bounding box) and `proj4string` (the projection of the object, which may be `NA`).

Loading and querying spatial data

```
library(rgdal)
lnd_sport <- readOGR("data/", "london_sport") # this loads a shapefile and names it lnd_sport
```

Allocating and changing projection

Query the current projection of an object:

```
proj4string(lnd_sport)
```

```
## [1] "+proj=tmerc +lat_0=49 +lon_0=-2 +k=0.9996012717 ..."
```

Change the Coordinate Reference System (CRS) if it has been incorrectly assigned:

```
proj4string(lnd_sport) <- CRS("+init=epsg:27700")
```

Note a warning message should appear which states that the coordinate reference system has been changed but the data has not been transformed. To transform the data use:

```
lnd_sport.wgs84 <-spTransform(lnd_sport, CRS("+init=epsg:4326"))
```

`epsg=` allows access to a very wide range of CRS - see the spatialreference.org/ website for details.

Spatial subsetting

Spatial aggregation

Spatial graphics

Basemaps and advanced functions