Using the R— GRASS interface

Current status

by Roger Bivand

Introduction

Interfaces between GRASS and R, the open source data analysis and statistical programming environment, have existed for some time. Details of the interface between GRASS 6 and R were described two years ago in Bivand (2005), but since then things have got a lot simpler.

Intermediate temporary files are the chosen solution for the GRASS 6 interface: **spgrass6**, using shapefiles for vector data and BIL binaries for raster data. R is started from within a GRASS session from the command line, and the **spgrass6** loaded with its dependencies, with the R interface being used to access and update GRASS data.

Installing the interface package

The GRASS 6 interface is available from CRAN, the Comprehensive R Archive Network. It depends on three packages, andm if not already available, these (**sp**, **maptools** and **rgdal**) should be installed within R using the `dependencies=` argument:

```
> install.packages("spgrass6", dependencies = TRUE)
```

To install on a server not running a graphical interface, set the CRAN mirror first with:

```
> chooseCRANmirror(graphics = FALSE)
```

The only potential difficulties for installation of these packages from source on Linux, Unix, or MacOS X are with **rgdal**, because of its external dependencies on GDAL and PROJ.4 libraries. On Unix/Linux, note that development files for GDAL are required, not just GDAL itself, if your GDAL was installed binary rather than from source. All the other packages are available as binaries for MacOS X users, but **rgdal** is not. Notes for MacOS X users about installing **rgdal** are to be found on the Rgeo website — see under **rgdal**. Windows binaries are available for all the packages, and work with GRASS 6 under Cygwin.

Using the package

```
> library(spgrass6)
> gmeta6()
```

The examples used here are taken from the "Spearfish" sample data location (South Dakota, USA, 103.86W, 44.49N), perhaps the most typical for GRASS demonstrations. The `gmeta6` function is simply a way of summarising the current settings of the GRASS location and region within which we are working. At the present stage of the interface, raster data transfer is done layer by layer, and uses temporary binary files. The `readRAST6` command here reads elevation values into a `SpatialGridDataFrame` object, treating the values returned as floating point, and the geology categorical layer into a factor:

```
> spear <- readRAST6(c("elevation.dem",
+     "geology"), cat = c(FALSE, TRUE))

> summary(spear)

Object of class SpatialGridDataFrame
Coordinates:
             min     max
coords.x1  589980  609000
coords.x2 4913700 4928010
Is projected: TRUE
proj4string :
[+proj=utm +zone=13 +a=6378206.4
+rf=294.9786982 +no_defs
+nadgrids=/home/rsb/topics/grass63/grass-6.3.cvs/etc/nad/conus
+to_meter=1.0]
Number of points: 2
Grid attributes:
  cellcentre.offset cellsize cells.dim
1            589995       30       634
2           4913715       30       477
Data attributes:
 elevation.dem        geology
 Min.   : 1066   sandstone:74959
 1st Qu.: 1200   limestone:61355
 Median : 1316   shale    :46423
 Mean   : 1354   sand     :36561
 3rd Qu.: 1488   igneous  :36534
 Max.   : 1840   (Other)  :37636
 NA's   :10101   NA's     : 8950
```

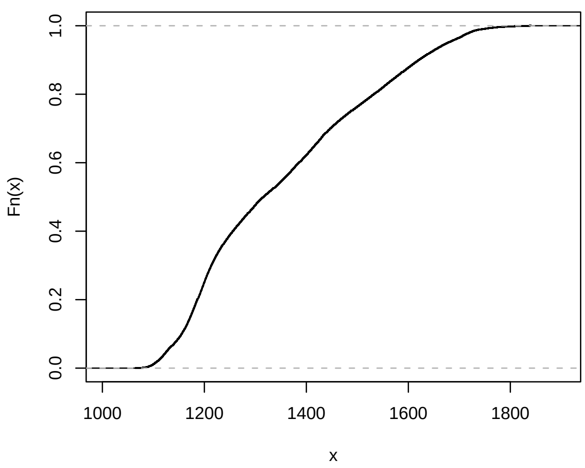

Figure 1: Empirical cumulative distribution function of elevation for the Spearfish location.

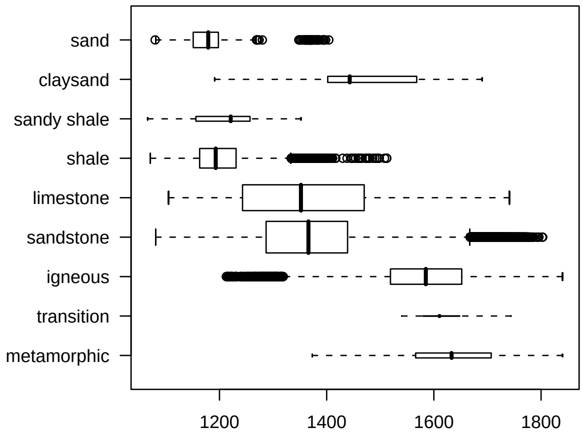

Figure 2: Boxplots of elevation by geology category, Spearfish location.

When the `cat=` argument is set to `TRUE`, the GRASS category labels are imported and used as factor levels; checking back, we can see that they agree:

```
> table(spear$geology)

metamorphic  transition     igneous
      11693         142       36534
  sandstone   limestone       shale
      74959       61355       46423
sandy shale    claysand        sand
      11266       14535       36561

> system("r.stats --q -cl geology",
+     intern = TRUE)

 [1] "1 metamorphic 11693"
 [2] "2 transition 142"
 [3] "3 igneous 36534"
 [4] "4 sandstone 74959"
 [5] "5 limestone 61355"
 [6] "6 shale 46423"
 [7] "7 sandy shale 11266"
 [8] "8 claysand 14535"
 [9] "9 sand 36561"
[10] "* no data 8950"
```

Figure 1 shows an empirical cumulative distribution plot of the elevation values, giving readings of the proportion of the study area under chosen elevations. In turn Figure 2 shows a simple boxplot of elevation by geology category, with widths proportional to the share of the geology category in the total area. We have used the `readRAST6` function to read from GRASS rasters into R; the `writeRAST6` function allows a single named column of a SpatialGridDataFrame object to be exported to GRASS.

The **spgrass6** package also provides functions to move vector features and associated attribute data to R and back again. The `readVECT6` function is used for importing vector data into R, and `writeVECT6` for exporting to GRASS:

```
> bugsDF <- readVECT6("bugsites")
> vInfo("streams")

    points      lines boundaries  centroids
         0        104         12          4
     areas    islands      faces    kernels
         4          4          0          0

> streams <- readVECT6("streams", type = "line,boundary",
+     remove.duplicates = FALSE)
```

The `remove.duplicates=` argument is set to TRUE when there are only for example lines or areas, and the number present is greater than the data count (the number of rows in the attribute data table). The `type=` argument is used to override type detection when multiple types are non-zero, as here, where we choose lines and boundaries, but the function guesses areas, returning just filled water bodies.

Because the mechanism used for passing information concerning the GRASS location coordinate reference system differs slightly between raster and vector, the PROJ.4 strings often differ slightly, even though the actual CRS is the same. We can see that the representation for the point locations of beetle sites does differ here; the vector representation is more in accord with standard PROJ.4 notation than that for the raster layers, even though they are the same. In the summary of the `spear` object above, the ellipsoid was represented by `+a=` and `+rf=` tags instead of the `+ellps=` tag using the `clrk66` value:

```
> summary(bugsDF)

Object of class SpatialPointsDataFrame
Coordinates:
              min     max
coords.x1  590232  608471
coords.x2 4914096 4920512
Is projected: TRUE
proj4string :
[+proj=utm +zone=13 +ellps=clrk66
+datum=NAD27 +units=m +no_defs
+nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat]
Number of points: 90
Data attributes:
      cat                  str1
 Min.   : 1.00   Beetle site:90
 1st Qu.:23.25
 Median :45.50
 Mean   :45.50
 3rd Qu.:67.75
 Max.   :90.00
```

This necessitates manual assignment from one representation to the other on occasion, and is due to GRASS using non-standard but equivalent extensions to PROJ.4.

There are number of helper functions in the **spgrass6** package, one `gmeta2grd` to generate a GridTopology object from the current GRASS region settings. This is typically used for interpolation from point data to a raster grid, and may be masked by coercion from a SpatialGrid to a SpatialPixels object having set cells outside the study area to NA. A second utility function for vector data uses the fact that GRASS 6 uses a topological vector data model. The `vect2neigh` function returns a data frame with the left and right neighbours of arcs on polygon boundaries, together with the length of the arcs. This can be used to modify the weighting of polygon contiguities based on the length of shared boundaries. Like GRASS, GDAL/OGR, PROJ.4, and other OSGeo projects, the functions offered by **spgrass6** are changing, and current help pages should be consulted to check correct usage.

Bibliography

Bivand, R. S., (2005) Interfacing GRASS 6 and R. *GRASS Newsletter*, 3, 11–16.

Roger Bivand
Economic Geography Section, Department of Economics, Norwegian School of Economics and Business Administration, Bergen, Norway
http://www.r-project.org/Rgeo
Roger.Bivand@nhh.no