Interfacing GRASS 6 and R: current status

Roger Bivand*

August 2006

1 Introduction

As Wegmann and Lennert (2005) show in the 2004 GRASS user survey, users of GRASS have found the interface between GRASS 5 and the R data analysis programming language and environment of at least some value. The R interface was mentioned by 119 respondents (40 %) in (Wegmann and Lennert, 2005, p. 15, Fig. 57), so that, despite the command line interface and syntax complications of R, the interface has served some purposes (see for example Grohmann (2004)). The interface as documented first in Bivand (2000), and fully in Neteler and Mitasova (2004), works with GRASS releases up to and including GRASS 5.4.0. This note is intended to show which design choices have been made when interfacing GRASS 6 and R, and how much progress has been made since the GRASS News note of mid 2005 on which this is based (Bivand, 2005).

The GRASS 5 interface to R is an R contributed package, and is available from CRAN, the Comprehensive R Archive Network as described by (Neteler and Mitasova, 2004, section 13.2). It ships with a snapshot of a subset of source files from the core GRASS libraries. Some of these files have been modified to suit the changed setting of a continuously open interface, and cannot readily be merged back into the main code base. When work on the GRASS 5 interface began, both computer speed, memory, and disc capacity were much greater problems than at present, and it was important to move from running GRASS through the R `system()` function to accessing GRASS data directly.

Things have changed, and a fresh start seems attractive for the interface between R and GRASS 6. The package, named **spgrass6**, has advanced a good deal since mid 2005, and is now available from CRAN. It depends on **sp**, **maptools**, and **rgdal**, all of which are on CRAN.

1.1 Installing the interface package

Installation from CRAN is simple on all platforms:

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

Mac OSX users may note that **rgdal** is not available as a `"mac.binary"` install, and should be installed from source by first installing the external software PROJ.4 and GDAL — required by GRASS 6 anyway, installing **sp**, and finally download the

*Economic Geography Section, Department of Economics, Norwegian School of Economics and Business Administration, Helleveien 30, N-5045 Bergen, Norway; Roger.Bivand@nhh.no

rgdal source package to a suitable temporary folder, and install with R CMD INSTALL ... your options ... rgdal*.tar.gz. Your options give the locations, if required, of `-with-gdal-config=`, `-with-proj-include=`, and/or `-with-proj-lib=` all within `-configure-args="` as described in section 1.2.2 of the "Writing R extensions" manual. Once **rgdal** is installed successfully, **spgrass6** can be installed.

2 The sp package

The **sp** package contains defintions of classes for spatial objects, and permits much of the interface to be simplified. In particular, using these classes means that no special display functions are required. The classes provide "slots" for projection and other region or window defining structural data, and to a certain extent take over the role of the `grassmeta` object from the GRASS 5 interface.

The package is being developed by a team of authors and maintainers of R contributed packages for spatial data analysis, including Edzer Pebesma, Paulo J. Ribeiro, Barry Rowlingson, Virgilio Gómez Rubio, and Roger Bivand. We are very grateful for any suggestions, bug reports, and other ideas to help improve our work.

It currently provides a foundation `Spatial` class, with a bounding box and a projection. These slots are inherited by all the other classes. `SpatialPoints` is a `Spatial` object with coordinates, and `SpatialPointsDataFrame` is a `SpatialPoints` object with data in a data frame slot, one row of data per point coordinate; `SpatialPoints` must have at least 2 dimensions. Data frames are the main work-horse of computing in R, they are very much like data base tables, where the number of rows (records) must be the same for each column, but the columns (fields) can vary by type.

Raster data are handled in the `SpatialPixels` and `SpatialGrid` objects, and their associated `*DataFrame` objects. The difference between the two is that while a `SpatialGrid` object has to be a full, rectangular grid, a `SpatialPixels` object can be incomplete. Both have a grid slot with a `GridTopology` object defining the grid itself.

Vector data of more complex types than point coordinates are handled as lines or rings, in hierarchies of objects. The top-level objects are `SpatialLines` and `SpatialPolygons`, and can be bound to object attributes in data frames, one data frame row per member of the `SpatialLines` or `SpatialPolygons` objects. The `SpatialLines` and `SpatialPolygons` objects are built of lists of `Lines` and `Polygons` objects, which themselves are lists of atomic `Line` and `Polygon` objects. No topology checking is done in **sp** on these objects.

2.1 Using graphics with sp objects

Display functions are provided for **sp** classes using both base and lattice graphics. The motivation for this is to make it possible for interface and data import/export packages, and data analysis packages, to concentrate on their core functions. This provision means that, while the legacy interface to GRASS 5 needed to provide plotting functionality, the new interface can rely on it being provided by **sp** if GRASS data is held in R in **sp** classes. The **sp** display facilities are under active development, with help from the developers of lattice/trellis graphics in R. Lattice graphics are analytic graphics more than presentation graphics, using panelled displays to explore the way different variables condition relationships. A typical example would be to explore anisotropy

in variogram displays by panels showing different directions; much of this has been accomplished in Edzer Pebesma's **gstat** package (Pebesma, 2004).

3 Using the spgrass6 package with raster data

Provided that the **spgrass6** package has been installed following the packages it depends upon, **sp** and **rgdal**, the interface is used more or less as before. R is started from within a GRASS session from the command line, and the **spgrass6** loaded with its dependencies:

```
> library(spgrass6)
```

The sample location used here is Spearfish, with the default region settings:

```
> system("g.region -g3")
```

```
n=4928010
s=4913700
w=589980
e=609000
t=1
b=0
nsres=30
nsres3=30
ewres=30
ewres3=30
tbres=1
rows=477
cols=634
rows3=477
cols3=634
depths=1
```

At the present stage of the interface, raster data transfer is done layer by layer, and uses temporary binary files. The following command reads the soil pH values into a `SpatialGridDataFrame` object, treating the values returned as floating point (double in R):

```
> soilsph <- readRAST6("soils.ph", ignore.stderr = TRUE)
```

```
> summary(soilsph)
```

```
Object of class SpatialGridDataFrame
Coordinates:
              min     max
coords.x1  589980  609000
coords.x2 4913700 4928010
Is projected: TRUE
proj4string : []
 proj4string : [+proj=utm]
 proj4string : [+zone=13]
 proj4string : [+a=6378206.4]
 proj4string : [+rf=294.9786982]
 proj4string : [+no_defs]
 proj4string : [+nadgrids=/home/rsb/topics/grass62_0820/grass-6.2.cvs/etc/nad/conus]
Number of points: 2
Grid attributes:
  cellcentre.offset cellsize cells.dim
1            589995       30       634
2           4913715       30       477
Data attributes:
    soils.ph
```

```
Min.   :    1.000
1st Qu.:    3.000
Median :    4.000
Mean   :    3.380
3rd Qu.:    4.000
Max.   :    5.000
NA's   :17750.000
```

Figure 1: Soil pH values for Spearfish

We can display the data using an `image` function for the data object class; the result is shown in Figure 1:

```
> table(soilsph$soils.ph)

    1     2     3     4     5
  698 45672 84102 153051  1145
> image(soilsph, "soils.ph", col = rev(cm.colors(6)))
> legend("bottom", legend = 0:5, fill = rev(cm.colors(6)),
+     cex = 0.8, bty = "n", horiz = TRUE)
```

A feature of the legacy interface was the ability to move category labels to R; this is at present emulated by matching the labels reported by GRASS using `r.stats -l` to the integer values present in the raster data. This is implemented in the `readRAST6` function, when the `cat=` argument is set to `TRUE` as appropriate. As before, category layers are represented in R as `factor` columns:

```
> spear <- readRAST6(c("elevation.dem", "landcover.30m"),
+     cat = c(FALSE, TRUE), ignore.stderr = TRUE)
> summary(spear)
Object of class SpatialGridDataFrame
Coordinates:
              min     max
coords.x1  589980  609000
coords.x2 4913700 4928010
Is projected: TRUE
proj4string : []
 proj4string : [+proj=utm]
 proj4string : [+zone=13]
 proj4string : [+a=6378206.4]
 proj4string : [+rf=294.9786982]
 proj4string : [+no_defs]
 proj4string : [+nadgrids=/home/rsb/topics/grass62_0820/grass-6.2.cvs/etc/nad/conus]
Number of points: 2
Grid attributes:
  cellcentre.offset cellsize cells.dim
1            589995       30       634
2           4913715       30       477
Data attributes:
 elevation.dem                        landcover.30m
 Min.   : 1066   Evergreen Forest            :135375
 1st Qu.: 1200   Grasslands/Herbaceous       : 67396
 Median : 1316   Pasture/Hay                 : 56263
 Mean   : 1354   Small Grains                :  9023
 3rd Qu.: 1488   Emergent Herbaceous Wetlands:  5673
 Max.   : 1840   (Other)                     : 18587
 NA's   :10101   NA's                        : 10101
```

Checking the imported data against the original counts, we can see that the numbers seem to agree.

```
> system("r.stats -lc landcover.30m")
11 Open Water 30
21 Low Intensity Residential 3370
22 High Intensity Residential 239
23 Commercial/Industrial/Transportation 1444
31 Bare Rock/Sand/Clay 26
32 Quarries/Strip Mines/Gravel Pits 1531
41 Deciduous Forest 5568
42 Evergreen Forest 135375
43 Mixed Forest 379
51 Shrubland 1394
71 Grasslands/Herbaceous 67396
81 Pasture/Hay 56263
82 Row Crops 1058
83 Small Grains 9023
85 Urban/Recreational Grasses 260
91 Woody Wetlands 3288
92 Emergent Herbaceous Wetlands 5673
* no data 10101
```

Figure 2 shows how much of the functionality of the legacy raster interface has been maintained, by relating elevation and landcover categories in the same way as Neteler and Mitasova (2004) relate soil types and elevation (pp. 339–340):

```
> G <- gmeta6()
> lcovareas <- table(spear$landcover.30m) * ((G$ewres *
+     G$nsres)/10000)
> boxplot(elevation.dem ~ landcover.30m, data = spear,
+     medlwd = 1, horizontal = TRUE, las = 1, width = lcovareas)
```

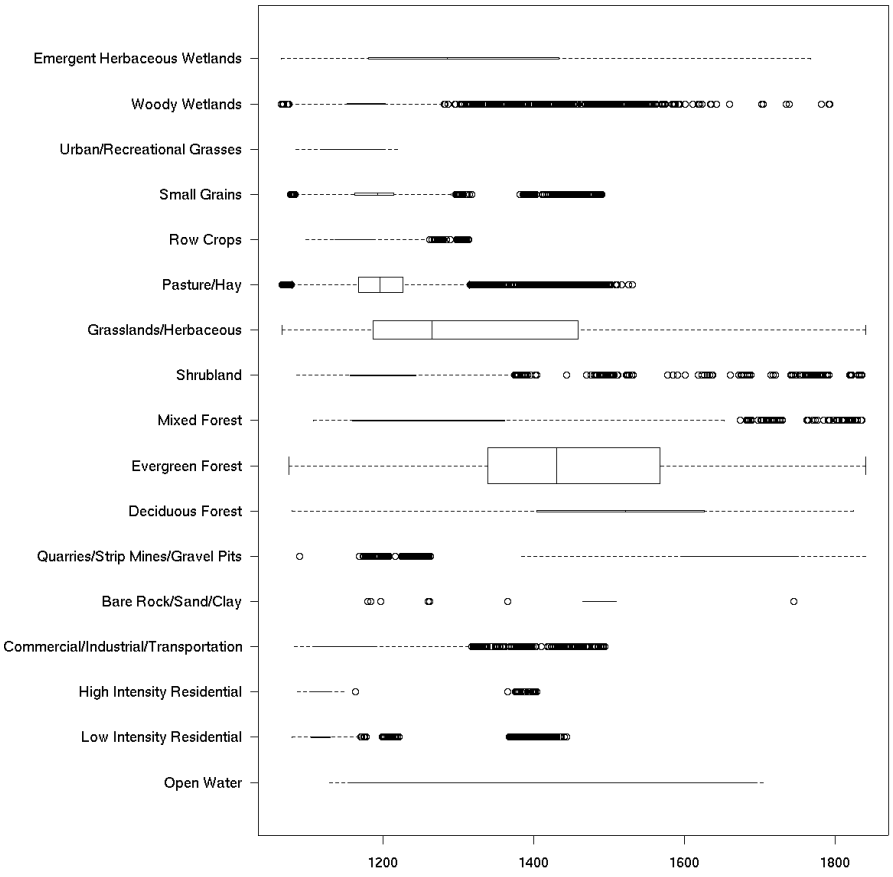

Figure 2: Boxplot of elevation by landcover types in the Spearfish region; the box widths represent the relative areal contributions of the landcover categories.

A function `writeRAST6` for moving numeric raster layers to GRASS from R has also been provided; this uses `r.in.gdal` and again the binary grid format. A helper function has also been written to generate a GridTopology object from the current GRASS region:

```
> grd <- gmeta2grd(ignore.stderr = TRUE)
> grd
```

```
                      X1      X2
cellcentre.offset 589995 4913715
cellsize              30      30
cells.dim            634     477
```

This object can be used to construct points for interpolation, in particular as a SpatialPixelsDataFrame object.

4 Using the spgrass6 package with vector data

Following suggestions by Miha Staut and others, some progress has been made on interfacing GRASS 6 vector data. The package provides functions to move vector features and associated attribute data to R and back again:

```
> bugsDF <- readVECT6("bugsites", ignore.stderr = TRUE)
```

```
> summary(bugsDF)
```

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

Going a little further, a line layer showing stream centerlines can be moved to R, and converted to a `SpatialLinesDataFrame` object:

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

```
> summary(streams)
```

```
Object of class SpatialLinesDataFrame
Coordinates:
         min       max
r1  589443.1  609526.8
r2 4913935.5 4928059.2
```

```
Is projected: TRUE
proj4string : []
 proj4string : [+proj=utm]
 proj4string : [+zone=13]
 proj4string : [+ellps=clrk66]
 proj4string : [+datum=NAD27]
 proj4string : [+units=m]
 proj4string : [+no_defs]
 proj4string : [+nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat]
Data attributes:
      cat        label
 Min.   :0.000   :116
 1st Qu.:1.000
 Median :2.000
 Mean   :1.621
 3rd Qu.:2.000
 Max.   :3.000
```

The choice of type and other decisions about data import can be guided by three helper functions:

```
> vInfo("streams", ignore.stderr = TRUE)
```

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

```
> vColumns("streams", ignore.stderr = TRUE)
```

```
  storageType  name
1     INTEGER   cat
2   CHARACTER label
```

```
> vDataCount("streams", ignore.stderr = TRUE)
```

```
[1] 0
```

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.

Having moved several layers into R, we can display them together by placing the streams and bugsites vector layers on the elevation raster layer:

```
> image(spear, "elevation.dem", col = terrain.colors(20),
+     axes = TRUE)
> plot(streams, col = "blue", add = TRUE)
> plot(bugsDF, pch = 19, col = "grey25", cex = 0.5, add = TRUE)
```

Vector layers may be exported using `writeVECT6`; here we sample 200 points from the current region as represented by the elevation DEM layer, and move them to GRASS:

```
> set.seed(1)
> smple <- overlay(spear, spsample(spear, 200, "random"))
```

```
> summary(smple)
```

```
Object of class SpatialPointsDataFrame
Coordinates:
                min       max
coords.x1  590228.7  608860.9
coords.x2 4914097.6 4927882.6
Is projected: TRUE
```

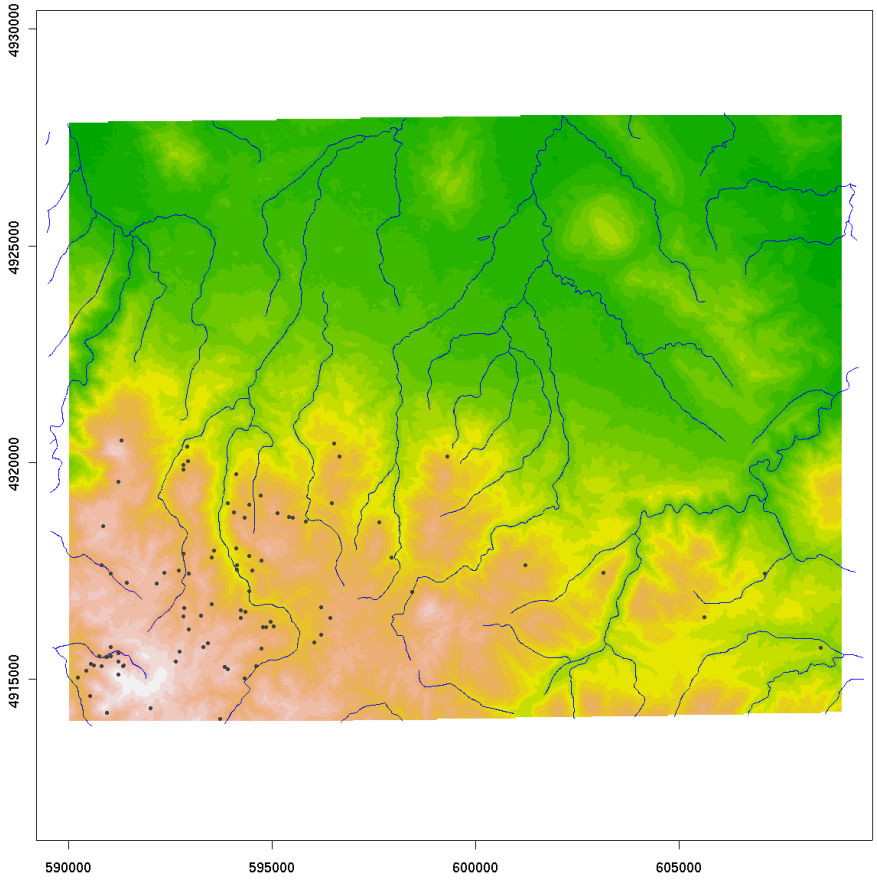

Figure 3: Spearfish elevation map with stream centerlines and bugsites.

```
proj4string : []
 proj4string : [+proj=utm]
 proj4string : [+zone=13]
 proj4string : [+a=6378206.4]
 proj4string : [+rf=294.9786982]
 proj4string : [+no_defs]
 proj4string : [+nadgrids=/home/rsb/topics/grass62_0820/grass-6.2.cvs/etc/nad/conus]
Number of points: 199
Data attributes:
 elevation.dem                       landcover.30m
 Min.   :1107   Evergreen Forest            :94
 1st Qu.:1209   Grasslands/Herbaceous       :56
 Median :1378   Pasture/Hay                 :30
 Mean   :1384   Deciduous Forest            : 5
 3rd Qu.:1542   Small Grains                : 4
 Max.   :1831   Emergent Herbaceous Wetlands: 4
                (Other)                     : 6
```

```
> writeVECT6(smple, "sp_dem", v.in.ogr_flags = "--overwrite",
+     ignore.stderr = TRUE)
```

```
> system("v.category sp_dem option=report")
> vColumns("sp_dem")
```

```
LAYER/TABLE 1/sp_dem:
type       count        min        max
point        199          1        199
line           0          0          0
boundary       0          0          0
centroid       0          0          0
area           0          0          0
all          199          1        199
```

```
       storageType       name
1          INTEGER        cat
2 DOUBLE PRECISION elevation_
3        CHARACTER landcover_
4 DOUBLE PRECISION  coords_x1
5 DOUBLE PRECISION  coords_x2
```

5 Conclusion

The use of classes defined in the **sp** package has made the construction of an interface between GRASS 6 and R more robust than the legacy interface. The new package wraps `system()` calls to GRASS commands in R code to check and control the conversion process. Because the **sp** class objects have a range of display methods, and coersion methods to class types used in other packages for analysing spatial data, these do not need to be part of the interface package. Reports on problems and bugs, and suggestions for the interface package may best be raised on the STATGRASS mailing list[1]. Further work will continue to streamline data transfer using the current region and resolution model, based on shared use of GDAL, OGR, and PROJ.4 in both GRASS and R.

References

Bivand, R. S., (2000) Using the R statistical data analysis language on GRASS 5.0 GIS data base files. *Computers and Geosciences*, 26, 1043–1052.

[1] `http://grass.itc.it/statsgrass/index.php`

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

Grohmann, C. H., (2004) Morphometric analysis in geographic information systems: applications of free software GRASS and R *Computers and Geosciences*, 30, 1055–1067.

Neteler, M. and Mitasova, H., (2004) *Open Source GIS: A GRASS GIS Approach. Second Edition*. Kluwer Academic Publishers, Dordrecht.

Pebesma, E.J., (2004) Multivariable geostatistics in S: the gstat package. *Computers and Geosciences*, 30: 683-691.

Wegmann, M. and Lennert, M., (2005) GRASS User Survey 2004 *GRASS-News*, 2, 2–16.