Using the R statistical data analysis language on GRASS 5.0 GIS data base files*

Roger S. Bivand
Department of Geography,
Norwegian School of Economics and Business Administration,
Breiviksveien 40, N-5045 Bergen, Norway.

11th February 2000

Abstract

Many researchers wish to explore and analyse spatial data, but typical software does not readily permit such integration. This note presents a simple interface between two open source software systems, the GRASS geographical information system, and the R statistical data analysis language. The platform used here is GNU/Linux, because both systems compile and install cleanly; R runs cleanly in Windows environments as well. The interface allows floating point and category data to be passed both ways for raster map layers and sites files; NULL GRASS raster cells interchange with R NA (not available) values. Because both systems are developing rapidly and GRASS database internals changing often, the interface uses ASCII transfer via temporary files generated by standard programs. The interface operates by running R from within the GRASS environment, and issues commands to GRASS programs through the R system() function. The accompanying code is constructed as an R package, and contains wrapper functions for R plotting, and for R analytical functions returning gridded output, such as trend surface and kriging prediction, kernel density estimation of point patterns, and bicubic spline interpolation. These typical spatial analytical techniques, also often available in some form in GIS, are amply buttressed in R by a large range of other statistical and graphical functions, giving substantial insight into the data or results being handled. The interface will be extended to vector data, and will be coordinated with other database integration packages in R and GRASS.
Key words: Geographical information systems, Open source software, Interface package.

1 Introduction

With the release of the open-source GIS GRASS 5.0 in early 1999, opportunities are presented for integration with the open-source R statistical data analysis programming environment (Ihaka and Gentleman, 1996, code obtained from [2]). In the examples presented, R is run interactively within the GRASS 5.0 environment, transfering data by writing and reading temporary text files; the operating system here is GNU/Linux. The note describes the implementation in R of functions needed to move data between GRASS and R, providing the user with a basic interface between the two environments.

Development of the leading Open Source GIS — GRASS — has been moved to Baylor University in Texas, where work on a new release incorporating floating-point raster cell values and NULL values different from zero is now in beta testing (Byars and Clamons, 1998, sources obtained from [3]). In parallel with this, the R statistical and data analysis language, also Open Source, is maturing very rapidly, and can now execute most S and S-PLUS code in an unmodified form. In the past, when S was available on academic license, integration between GRASS and S existed in a loose-coupled form for integer raster cell values sampled at points given in a site layer.

The issues involved in linking two complex and fast-changing programming environments are encapsulated in a comprehensive way in the R functions included in the code accompanying this note. While the progress reported in this paper is based on Open Source Unix-like operating systems, it is worth noting that both GRASS and R have been compiled for MS Windows systems. Programming techniques for R are covered in Venables and Ripley (1997), and in materials available at the R archive [2].

*Paper in process of acceptance at Computers & Geosciences, and is subject to copyright restrictions.

In work to date, the interface used is that of the statistical analysis system, run from within the GIS environment. Given major design differences in memory management — GRASS uses the underlying file system, while R maps all active objects into a static area of memory allocated when the program is started, managed by a garbage collector — and other problems, it has been necessary to decide on a representation suiting the data analysis and visualization tasks being performed. This means here that the statistical programming environment is run from within GRASS, permitting GRASS command line instructions, including those requiring interaction, to be issued from within R using the `system()` function, as shown in the following execution log snippet. Here the system prompt is `$`, the GRASS prompt `GRASS:~/wd >`, and the R prompt `>`. Here, the GRASS `g.list` command listing available site data base files in the current location is accessed from within R, next, the same functionality is achieved with the interface function `list.GRASS()` returning an R character object with sites file names, and finally the GRASS GUI `tcltkgrass` is started from within R, giving point-and-click access to much GRASS functionality, including display:

```
$ grass5.0beta5
... (choice of GRASS location)
Welcome to GRASS 5.0beta5 (December 1999)

Geographic Resources Analysis Support System (GRASS) is Copyright,
1999 by the GRASS Development Team, and licensed under terms of
the Gnu Public License (GPL).

This new release of GRASS is coordinated and produced by the GRASS
Development Team Headquartered at Baylor University with development
sites located at the University of Hannover and University of Illinois.
...
When ready to quit enter:                exit
GRASS:~/wd > R

R : Copyright 2000, The R Development Core Team
Version 0.99.0  (February 7, 2000)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type "?license" or "?licence" for distribution details.
...
Type "q()" to quit R.

> system("g.list sites")
----------------------------------------------
site list files available in mapset rsb:
ex.pcbs.in     ex.pcbslog.in  pcbs1    pcbslog.in  pcbslog2  test  xx
ex.pcbscat.in  pcbs.in       pcbslog  pcbslog1    s.test    tr1

----------------------------------------------
> print(list.GRASS("sites"))
 [1] "ex.pcbs.in"    "ex.pcbslog.in" "pcbs1"         "pcbslog.in"
 [5] "pcbslog2"      "test"          "xx"            "ex.pcbscat.in"
 [9] "pcbs.in"       "pcbslog"       "pcbslog1"      "s.test"
[13] "tr1"
> system("tcltkgrass&")
>
> q()
Save workspace image? [y/n/c]: n
GRASS:~/wd > exit

                 Goodbye from GRASS GIS

$
```

Running under Unix-family operating systems, GRASS only customizes the user's program execution environment, adding specific definitions needed for GRASS programs to be able to find the files and metadata required for their work. GRASS does not then represent a major memory overhead, and R can be launched with plenty of space for its computations. The example from North-West Leicestershire reported below did not need more than 12Mb heap memory for analysis of a data set with 57600 raster cells, and 8Mb heap memory was enough for the other examples.

2 Interface functions

General issues involved in interfacing statistical and GIS software are discussed by Anselin et al. (1993) and Anselin and Bao (1997). The motivation for writing an interface in the interpreted language of one of the linked software systems as presented below is to make a simple interface available to any analyst willing and able to download two open source programs and install them, at present under a Unix-like operating system (see also Bivand, 1996). The resources used here were RedHat Linux 6.1 (2.2.12-20 kernel, C compiler: egcs-2.91.66, Fortran compiler: GNU Fortran 0.5.24-19981002), GRASS 5.0 beta 5, and R-0.99.0.

The accompanying code has been written as an R package, containing R code, full manual pages with examples, and a test data set, described as the third example below. The package may be installed under R by executing `$ R INSTALL GRASS` as with other packages. The functions use a number of packages over and above those found in the standard R distribution to run the examples, but all are readily available from [2]. The manual page for the data set gives details of the parameters needed to initialize the GRASS location for the Pontypool example, and once GRASS has been started using this location, the installation of the package can be checked by executing `GRASS: > R CMD check GRASS`. On a 90MHz Pentium, the exhausive test of package functionality takes several minutes, generating substantial graphical output.

Two raster interface functions are included in the accompanying code, `rast.get()` to read one or more GRASS data base files (raster map layers) into R, with an option to retain the category labels of the values, and `rast.put()` to export single R numerical vectors to GRASS. In both cases, the enhancements to GRASS 5.0 have been used, taking account of floating point numerical values and NULL (not available) values different from zero. In previous releases of GRASS, values of zero might be numerical zero or NULL, and only integer values were used, inhibiting or complicating analysis. The functions are written using the R language, supplemented with a small utility function, `list.GRASS()`, to list existing GRASS data base files. Underlying support is provided by the `gmeta()` function, which retrieves location and region metadata using the GRASS `g.gisenv` and `g.region` commands, and constructs a selection of auxilliary data objects for use in other functions.

The gmeta() function creates an object of class “grassmeta”, using the R method despatch object naming convention. Objects of this class are lists with a wide range of metadata components, some of which are reported by summary.grassmeta(), using the method despatch mechanism. Method despatch is employed to permit generic functions, such as summary() or plot(), to call class-specific functions by concatenating their own names with the class of the object given as their first argument.

Two interface functions are also provided for sites data base files, `sites.get()` and `sites.put()`, which are suited to the revised sites definition implemented in GRASS 5.0. While `rast.put()` only checks that the number of data items being moved from R to a GRASS raster map layer agrees with the number of cells in the current region, `sites.put()` issues a warning if some of the sites are outside the region, and exits without transferring data if all the points are beyond the region defined in the “grassmeta” data object.

At present, R can only read from named files or sockets, while GRASS can only write to named files or pipes. While it would have been possible to use an additional interpreted language, for instance Perl, as glue, it was decided to write ASCII data to, and read ASCII data from, temporary files, despite the consequences for performance of this choice. A further option will be to dynamically load underlying GRASS data base access function libraries into R, but this has been avoided for the present until GRASS 5.0 stabilizes further. ASCII data files are easier to debug than binary integration, although attention needs to be paid to loss of significant digits in floating point transfers.

GRASS map layers and sites data base files are read directly into R data frame objects. Since the GRASS `r.mapcalc` command and R impose similar conventions on map layer and variable names, arithmetically ambiguous names like “a-b” should not occur on either side, and no sanity check is made; layer and variable names are assumed to belong to the subset of names valid in both environments. When category labels are imported and R factors created, substantially more internal processing is involved, both in terms of copies of data in memory and time. R factors are data objects with integer codes and levels labels, and are recognized by a range of R functions, such as `boxplot()`, illustrated below.

The metadata are also used by `plot.grassmeta()`, which provides a simple interface between “grassmeta” data objects and the `image()` function, to give the dimensions of the plotting window, and to reverse the row order of the image matrix. GRASS, as many GIS and image display systems, looks at the world from top left down, while R assumes that data begin from bottom left. Factor variables, originally GRASS map layers with category labels, may be plotted by taking their integer `codes()` values, and passing through an appropriate range for these values in the `image()` function zlim argument.

The following code comprises the `plot.grassmeta()` function. It takes one required argument, a “grassmeta” data object defining the current region, and a number of optional arguments. The first is a numerical vector with data ordered as GRASS raster cells for plotting with `image()`, the next two are labels for the axes passed to `eqscplot()`, and the final ellipsis permits further unspecified options to be passed to `image()`. The test on the first line is needed because the function can be called directly, avoiding method despatch class checking. The next line requires that the MASS package be loaded into R, here because it contains the useful `eqscplot()` function, imposing dimensional equality on the scales of the axes. This function is used to plot a

base map, and if no "layer" argument is given, `plot.grassmeta()` exits, leaving the current plotting device ready for more graphics to be added by functions like `points`, `lines`, or `polygon`. If a layer argument has been given, and if its length is equal to the number of cells in the current region as recorded in the "grassmeta" data object, the `image()` function is called and added to the current plot, with arguments of easting coordinates of raster cell midpoints, northing coordinates of raster cell midpoints, and a suitably arranged table of values from the layer to be mapped. The re-arrangement is made necessary by GRASS having an "upper left" origin by default, as opposed to "lower left" in `image()`.

```
plot.grassmeta <- function(G, layer=NULL, xlab="", ylab="", ...) {
    if (class(G) != "grassmeta") stop("Data not a grass object")
    require(MASS)
    eqscplot(G$xlim, G$ylim, xlab=xlab, ylab=ylab, type="n", ...)
    if (!is.null(layer)) {
        if (length(layer) != G$Ncells)
            stop("GRASS object metadata do not match layer length")
        image(x=G$xseq, y=G$yseq, z=t(matrix(layer[G$reverse],
            nrow=G$Nrow, ncol=G$Ncol, byrow=T)), add=T, ...)
    }
}
```

Finally, wrapper functions are provided in the package for initializing data objects for point pattern analysis, for interpolation from irregular points to rasters, and for output from two-dimensional kernel density estimation, trend surface analysis, and kriging, using the metadata retrieved by `gmeta()`. They are extensively used in the worked examples provided in the package, and require the prior installation of the Pontypool GRASS location, and the following R packages: modreg (standard); akima, and MASS and spatial in the VR collection (optional).

3 Examples

Three brief examples will be presented to illustrate the uses of the interface. The first is taken from the North-West Leicestershire GRASS data set made available with the "GRASS-seeds" tutorial by the Assist project [1]. This covers a 12km by 12km section of central England with 50m by 50m raster cell sizes. Two map layers have been imported into R, "topo", the elevation in meters interpolated from digitised contours, and "landcov", a category labelled land cover classification from Landsat TM data. Exercise B.1 in the "GRASS-seeds" tutorial involves the use of the r.average and r.report commands to calculate average height by land cover category. Here we not only use the R `summary()` function by category: `tapply(topo, landcov, summary)` to add a Tukey five number summary to the mean, as shown in Table 1, but also display in Figure 1 boxplots of elevation by land cover category: `boxplot(topo ~ landcov)`. Such exploratory and graphical statistical functions give a much clearer impression of the underlying distributions than their means.

Table 1: Summary statistics of elevation in meters by land cover categories for North-West Leicestershire.

Land cover	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Arable	40	63	81	104.00	153	270
Industry	40	42	59	83.61	120	220
Pasture	40	75	145	133.50	180	278
Quarry	40	59	161	135.30	177	198
Residential	40	53	60	76.30	80	218
Scrub	40	100	161	148.20	196	270
Water	47	65	99	99.27	131	221
Woodland	40	118	156	142.90	174	248

The second example is from a geomorphometric classification of the topography of Kosovo, based on GTOPO30 elevation data (30 second grid data from [4], and projected to UTM zone 34), described in more detail with code examples by Bivand (1999). This data set is interesting because it is imported into R from floating point GRASS data base files, which also use NULL values to represent cell locations beyond the borders of the province; the data set has 163 rows and 164 columns, with a cell size of 1000m in each direction. Before undertaking the classification, the elevation data were explored as shown in Figure 2, after first having dropped NULL raster cells, using a histogram function (`truehist()`, Venables and Ripley, 1997) permitting control of bin widths and the starting point of the display, and a density trace with default bandwidth (Jacoby, 1997). A hypsometric analysis was performed, and is also shown in Figure 2; unlike Luo (1998), use was made of Pike and Wilson's (1971) result that the hypsometric integral and the elevation-relief ratio are the same. Finally,

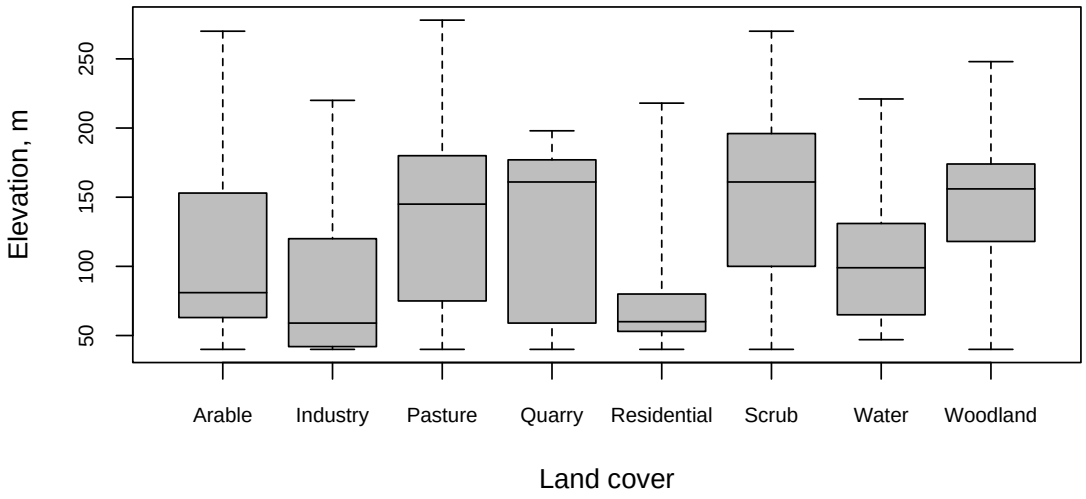

Figure 1: Boxplots of elevation by land cover categories for North-West Leicestershire.

essentially the same diagram was made by plotting the empirical cumulative density function of the elevation data (Jacoby, 1997). The code used to prepare the diagrams in Figure 2 is shown below:

```
> G <- gmeta()
> morph <- rast.get(G, rlist=c("ER","LR","plan","prof","slope","elev"))
> morph1 <- morph[!is.na(morph$elev),]
> mean(morph1$elev)
[1] 816.4051
> median(morph1$elev)
[1] 689.452
> truehist(morph1$elev, col="gray", xlab="elevation", xlim=c(0,2500))
> plot(density(morph1$elev), xlim=c(0,2500))
> rel.elev <- (morph1$elev - min(morph1$elev))/
+ (max(morph1$elev) - min(morph1$elev))
> plot((seq(1:length(rel.elev)))/length(rel.elev),
+ rev(sort(rel.elev)), type="l")
> library(stepfun)
> plot(ecdf(morph1$elev), verticals=F, do.p=F)
```

As Jones (1998) has shown, the Zevenbergen and Thorne (1987) method for computing slopes from gridded elevation data performs better than many alternatives (see also Burrough and McDonnell, 1998). Using this method, slope gradient and plan and profile curvature were added to elevation, local relief in a 3×3 window, and the local elevation-relief ratio, again in a 3×3 window, as a basis for classification, logarithms being taken of three of the variables. Figure 3 shows the topography of Kosovo, and the results of a classification of 10948 cells into 5 classes using the `clara()` function from the cluster package, while Table 2 displays the distributions of elevation by class.

Table 2: Summary statistics of elevation in meters by geomorphometric classes for Kosovo.

Class	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1	290.3	572.1	655.6	690.9	771.5	1811
2	318.1	717.0	908.7	1036.0	1298.0	2244
3	513.1	939.2	1223.0	1327.0	1674.0	2474
4	332.3	589.0	699.6	747.0	856.3	2405
5	293.8	428.9	505.5	495.8	558.1	1023

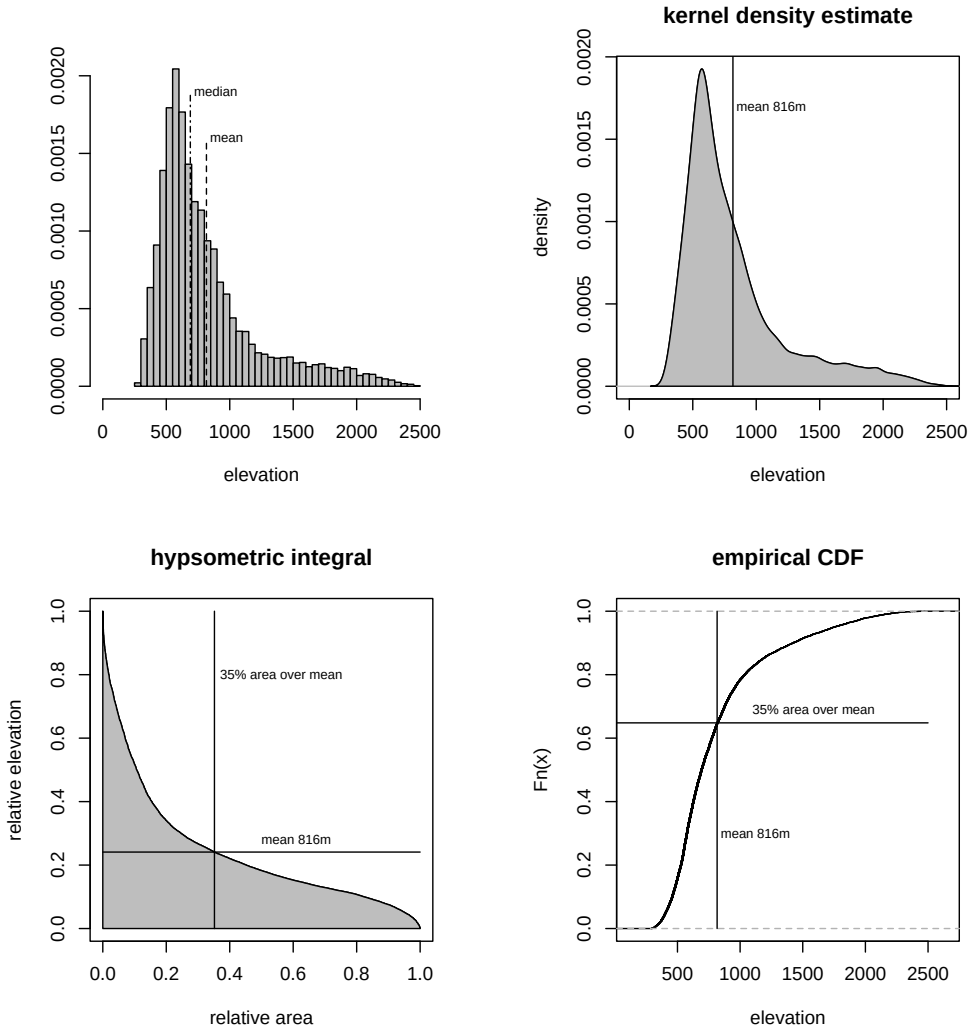

Figure 2: Graphical data analysis of elevation in Kosovo, using a variety of R functions; for comparison with Luo (1998), skewness: 1.583, kurtosis: 2.219.

The third example is the data set distributed with the accompanying code. It records contamination of the environment with polychlorinated biphenyls around the site of a large plant for the incineration of chemical wastes near Pontypool, South Wales. The data frame has 70 rows and 3 columns: UTM zone 30 eastings and northings, and standardised cumulated scores for seven types of PCB. The original data were positioned in relation to National Grid coordinates, and have been reprojected to UTM zone 30, using the WGS84 ellipsoid (Bailey and Gatrell, 1995, pages 149-150). The following code shows the interpolation of surface values of logarithms of contaminant values for the GRASS raster grid using `interp.new.G`, a wrapper function for `interp.new` in the `akima` package, which implements Akima's new code for R (Akima, 1996); NAs are inserted into the grid outside the convex hull of the points. Next, the R data frame is transferred to GRASS as sites floating point data, taking logarithms of PCBs values, and interpolated to the whole GRASS grid using `s.surf.rst`, by regularized splines with tension (Mitasova et al., 1995). It may be noted that the `system()` function would permit the analyst to loop over a sequence of tension and smooth values, storing the results in separate output raster files. Finally, the output of `s.surf.rst` is moved from GRASS to R, and `summary()` taken of the differences between the two interpolations. The graphic output is shown in Figure 4; some of the code needed for icing is omitted in the extract below.

```
> data(pcbs)
> G <- gmeta()
> summary(G)
Data from GRASS 5.0 LOCATION pontypool with 88 columns and 108 rows.
The west-east range is: 68000, 70200, and the south-north: 74500, 77200;
West-east cell sizes are 25 units, and south-north 25 units.
> akspl <- interp.new.G(G, pcbs$east, pcbs$north, log(pcbs$pcbs))
```

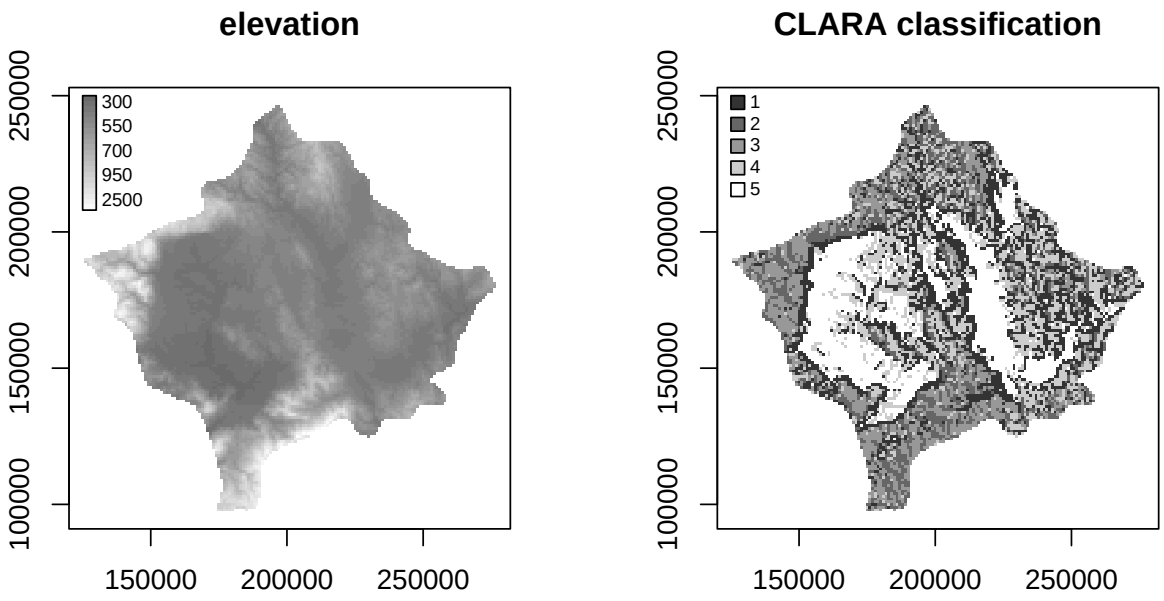

Figure 3: Left: elevation in Kosovo; right: geomorphometric classification into five surface topography classes; scale in grid meters (both figures were made using plot.grassmeta() supplemented by extra commands to construct and place the legends).

```
> summary(akspl)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
   1.026    2.104    2.390    2.511    2.719    8.792 3456.000
> plot(G, akspl, col=gray(22:8/24))
> title("Bicubic spline interpolation")
> sites.put(G, lname="ex.pcbslog.in", east=pcbs$east,
+ north=pcbs$north, var=log(pcbs$pcbs))
> system(paste("s.surf.rst input=ex.pcbslog.in elev=ex.rst",
+ "tension=160 smooth=0.0 segmax=700"))
> exget <- rast.get(G, rlist=c("ex.rst"))
> summary(exget$ex.rst)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.9214  2.0520  2.2130  2.3440  2.4500  8.3640
> plot(G, exget$ex.rst, col=gray(22:8/24))
> title("Regularized spline with tension")
> summary(akspl-exget$ex.rst)
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.       NA's
-1.771e+00 -1.053e-01  9.931e-03  1.317e-02  1.256e-01  2.027e+00  3.456e+03
```

4 Conclusion

The code accompanying this note is a first step in making GRASS data base files available for analysis in R, and could be extended by reading data into R directly in binary form, and by including file types other than raster and sites, such as vector data, and will be coordinated with other database integration packages in R and GRASS. The functions presented do not use language constructs specific to R, and should therefore work with S; since they use simple file transfer, they could also easily be adapted to work with other GIS with command line interfaces at the Unix shell level. Extension to graphical user interfaces is not a major priority, because users of both GRASS and R can be assumed to be willing to use command line interfaces. Further, access to open source platforms for these software systems is only impeded by organisations unwilling to allow their staff or students to install GNU/Linux or other Unix-based alternatives. R does run on Windows platforms, and GRASS may be compiled, but the obvious route for analysts with work to do is to use Open Source platforms, guaranteeing access to as much or little source code as is needed to tackle the problem at hand.

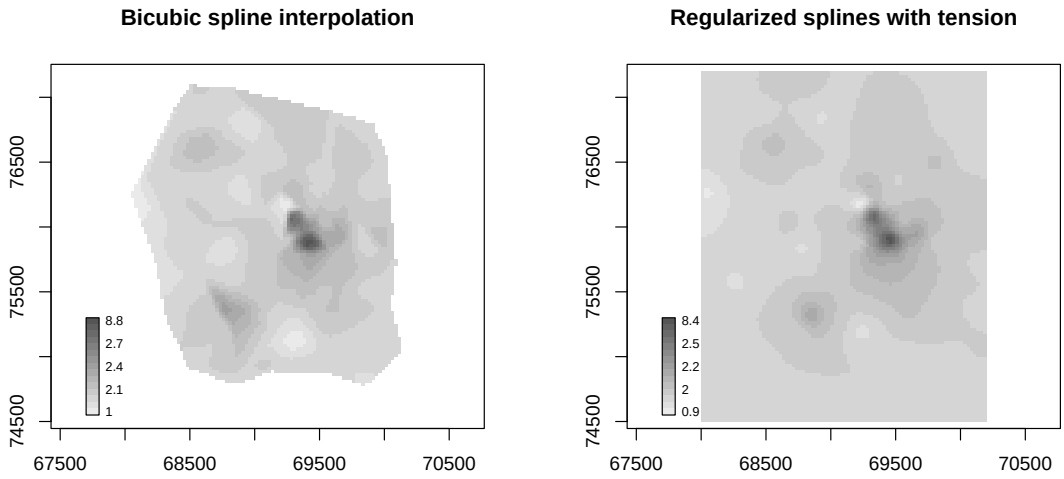

Figure 4: Left: Bicubic spline interpolation within sample convex hull `interp.new()` (Akima, 1996); right: Regularized splines with tension for full grid `s.surf.rst` (Mitasova et al., 1995).

References

Akima, H., 1996. Algorithm 761: scattered-data surface fitting that has the accuracy of a cubic polynomial. ACM Transactions on Mathematical Software, 22 (3), 362-371

Anselin, L. and Bao S. 1997. Exploratory spatial data analysis linking SpaceStat and ArcView. In: Fischer, M. M., Getis, A. (Eds) Recent developments in spatial analysis. Springer-Verlag, Berlin, pp. 35–59.

Anselin, L., Dodson, R. and Hudak, S. 1993. Linking GIS and spatial data analysis in practice. Geographical Systems 1 (1), 3–23.

Bailey, T. and Gatrell, A. 1995. Interactive spatial data analysis. Longman, Harlow, 413 pp.

Bivand, R. S. 1996. Scripting and toolbox approaches to spatial analysis in a GIS context. In Fischer, M. M., Scholten, H., Unwin D. (Eds) Spatial analytical perspectives on GIS in the environmental and socio-economic sciences. Taylor & Francis, London, pp. 39–52.

Bivand, R. S., 1999. Integrating GRASS 5.0 and R: GIS and modern statistics for data analysis. In: Proceedings 7th Scandinavian Research Conference on Geographical Information Science, Aalborg, Denmark, pp. 111–127.

Burrough, P. A., McDonnell, R. A. 1998. Principles of geographical information systems. Oxford University Press, Oxford, 333 pp.

Byars, B., Clamons, S. 1998. GRASS is back! GIS World 10 (2), 60–63.

Ihaka, R., Gentleman, R. 1996. R: a language for data analysis and graphics. Journal of Computational and Graphical Statistics, 5 (3), 299–314.

Jacoby, W. G. 1997. Statistical graphics for univariate and bivariate data. Sage Publications, Thousand Oaks, 97 pp.

Jones, K. H. 1998. A comparison of algorithms used to compute hill slope as a property of the DEM. Computers & Geosciences 24 (4), 315–323.

Luo, W., 1998. Hypsometric analysis with a geographic information system. Computers & Geosciences 24 (8), 815–821.

Mitasova, H., Mitas, L., Brown, W.M., Gerdes, D.P., Kosinovsky, I., Baker, T., 1995, Modeling spatially and temporally distributed phenomena: New methods and tools for GRASS GIS. International Journal of Geographic Information Systems, 9 (4), 433-46.

Pike, R. J., Wilson, S. E. 1971. Elevation-relief ratio, hypsometric integral, and geomorphic area-altitude analysis. Geological Society of America Bulletin 82 (4), 1079–1084.

Venables, W. N., Ripley, B. D. 1997. Modern applied statistics with S-PLUS. Springer, New York, 548 pp.

Zevenbergen, L. W., Thorne, C. R. 1987. Quantitative analysis of land surface topography. Earth Surface Processes and Landforms, 12 (1), 47–56.

Internet references

[1] Assist project data set.
http://www.geog.le.ac.uk/assist/grass/data/leics/leics.tar.Z

[2] Comprehensive R Archive Network. http://cran.r-project.org/

[3] GRASS home page. http://www.baylor.edu/~grass

[4] Source of Kosovo elevation data.
http://edcwww.cr.usgs.gov/landdaac/gtopo30/gtopo30.html