DSC 2001 Proceedings of the 2nd International Workshop on Distributed Statistical Computing
March 15-17, Vienna, Austria
http://www.ci.tuwien.ac.at/Conferences/DSC-2001
K. Hornik & F. Leisch (eds.) *ISSN 1609-395X*

R and geographical information systems, especially GRASS

Roger Bivand[*]

Abstract

The paper starts by sketching the key modes of spatial data analysis (point pattern, continuous surface, areal/lattice), and how they fit into legacy GIS data models. Based on development work with the GRASS release 5 team (GRASS 5 is GPL), status is reported for using R in the analysis of sites (point) data and for raster data, as well as using R as an intermediate analytical environment for interpolating from sites data to discretised continuous surfaces. Indications are given for broader interfacing using the GDAL library for GIS raster data and remote sensing data for other GIS than GRASS. So far, there are few vector data results, although again, interesting open source possibilities seem to be emerging.

1 Spatial data analysis

Spatial data analysis ranges from the visualization and exploration of spatial data, through spatial statistics to spatial econometrics. The techniques involved are intended to explore for and demonstrate the presence of dependence between observations in space. Typically, observations are classified into three broad types: fields or surfaces with values at least theoretically observable over the whole study area, as in geostatistics, point patterns representing the occurrence of a phenomenon, such as reported cases in epidemiology, and finally lattice observations, where attribute values adhere to a tesselation of the study area. This last form has much in common with time series studies, and shares a number of key testing techniques with econometrics.

[*] Economic Geography Section, Department of Economics, Norwegian School of Economics and Business Administration, Bergen, Norway

Since observations of spatial data are as unlikely to be independent as observations on time series, it is perhaps surprising that not more use has been made of this source of information. With an adequate choice of explanatory variables, this spatial dependence may be readily drawn into a model. The literature on spatial statistics is substantial (see for example [7, 8, 21, 24, 15, 1, 16], and more recently [9, 3, 13]).

Attention has also been given to the potentials for integrating Geographical Information Systems (GIS) with modelling and analysis tools, for instance in [14, 10, 19], and [20]. Some of the analytical tools are available in the SPLUS spatial statistics module [17], and links between the ARC-INFO GIS and the ArcView desktop GIS are available for SPLUS. Special numbers of journals have been devoted to exploratory spatial data analisis (*The Statistician*, 1998, number 3), and computing environments for spatial data analysis (*Journal of Geographical Systems*, 2000, number 3), among others.

Within R, a range of packages are designed for use with spatial data of various kinds, or can be used for such data as well as for non-spatial data. A survey dating from 1998 can be found in [5], written with Albrecht Gebhardt of the University of Klagenfurt, one of the people who has contributed most to repackaging and releasing tools for spatial data analysis. For the analysis of spatially continuous data, he is maintaining the `sgeostat` package, and is working on compiled functions to increase its speed to that of the functions in `spatial`; he also maintains `akima`, `ash`, and `tripack`. A promising package in development is Paulo Ribeiro and Peter Diggle's `geoR`[1].

For point pattern data, `spatial` and `splancs` are available, and other tools are under development[2]. So far no code has been posted for area data analysis, where the clear benchmark is Luc Anselin's GAUSS-based closed source SpaceStat[3]. Some mapping functionality is available from work by Ray Brownrigg[4], discussed in [6].

A starting point for spatial data analysis is that positional information for the observations matters. For point pattern analysis, all we have is where the points are located, given most often as coordinates in two dimensions. The only realistic analogy in data analysis is with time series, in which either local position relative to an arbitrary origin, or absolute position in for example UTC is given. However, spatial data vary a great deal both in the ways in which their position attributes are recorded, and in the adequacy of documentation of how position has been determined. This applies both to secondary data and to data capture by remote sensing, Global Positioning System input, or data capture from analogue maps. This also constitutes a specific difference from the analysis say of medical imagery, which only requires a local coordinate system. Knowledge about the spatial reference system is needed to establish the positional coordinates' units of measurement, obviously needed for calculating distances between observations, and/or for describing the network topology of their relative positions.

[1] http://www.math.lancs.ac.uk/~ribeiro/geoS.html

[2] Adrian Baddeley and Rolf Turner have made a beta of `spatstat` available on http://www.maths.uwa.edu.au/~adrian/spatstat.html

[3] http://www.spacestat.com

[4] ftp://ftp.mcs.vuw.ac.nz/pub/statistics/map

2 GIS data models

GIS are designed to be good at handling spatial reference systems, and to allow data of different original forms to be converted to a common reference system. More recently, a trend has started to emerge in which objects are only stored with position given in geographical coordinates (degrees easting and northing, longitude and latitude), and projected on-the-fly for requested displays. Associated with this is the computation of topologies on-the-fly; both of these operations have traditionally been seen as compute-intensive, but modern hardware can make the pre-processing of imported data unnecessary. The data models described here are the legacy ones, more typical of the usual workings of GIS software.

Most GIS have developed from paper map analogies, rather than from database entity-relationship approaches or from computer aided design. Because the systems were chiefly designed for surveyors and others maintaining and renewing existing paper archives of maps of great operational value, no major break with tradition occurred. One of the consequences of this is that GIS only rarely handle three dimensional data adequately, another that time is also an afterthought, typically with different map layers for different points in time. Formal presentations of geometries for spatial data may be found in [18] and [25].

Two major families of GIS may still be found, despite the fact that they are both specific realisations of more general models for handling spatial data. One family fits the surveying paradigm better, by focussing on the precise position of points, and objects derived from them, such as lines and polygons — this is the vector GIS family. The other family — raster GIS — presupposes a regular tesselation of the area of interest into units, typically squares or rectangles, at a chosen resolution. A hybrid subtype uses quadtrees to vary the resolution of the tesselation across the area to follow data variation more adequately. Property boundaries or utilities inventory data are best handled in vector systems, while many environmental assays, especially those involving remote sensing, suit raster GIS better. In fact, the two families can be equivalent at a chosen (fine) resolution in theory, although the raster data layers might become rather bulky. In practice, the demarcation between the families is also removed by the widespread use of scanned images (such as legacy paper maps) — which are store in an image format such as TIFF, for “heads-up” on-screen digitising of vector points.

Raster family GIS see the world as a collection of map layers, not necessarily at the same resolution, and not necessarily with exactly the same edges, but most often in the same projection, to ease on-the-fly resampling for differening resolution. They support a rich variety of operations, sometimes known as map algebra, and easily operationalise the overlay function, moving windows, and filtering (for a full description of techniques, see [23]). For instance, given a digital elevation model (DEM) map layer, slope and aspect layers may be generated by simple filtering (with the exception of raster cells at the map edges). Most raster GIS and remote sensing software are partly integrated with vector data, from just being able to plot vector points and lines on a raster display, to moving data from one domain to another without too many compromises. A specific issue is that standard raster layers are “crisp” at the tesselation boundaries, while vector objects are “crisp” at their own boundaries. Neither are good at representing "non-crisp" categories or classifications, though work is progressing using fuzzy classification (see [11], [12]).

Vector family GIS typically store their points and derived geometries with tables of attribute values expressing what they represent, for instance whether a point is a street lamp or a church tower, or whether a line is a drainage channel centre line or a contour. They will usually be designed to accommodate the existence of lakes on islands in lakes, both in the internal data structure and in display functions (filling contained polygons in given order). They typically ingest data both by storing the data themselves, and also for line and polygon data by storing adequate representations of their topologies, permiting queries such as: which polygons with attribute value x neighbour this polygon? Vector GIS are good at data retrieval from spatial queries, because of the attribute tables pointed at by object pointers, recording all the attributes of an object, and because of the pre-stored topologies, although it may now be feasible to grab topologies on-the-fly.

One general matter to take on to discussions of interfacing GIS and statistics and other data analysis software is how to ensure the integrity of the positional data and its metadata — projection, ellipsoid, datum, and where relevant topology — in relation to the attribute data. Should each data item be structured with everything on board, carrying copies on under replication? Should one perhaps assume a fixed window and resolution for raster GIS interfaces? Are there mechanisms that can be followed and maybe implemented to help in this?

Among the useful sources of information are OpenGIS, which is an industry standardisation alliance, and in the open source arena, a number of the projects coordinated through the www.remotesensing.org website. In particular GDAL gives access to multiple raster formats, in many cases permiting inter-system conversion without loss of georeferencing. The same site hosts the maintained version of PROJ.4, a library and helper programs for projection, which could perhaps replace the current code in the `maps` package mentioned above. The person responsible for both of these is Frank Warnerdam, who has also initiated discussion on design of open source vector databases[5].

3 The R-GRASS interface

GRASS, the Geographic Resources Analysis Support System, was initially developed within the US Army, among other things for monitoring the environmental effects of military exercises; it is a raster GIS. It was written in C as a suite of programs run from the Unix shell prompt, from within a shell process in which environment variables are defined on startup to specify the current "LOCATION", the study area, and the user's "MAPSET". It tries to be careful to prevent users overwriting each other's files by creating a user hierarchy under each LOCATION. Each such MAPSET will have either a default or user-defined "WINDOW", the extent and resolution to be used for new map layers created by the user. GRASS

[5] PostgreSQL does include some spatial data support, but opinion is divided as to its applicability, in particular whether there is enough topology

also supports sites data, point locations with one or more attributes, and vector data for analysis with raster layers.

GRASS was US Government style open source (public domain), and was dropped by the engineering research laboratory in 1996. It was transferred at that point to Baylor University, in an unusually difficult phase in the development of the software. Up to and including release 4 (currently the GPL'ed 4.3), GRASS did not handle NA well, using integer zero, nor did it use floating point, just integer. During the early 1990's, many people had been prototyping floating point and NULL-value modifications, but they did not make it through to production code. The release 4.x series is now closed apart from patches to fix bugs, and GRASS 5.0 is close to being released as stable — it is now at beta 11.2. GRASS 5.0 development is coordinated by Markus Neteler from the University of Hannover[6]. GRASS 5.0 has full floating point support, NULL-value support, and a range of improvements which now also make compilation under Cygwin feasible; it is released under GPL. The work reported here is for interfacing GRASS 5.0 and R. The current package, GRASS_0.1-6 in contrib/Devel on CRAN, is for the new memory model from version 1.2.0 and more recent, an older version for R versions before 1.2.0 is also available on CRAN.

The first version of the interface as presented in [4] used ASCII temporary files to move data between R and GRASS for sites — using `read.table` and `write.table()` on the R side, and `scan()` to import raster layers as vectors, initially with metadata attached to each raster layer, later with metadata encapsuled separately but also transferred through ASCII, risking loosing precision on the way. The chief breakthrough at this stage was the very simple realisation that R could be run from withing the GRASS shell, inheriting the settings of all of the environment variables needed by GRASS programs. These could in turn be run from within R through `system()`, permitting the interface to be encapsulated in a library of R functions. The nesting of shells used is illustrated in Fig. 1. It may be worth not-

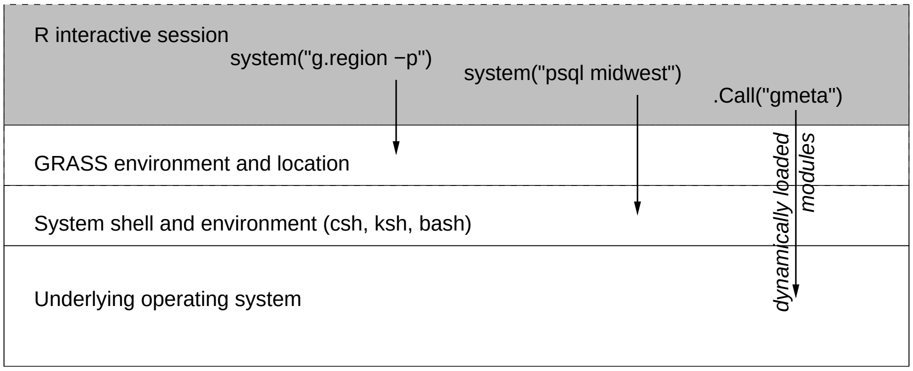
R interactive session
system("g.region −p")
system("psql midwest")
.Call("gmeta")
GRASS environment and location
System shell and environment (csh, ksh, bash)
Underlying operating system
dynamically loaded modules

Figure 1: Nesting of shell levels in the R/GRASS interface

[6] http://www.geog.uni-hannover.de/grass/index2.html

ing that a number of legacy GIS had similar Unix-shell interfaces (e.g. Genamap), but most used their own character-based interactive solutions, and now many are simply GUI with some underlying scripting capability. GRASS is in this sense one of the few GIS that remains modular.

Since raster layers can be large, and because work was begun before both the `Rstreams` library was released and before connections became part of R base (see [22]), getting raster layers from the GRASS database and putting them back was converted to compiled C code. This also avoids using temporary binary files, and tries to ensure that the current location window and resolution are used. Metadata may also be extracted directly. Compiled code was also introduced for metadata helper functions, essentially mapping the sequence of imported raster cell vector elements back onto their coordinates. In addition it became clear that the world of raster GIS begins upper left, not lower left as in R: raster row 0 runs along the northern edge of the study area, not the southern. Consequently, `image()` needs the row order of the data reversed, which is now handled through compiled code.

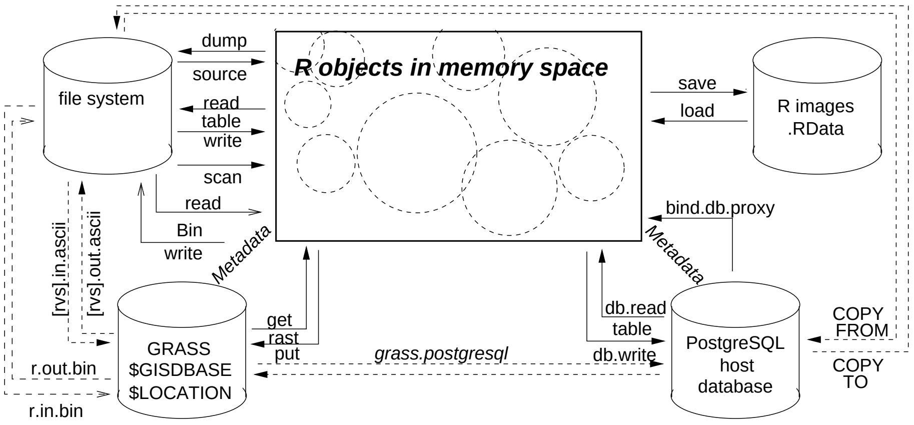

Figure 2: Transfer of data between R, GRASS, a database (here PostgreSQL) and the underlying file system

Fig. 2 shows the routes that data transfer can take; the current release supports both direct binary transfer and transfer through temporary ASCII files. Work has also been done using the `RPgSQL` R-PostrgeSQL interface with the GRASS-PostgreSQL interface, and this option will be returned to when the GRASS vector format stabilises. For raster data, we found that the direct link — reading directly from GRASS using the GRASS `libgis.a` — was substantially more efficient. With the new R memory model, the issue of estimating memory usage before continuing analysis has been resolved, so that using a database for subsetting is not necessarily an advantage, since it can be done in R. It also turned out to be possible (though probably not good practice) to transfer labeled category map layers from GRASS to R as factors without importing them as character vectors and converting in R, by simply copying the GRASS category codes (renumbered to start from 1 without gaps) into the vector as integers, and giving the resulting object a character vector of levels taken straight from GRASS, and class factor. The example shown in Fig. 3 is only a small example from a 12km square area of N.W. Leicestershire at a resolution of 50m, giving 57600 raster cells, showing the relative distributions of elevation by land cover class, and was constructed with the following R code:

```
> library(GRASS)
Running in /usr/local/grass-5.0b/data/leics/rsb
> G <- gmeta()
> summary(G)
Data from GRASS 5.0 LOCATION leics with 240 columns and 240 rows;
UTM, zone: 0
The west-east range is: 444000, 456000, and the south-north: 310000, 322000;
West-east cell sizes are 50 units, and south-north 50 units.
> system("g.list rast")
----------------------------------------------
raster files available in mapset rsb:
aspect    dslope    landcov4  slope1  slope3
avheight  landcov1  slope     slope2  tmp

raster files available in mapset PERMANENT:
contours  image    plant  rail   segment  spillage  urban
crash     landcov  popln  roads  source   topo      water

----------------------------------------------
> leics <- rast.get(G, rlist=c("topo", "landcov"), catlabels=c(F, T))
> str(leics)
List of 2
 $ topo     : num [1:57600] 80 80 80 80 80 80 80 80 78 76 ...
 $ landcov.f: Factor w/ 9 levels "Background","In..",..: 6 6 6 6 6 6 6 6 6 6 ...
> tlc <- table(leics$landcov)
> tlc

 Background    Industry Residential      Quarry    Woodland      Arable
          0        1671        6632         555        1749       22913
    Pasture       Scrub       Water
      16816        6558         706
> boxplot(leics$topo ~ leics$landcov, width=tlc[2:9])
```

Before posting to CRAN, the interface was being downloaded several times a day by GRASS users; now the count is a little lower. Little user reaction has been seen since early teething troubles were sorted out; the problems isolated and removed were concerned with precision in the transferred ASCII data. The GRASS developers are supporting the interface, and have re-arranged the installed directory structure to suit the compilation of the package C code for a standard location for GRASS header files and `libgis.h` location. The functions in this main library have also been revised to isolate and remove all `exit()` calls not going through the GRASS error handler, so that triggering an error in an internal GRASS library call does not exit R; this work is linked to Frank Warnerdam's `libgrass` (available from www.remotesensing.org), although so far it has seemed easier to expect users needing the interface to have GRASS itself installed. The GRASS fatal error function hands on the string returned to the internal R `error()` function, as shown

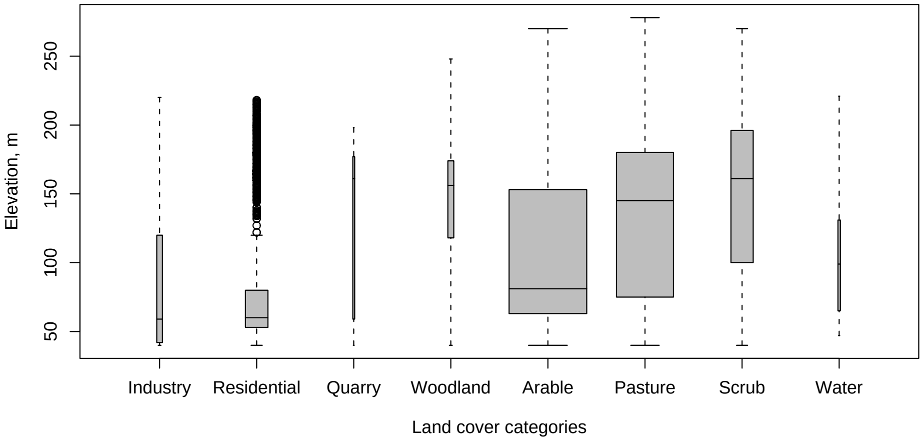

Figure 3: Boxplots of N.W. Leicestershire elevation by land cover type

by this code for the interface initialiser, included at the beginning of each interface function:

```
void R_G_init(char *name) {
    G_set_error_routine(R_handler);
    G_sleep_on_error(0);
    G_gisinit(name);
}

int R_handler(char *message, int fatal) {
    if(fatal == 1) error(message);
        else warning(message);
}
```

4 Work in progress

The GRASS vector database structure is under revision, in particular its use of pointers to tables of attribute data. These data tables would most usefully be represented in R as data frames, with a privileged identifier indexing the site, line or polygon with which the chosen row should be linked. Present GRASS vector data also stores a topology for line and polygon structures, rather like that implemented in the S `map.builder`. In both cases, labelling polygons in order to link them to identifiers happens after the topology is built, by point-in-polygon. Neither seem to be good at handling islands in lakes — the underlying data structures do not seem to admit multiple polygons for a single pointer to attribute data. I am aware that Nicholas Lewin-Koh was doing some work on reading ArcView shape files into R, but am not sure of current status — given his work on Ggobi, there are possibilities that some of the visualisation will happen there rather than in R, because linked choropleth maps and data plots are very sought after.

A point of criticism raised by Luc Anselin [2] is that there are no functions for R for area data analysis, and in particular for constructing spatial weights matrices. Some functions have been written, both for the data analysis, and for creating and lagging weights matrices, but these have not yet been packaged and documented, although some early results are given in [5]. Weights matrix construction presently either reads in sparse neighbour tables in GAL format like the SPLUS module, or uses `tripack` to triangulate neighbours from points, or finds nearest or near neighbours, but more work remains. In addition, completion is linked to a design decision on whether to implement a vector data model with topology — then the neighbours are available immediately — or without, in which case they have to be generated by spatial searching to match polygon boundaries.

The question of how to handle metadata capsules attached to spatial data is still very much open. One possibility for the future may be based on the work of OpenGIS, who have published an XML-based proposal, GML, in OGC RFC 11[7]. GML differentiates between geometry collections, feature collections, and spatial reference systems at the DTD (Document Type Definition) level. The underlying logic is that an XML object should contain all of its metadata - that it is *self-describing*, or pointers to DTD links providing access to metadata, so that applications exchanging XML streams or files can rely on knowing what the arriving or departing data are. XML is one of the active areas in the Omega project, and both DTD's and XML files based on them can be read into S-like languages, and data can be written to an XML file using the specified DTD.

Cursory use of the new base R connections mechanism shows that it can be most useful in widening GIS-R interfacing. PCRaster files (CSF format) have been read, and since they are the prefered format for `gstat`[8], reading and writing such raster files would be a way of using it as additional loosely linked geostatistics functionality. Fortunately, CSF is very well documented and open source. The same cannot be said of MF-Works, software tracing back to Toplin's original MAP, but now closed source. Raw binary export files can be read and files written with `writeBin` can be imported back, but with reversed logic applied to the byte-swapping button in the software. That taken into account, megacell raster map layers can be read into R, analysed, and displayed. In many cases, simple graphical tools give great added value, be it boxplots by category or empirical cumulative distribution plots. Given these capabilities, other interfaces may be built between R and closed-source GIS, for which no libraries are available to link against.

It has been very encouraging to see time series brought into the R release itself, as well as being represented among the contributed packages. There is still a lot to do before just a mapping function could reach the same status, I suppose, and it is not yet clear that this is really something everyone needs. However, as a geographer, I do feel a certain need to be able to let my students plot maps of data from the standard data sets then they do actually come from specific places. Maybe even

[7] http://www.opengis.org/techno/rfc11info.htm

[8] http://www.gstat.org

the default method for plotting data with positional attributes might be a map — if there is no spatial story nothing is lost, but at least the possibility would have been examined.

References

[1] Luc Anselin. *Spatial econometrics: methods and models.* Kluwer, Dordrecht, 1988.

[2] Luc Anselin. Computing environments for spatial data analysis. *Journal of Geographical Systems*, 2:201–220, 2000.

[3] Trevor C. Bailey and Antony C. Gatrell. *Interactive spatial data analysis.* Longman, Harlow, 1995.

[4] Roger Bivand. Using the R statistical data analysis language on GRASS 5.0 GIS database files. *Computers & Geosciences*, 26:1043–1052, 2000.

[5] Roger Bivand and Albrecht Gebhardt. Implementing functions for spatial statistical analysis using the R language. *Journal of Geographical Systems*, 2:307–317, 2000.

[6] Roger Bivand and Markus Neteler. Open source geocomputation: using the R data analysis language integrated with GRASS GIS and PostgreSQL data base systems. http://reclus.nhh.no/gc00/gc009.htm, 2000.

[7] Andrew D. Cliff and J. Keith Ord. *Spatial autocorrelation.* Pion, London, 1973.

[8] Andrew D. Cliff and J. Keith Ord. *Spatial processes — models and applications.* Pion, London, 1981.

[9] N. A. C. Cressie. *Statistics for spatial data.* Wiley, New York, 1993.

[10] Manfred M. Fischer, Henk J. Scholten, and David Unwin, editors. *Spatial analytical perspectives on GIS.* Taylor & Francis, London, 1996.

[11] Peter F. Fisher. The pixel: a snare and a delusion. *International Journal of Remote Sensing*, 18:679–685, 1997.

[12] Peter F. Fisher. Sorities paradox and vague geographies. *Fuzzy Sets and Systems*, 113:7–18, 2000.

[13] A. Stewart Fotheringham, Chris Brunsdon, and Martin Charlton. *Quantitative geography: perspectives on spatial data analysis.* Sage, London, 2000.

[14] Stewart Fotheringham and Peter Rogerson, editors. *Spatial analysis and GIS.* Taylor & Francis, London, 1994.

[15] Daniel A. Griffith. *Advanced spatial statistics.* Kluwer, Dordrecht, 1988.

[16] Robert P. Haining. *Spatial data analysis in the social and environmental sciences.* Cambridge University Press, Cambridge, 1990.

[17] S. P. Kaluzny, S. C. Vega, T. P. Cardoso, and A. A. Shelly. *S+SPATIALSTATS users manual version 1.0.* Seattle, 1996.

[18] Robert Laurini and Derek Thompson. *Fundamentals of spatial information systems.* Academic Press, London, 1992.

[19] Paul Longley and Michael Batty, editors. *Spatial analysis: modelling in a GIS environment.* Geoinformation International, Cambridge, 1996.

[20] Paul Longley, Sue M. Brooks, Rachael McDonnell, and Bill Macmillan, editors. *Geocomputation: a primer.* Wiley, Chichester, 1998.

[21] Brian D. Ripley. *Spatial statistics.* Wiley, New York, 1981.

[22] Brian D. Ripley. Connections. *R News*, 1:16–17, 2001.

[23] C. Dana Tomlin. *Geographic information systems and cartographic modelling.* Prentice-Hall, Englewood Cliffs, NJ, 1990.

[24] G. J. G. Upton and B. Fingleton. *Spatial data analysis by example: point pattern and quantitative data.* Wiley, London, 1985.

[25] Michael F. Worboys. *GIS — a computing perspective.* Taylor & Francis, London, 1995.