DSC 2003 Working Papers
(Draft Versions)
http://www.ci.tuwien.ac.at/Conferences/DSC-2003/

RArcInfo: using G.I.S. data with R

Virgilio Gómez Rubio

Abstract

Geographical Information Systems (G.I.S.) have become powerful tools for integration, visualisation and analysis of spatial data, also providing the capability of integrating data from different sources. However, the number of statistical methods implemented in most GIS is not comparable to those we can find within R.

RArcInfo is written with the aim of importing data from two of the most used formats in the G.I.S. community: Arc/Info V 7.x binary coverages and E00 files, which are supported by nearly all the G.I.S. software available nowadays.

Once the data are imported into R, the user can use several packages to perform spatial analysis on the data, access to databases to integrate new data and other interesting possibilities, which increase the spectrum of statistical tools available to mine data.

1 Introduction

Geographical Information Systems support a wide variety of formats to represent spatial data but a few of them have been imposed as standards *de facto* to represent vector data. ESRI shapefiles, ESRI E00 files and Arc/Info V 7.x binary coverages are a few examples.

Vector data formats encourage the representation of spatial objects, such as points, arcs, polygons or areas, and attributes related to them, such as topology or measurements of different variables (altitude, population, etc.).

On the other hand, *raster data formats* represent the space by using a continuous grid (homogeneous or not), in which every cell has associated one or several values (elevation, temperature, etc.). Usually this kind of data are managed using images, in which pixels represent values or categories of a given variable.

While there has been some effort in the R community to provide support for raster data, be it through image manipulation or via the R/GRASS interface [1, 5], there is nearly no support for vector data. Some data sets include vector data as sets of polygons, but undoubtly the most realistic support for vector data is found in the package *spdep*, developed by Roger Bivand and others.

This package is mostly devoted to spatial dependence analysis, but a simple and efficient format is used to store vector data coverages (mostly coverages attribute tables and polygon definitions). Furthermore, it provides several interesting examples showing how to analyse spatial data and create maps with the results.

Nicholas Lewin-Koh is developing a promising package called *maptools* to manage ESRI shapefiles within R, but it seems that the development has stopped and the source needs some modifications to be compiled properly. Another drawback is that this format doesn't provide topological information, only point, arc or polygon definitions and related attributes.

Barry Rowlingson [10] has started the development of a general package, called Rmap, to manage maps in a general way. It supports a wide variety of vector data formats by using the OGR library. OGR provides a unified framework to access to many different vector data formats. Rmap also allows map projections through the PROJ4 library.

Arc/Info V 7.x binary coverages and E00 files are highly structured formats for representing vector data, since they provide a full framework for storing points, arcs, polygons, topology and related information.

It is worth mentioning that usually information is split into several *coverages* or layers, each one devoted to a single subject, and that some of them may only contain points or arcs. For example, for a country, it is possible to have different coverages for cities (point data), roads (arc data) and counties (polygon data).

RArcInfo not only imports the data, but also creates a suitable framework of objects (mostly data frames and lists) to keep the internal representation of the data. Besides, it provides a few functions to create maps, reduce the number of arcs per polygon and calculate neighbouring regions.

2 Vector data representation in E00 files and Arc/Info V 7.x binary coverages

2.1 General description

Point is the basic element, which is defined by giving its position in some coordinate system (Cartesian, polar, longitude/latitude, U.T.M., etc.). In 3D G.I.S. altitude may be added as a new dimension. Usually the points are also called *nodes* or *vertices*, since they fall at the arcs and polygons joints.

Once nodes are set, arcs are defined, by giving an ordered sequence of points. Arcs are usually open arcs, i. e., the initial and final nodes differ. Polygons are also defined in such a way, and they can be thought as sets of arcs which build one or several closed arcs. A trivial example showing points, arcs and polygons appears in Figure 1.

At this level topology arises. Topology can be defined as the way points, arcs and, specially, polygons are related one another. It is usually described by stating the polygons an arc separates, neighbouring polygons, etc.

Some vector data formats (for example, ESRI shapefiles) do not provide a topology description, and they only define points, arcs or polygons (depending on the type of the coverage) and related attributes. Of course, topology can be derived

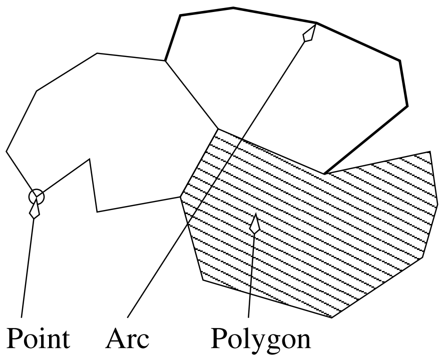


Figure 1: Main objects in vector data representation.

by splitting polygons into arcs and looking for the relationships between arcs and polygons.

An easy way to specify the topology is, for every arc, indicate the polygons that fall to the left and to the right, where left and right are defined depending on the order of the nodes in the arc.

Points, arcs and polygons are usually refered by an internal identifier, which is an easy way to store the data and allows linking with the attribute tables and external data.

Topology information is stored in a separated table, together with arcs and polygons definitions. Besides, it is common to have two attribute tables, one for the arcs and other for the polygons. More tables containing other kind of data (altitude, temperature, etc.) may also appear in a coverage, although this data could also be in the attributes table.

2.2 E00 files and binary coverages

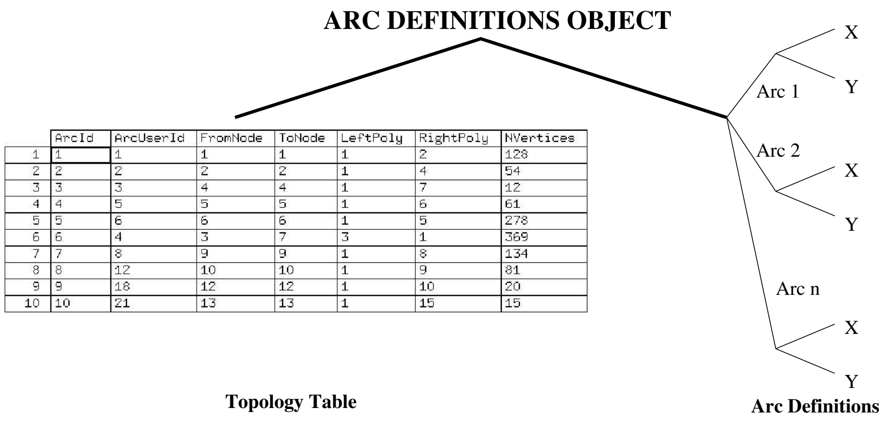
E00 files and Arc/Info binary coverages share the same data structures. E00 files are ASCII files and they only store a single coverage, while Arc/Info binary coverages can store more than one coverage which are split into different files and directories.

In a single coverage, arcs are defined by giving a sequence of points, for which just the initial and final node are given an identifier (in the arc topology table). Polygons are described by a sequence of arcs, specified by their internal identifiers. If the arc is reversed, then its identifier will be negative.

Fields found in a topology table related to arcs are *arc identifier*, *arc user identifier*, *initial* and *final nodes*, *left* and *right polygons*, and *number of vertices*. For polygons, the topology table is made of *polygon identifier*, *maxima* and *minima coordinates*, and *number of arcs*.

Figure 2 is an example of how arc definitions are imported into R. Polygon data is stored in a similar structure, but instead of X and Y coordinates, there are three other fields: *arcs identifier*, *initial node* and *adjacent polygon*.

Other issue worth mentioning are centroids, which represent the *centre* of the polygons and they are very useful when just a point is needed to represent the entire polygon. They are stored in another table, whose fields are *polygon identifier*, *X* and *Y coordinates*, and *number of point labels* related to it (which are used when

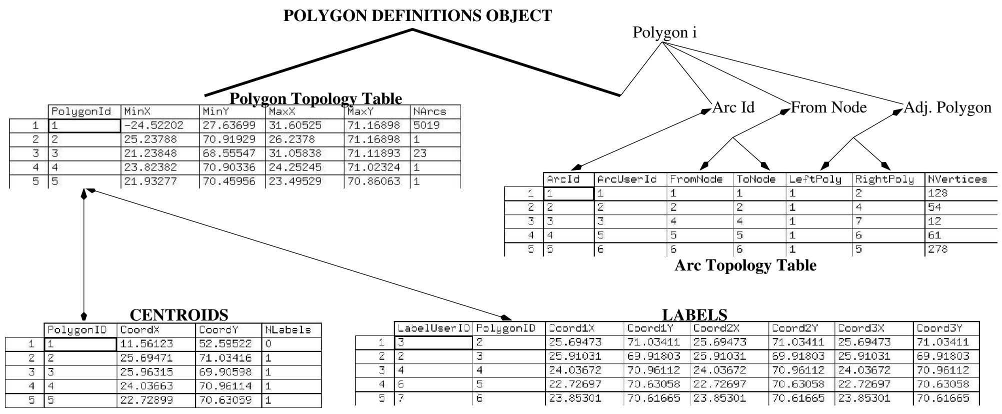


	ArcId	ArcUserId	FromNode	ToNode	LeftPoly	RightPoly	NVertices
1	1	1	1	1	1	2	128
2	2	2	2	2	1	4	54
3	3	3	4	4	1	7	12
4	4	5	5	5	1	6	61
5	5	6	6	6	1	5	278
6	6	4	3	7	3	1	369
7	7	8	9	9	1	8	134
8	8	12	10	10	1	9	81
9	9	18	12	12	1	10	20
10	10	21	13	13	1	15	15

Figure 2: Structure of an arc definitions (from file *arc.adf*) imported in R. Lines represent lists, while X and Y are both vectors of coordinates.

plotting text related to the polygons).

Figure 3 shows how different table data are related one another. These relationships can be used when selecting arcs or polygons related to a given variable stored in the attribute tables.



Polygon Topology Table

	PolygonId	MinX	MinY	MaxX	MaxY	NArcs
1	1	-24.52202	27.63699	31.60525	71.16898	5019
2	2	25.23788	70.91929	26.2378	71.16898	1
3	3	21.23848	68.55547	31.05838	71.11893	23
4	4	23.82382	70.90336	24.25245	71.02324	1
5	5	21.93277	70.45956	23.49529	70.86063	1

Arc Topology Table

	ArcId	ArcUserId	FromNode	ToNode	LeftPoly	RightPoly	NVertices
1	1	1	1	1	1	2	128
2	2	2	2	2	1	4	54
3	3	3	4	4	1	7	12
4	4	5	5	5	1	6	61
5	5	6	6	6	1	5	278

CENTROIDS

	PolygonID	CoordX	CoordY	NLabels
1	1	11.56123	52.59522	0
2	2	25.69471	71.03416	1
3	3	25.96315	69.90598	1
4	4	24.03663	70.96114	1
5	5	22.72899	70.63059	1

LABELS

	LabelUserID	PolygonID	Coord1X	Coord1Y	Coord2X	Coord2Y	Coord3X	Coord3Y
1	3	2	25.69473	71.03411	25.69473	71.03411	25.69473	71.03411
2	2	3	25.91031	69.91803	25.91031	69.91803	25.91031	69.91803
3	4	4	24.03672	70.96112	24.03672	70.96112	24.03672	70.96112
4	6	5	22.72697	70.63058	22.72697	70.63058	22.72697	70.63058
5	7	6	23.85301	70.61665	23.85301	70.61665	23.85301	70.61665

Figure 3: Relationships between arcs, polygons, centroids and labels. Notice that values of *From Node* and *Adjacent polygon* depend on the way we move along the arcs.

The degree of accuracy of the data is set by precision, which can be simple and double (more accurate), depending on the actual number of digits used to store numerical values. RArcInfo can import coverages in both precisions.

Closely related to this subject is tolerance, which is the minimum threshold (or distance) two point must be separated to be considerate as different. This means that previous points that were closer than the distance specified in the original data were joined into a single point. Tolerance table has three fields: *type* (as shown in [7]), *status* (whether it is active or not, i.e., it is in fact satisfied by the data or not) and *value* (minimum threshold).

TXT annotations can also be imported in a similar way than arcs and they are imported into a similar object. Complete information can be found in the package documentation.

2.3 Peculiarities of binary coverages

Binary coverages usually contain more than a single coverage. Each one is stored in a separated directory, but all the information about attribute tables is under a common directory called *info*, although the actual data may be under a coverage directory (i.e., what is called an *external table*).

FILE	FUNCTION	DESCRIPTION
aat.adf	get.tabledata	Arc attribute table.
arc.adf	get.arcdata	Arc coordinates and topology.
bnd.adf	get.bnddata	Coverage boundary (maxima and minima coordinates).
cnt.adf	get.cntdata	Polygon centroid table.
lab.adf	get.labdata	Label point coordinates and topology.
pal.adf	get.paldata	Polygon definitions and topology.
pat.adf	get.tabledata	Polygon or point attribute table.
tic.adf	get.tabledata	Tic coordinates.
tol.adf	get.toldata	Coverage processing tolerances (single precision coverage).
par.adf	get.toldata	Coverage processing tolerances (double precision coverage).

Figure 4: Files usually found in a binary coverage directory.

Under a coverage directory we can find different files. A complete description can be found in [7], but in Figure 4 the most common are shown. Italic has been used to mark those files that contain tables. The function supplied by RArcInfo to import the file is also showed in the table.

There is another function called *read.coverage* that tries to *automagically* import all the features in a coverage, but it may fail if some of the main files is missing.

This may happen, for example, when the coverage has only points or arcs. In these cases, file *pal.adf* will not appear. On the other hand, if the coverage stores several polygon layers based on the same arc structure they will be stored in files ending by *.pal*. All of them represent a different polygon definition. This happens, for example, when representing different administrative regions at different levels (municipalities, counts, countries,), since they can be built with the same set of arcs.

2.4 Peculiarities of E00 files

All the data mentioned before is stored in an ASCII file, whose extension uses to be E00. RArcInfo can't read directly E00 files (although it is supported in the AVCLIB), but they can be converted into a binary coverage with the function *e00toavc* and later imported.

Complete information about the structure format can be found in [8].

3 Overview of RArcInfo

RArcInfo has the capability of importing all the data mentioned before, maintaining nearly the same logical structure. Tables are stored as data frames, while all the other information is stored using lists. A brief description of the package capabilities and functions can be found in the documentation. Just load the package and write `help(RArcInfo)`.

For all the data not stored in tables there is a function of the form *get.XXXdata*, where *XXX* is the file type (arc, pal, etc.), to load data from different types of files.

Arcs and polygons definitions are stored in a list with two main components, as shown in Figures 2 and 3. The first one is a data frame containing topological information, while the second one is a list containing points (or arcs) needed to define arcs or polygons.

Boundary limits (in *bnd.adf*), T.I.C. points (in *tic.adf*), attributes and other relevant tables can be imported with the command `get.tabledata`, although for boundaries a separated function called *get.bnddata* is also provided.

RArcInfo provides three functions to plot arcs and polygons: *plotarc*, *plotpal* and *plotpoly*. The first two functions just plot a given set of arcs or polygons, while the third one allows the user to fill polygons with colours, which is quite useful to create thematic maps. Additional plotting arguments can be passed as well.

As it was mentioned before, function *read.coverage* can be used to read a whole coverage. This should be the first option and, if it fails, try to load files one by one. It returns a list of objects returned by different functions (get.arcdata, get.paldata, etc.).

4 RArcInfo and other spatial packages

With the data imported into R, the possibilities of analysis are highly incremented, ranging from basic statistics to more developed spatial analysis, since R provides a number of packages on this subject.

Data may be analysed with the usual techniques, but more advanced spatial analysis can be performed. A revision of these packages is available from several sources, like journals[1, 4], proceedings of the DSC 2001 [3] or R newsletter [9, 11, 2, 6].

A pretty interesting report of how to integrate several tools (R, GRASS and PostgreSQL) to perform analysis of spatial data has been written by Roger Bivand and Markus Neteler [5].

5 Examples

The data used in these examples are the 'Administrative Regions/Boundaries of Europe', which are publicly available from UNEP-GRID at Geneva [1] as two E00 files.

The next lines show an introductory session in which these data are imported into R . First of all, the data must be converted into a binary coverage. Complete information about the output can be obtained in the help files, where the objects returned by the functions are also described.

[1] http://www.grid.unep.ch/data/grid/gnv158.html

Although some error messages appear, it is not worth worrying about them. Those related to coverage *eunamll* are produced because, in fact, it is not a real coverage (it only contains a data table) and the usual important files, those related to arc and polygon definitions, are missing.

The symbols `(...)` have been used to substitute output when it was too long and unworthy. It has been used, for example, to reduce the output when displaying vectors or tables.

5.1 Population in Europe

As shown in the previous output, there are some data about the population in several countries. The next example shows how to summarise population in 1991 and how to create a map using that information. Figure 5 shows the resulting map. It seems that population in several countries was not available and it was substituted by zero. That's why many high populated countries appear in the graphic as having low population levels.

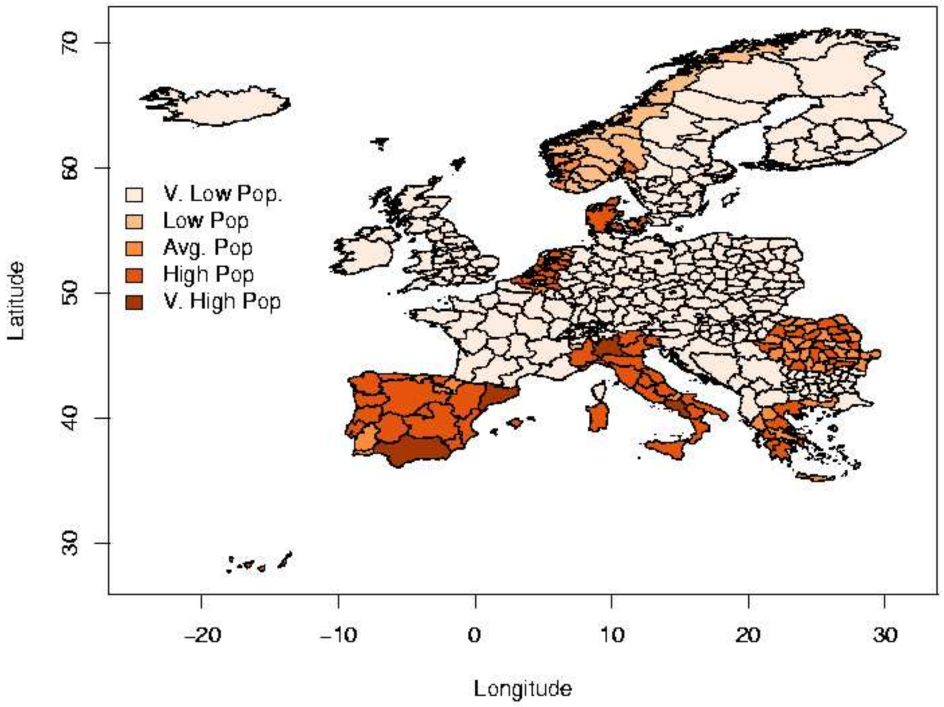


Figure 5: Map of the population in Western Europe. Probably zero population was used to mark non available data.

5.2 Europe Administrative Regions

This example tries to reproduce the map (distributed as a GIF image) which can be found together with the data. It represents, using different colours, countries and borders lines, depending on whether they separate countries or regions within countries.

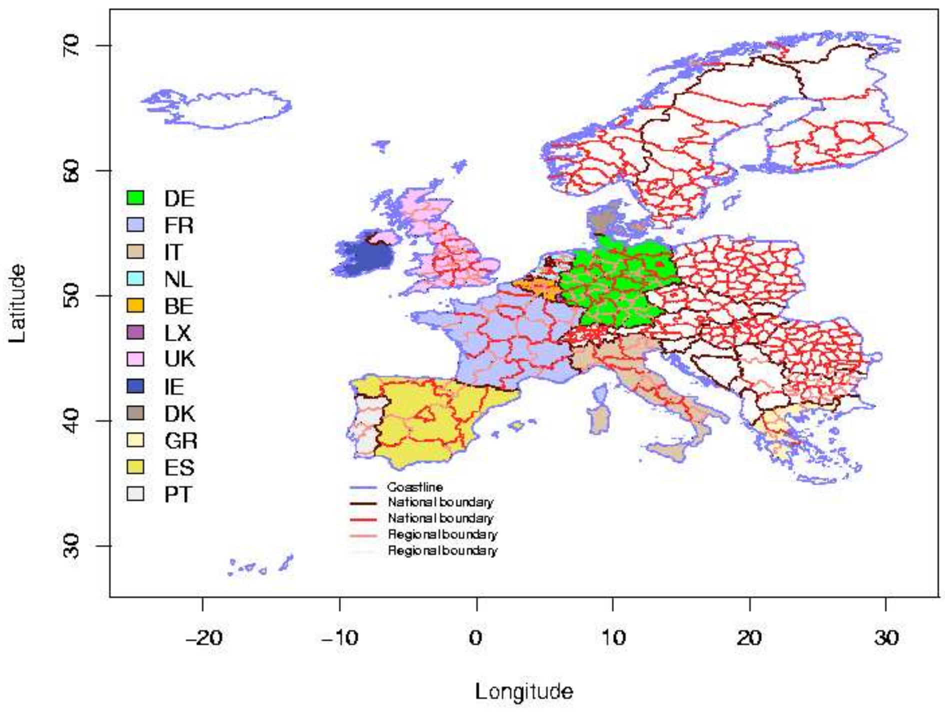


Figure 6: Map of administrative regions in Western Europe.

6 Future work

Many more things can be done in order to improve this package. Some of them could be:

- Improve object representation by creating real objects and not just lists of lists and data frames.
- Write a converter to the objects structures used in the package *maptools*. This would allow conversion from E00/binary coverages into shapefiles within R.
- Write a converter to the data format used in the package *spdep*. This would simplify the use of this package with imported data.

- Write some kind of exporter. I have written some code to export data into an E00 file, but it is far from being finished.

But perhaps the most amazing option would be integrate vector data format inside R, be it through the RGRASS interface or as a separated package. This effort would require consensus among the R community and people who have developed package related to spatial statistics to create an unified format for spatial data. Efforts are going into this direction!

7 Acknowledges

I'd like to thank several people for their comments and support, especially to J. Ferrándiz and A. López (for encouraging me to write this package and continuous advice), D. Morisette (for his marvellous AVCELIB library), E. Neuwirth (for his useful comments, testing and the *RColorBrewer* package) and Roger Bivand (for his invitation to submit this paper and comments).

This work has been partly funded by projects:

- *EUROHEIS: An European Environment and Health Information System for Exposure and Disease Mapping and Risk Assessment* - codes SI2.132454 (99CVF2-606), SI2.291820 (2000CVG2-605) and SI2.329122 (2001CVG2-604).
- *Collaboration Agreement between Consellería de Sanitat and Universitat de València.*

References

[1] R. Bivand. Using the r statistical data analysis language on grass 5.0 gis database files. *Computers & Geosciencies*, 26:1043–1052, 2000.

[2] R. Bivand. More on spatial data. *R News*, 1(3):13–17, September 2001.

[3] R. Bivand. R and geographical information systems, especially GRASS. In K. Hornik and F. Leisch, editors, *Proceedings of the 2nd International Workshop on Distributed Statistical Computing, March 15-17, 2001, Technische Universität Wien, Vienna, Austria*, 2001. ISSN 1609-395X.

[4] R. Bivand and A. Gebhardt. Implementing functions for spatial statitical analysis using the r language. *Journal of Geographical Systems*, 2:307–317, 2000.

[5] R. Bivand and M. 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.

[6] O. F. Christensen and P. J. Ribeiro. georglm: A package for generalised linear spatial models. *R News*, 2(2):26–28, June 2002.

[7] D. Morissette. Arc/Info binary coverage format analysis. *http://pages.infinit.net/danmo/e00/docs/v7_bin_cover.html*, 2000.

[8] D. Morissette. Arc/Info export (e00) format analysis. *http://pages.infinit.net/danmo/e00/docs/v7_e00_cover.html*, 2000.

[9] P. J. Ribeiro, Jr. and P. J. Diggle. geoR: A package for geostatistical analysis. *R News*, 1(2):14–18, June 2001.

[10] B. Rowlingson. Rmap. *http://www.maths.lancs.ac.uk/Software/Rmap/*, 2003.

[11] M. Schlather. Simulation and analysis of random fields. *R News*, 1(2):18–20, June 2001.