A Process and Environment for Embedding The R Software into TerraLib

Pedro Ribeiro de Andrade Neto[1],
Paulo Justiniano Ribeiro Junior[1]

[1]Laboratório de Estatística e Geoinformação (LEG)
Statistics Department – Federal University of Paraná (UFPR)
Curitiba – PR – Brazil

{pedro,paulojus}@est.ufpr.br

Abstract. *Geographical information systems (GIS) and statistical software can enhance their functionality using each other capabilities. This paper describes an environment for coupling the statistical program R and the GIS library TerraLib. The implementation uses and adapts the package aRT being the key aspect the fact that both applications share a database connection which is used for exchanging information. The GIS can send commands to be executed by R, retrieving the results directly from the database. An environment for both implementation agents, statisticians and programmers, is therefore provided work together using their own skills, and a third agent, the final user, having access to R functions through a friendly interface.*

1. Introduction

Coupling between Geographic Information Systems (GIS) and statistical software is a topic of relevance nowadays given the necessity to provide to both each other capabilities. Statistical software can benefit from spatial queries, storage capacity, data arrangements and manipulation of geographic objects from a GIS. Such functionality can be used to build relevant covariate information, neighboring information for spatial analysis among others. GIS can benefit from specialized statistical methods and models in general, with primary interest on topics such as: geostatistics [Goovaerts 1997], global and local spatial patterns [Getis and Ord 1996], analysis of point patterns [Diggle 2003], and spatial statistics in general [Bailey and Gattrell 1995, Cressie 1991].

R is a language and environment for graphics and statistical analysis [R Development Core Team 2005]. The language tools, interfaces and widely used statistical procedures are available in the core of the software. Add-on *packages* have been developed implementing further topics, specialized methods and functionalities. R has grown as a typical Internet project with contributors from all over the world and the development is very much active with constant updates and new contributions. One of the advantages of this mechanism is the continuous enhancement of R resources incorporating cutting edge statistical methodology. R is an open-source project and administered by *The R Foundation for Statistical Computing*.

R has packages implementing statistical methods related to topics relevant to GIS needs. For instance, within the scope of spatial statistics the package geoR [Ribeiro Jr. and Diggle 2001] (among others) implements models for continuous spatial variation (geostatistics), spdep [Bivand 2005] for area data and splancs

[Rowlingson et al. 2005] and spatstat [Baddeley et al. 2005] for analysis of point process. Being a free software, R enables implementing access interfaces to (and from) other programs in an optimized way using its shared library. Integration with R goes beyond just supplying current needs for statistical computation: it can keep the GIS always up-to-dated with recent research and development of statistical modeling of spatial data.

In the literature, there are two typical ways of implementing integration between GIS and R. The first approach is to generate R packages for information exchange. Some examples include the packages GRASS [Bivand and Neteler 2000], RArcInfo [Gómez-Rubio 2005] and aRT [Andrade Neto et al. 2005]. These implementations requires that an user of such tools needs have familiarity with the R language. Therefore these packages are targeted to statisticians and/or R users, but not most of the GIS users.

A second approach for the integration tries to avoid the necessity of the GIS user being a proficient R user in order to access its resources. This is typically implemented building graphical interfaces and calling R functions in the background as for example in [Ono and Murayama 2003, Tait et al. 2004]. Both of these works describe an already implemented tool and their functionality, although not covering how to aggregate new functionalities. Yet another feature is the fact they use files to exchange information with R.

Here we suggest another approach to the integration problem based on two basic features (i) it is meant for a non-R user however still allowing for incorporation of new functionalities and not confined to a pre-defined set of tools already implemented and (ii) a Database Management Systems (DBMS) is the interface for data exchange. TerraLib is a set of classes and functions for building customized GIS [Camara et al. 2004] applications. It is also an open source project and the main purpose is to search for development of a new GIS generation, based mainly in the DBMS. The TerraLib model stores data in a database and this feature enables close coupling with others programs using the database as a tool for information exchanging.

Bearing in mind that TerraLib is a library for building customized GIS, this article looks for an environment which provides R as a library for statistical analysis inside TerraLib. The key point to be addressed is: given a particular statistical analysis which can be performed in R, how to incorporate it into a TerraLib based application? This coupling must satisfy requirements and obey the limits of the three different integration agents:

1. statisticians or R users in general, who implement methods of data analysis using R and wrap them generating TerraLib coupable functions, without having any knowledge of TerraLib programming;
2. TerraLib programmers can quickly develop interfaces for calling wrapped R code, which consists basically of functions and a description of their arguments. This can be done without knowing about R internals such as memory allocation, data structures, among others.
3. final users of a TerraLib based applications will be capable to perform data analysis implemented in R, without knowing the R syntax or even without noticing their analysis are executed by R.

The main objective of this article is to present an environment supporting this process using a DBMS for information exchange between the GIS and R. Section 2 presents a description of package aRT and how to generate functions to be coupled. The integrating

environment is introduced in Section 3 and Section 4 provides details on the implementation and results. Closing remarks and the actual stage of the project are given in Section 5.

2. R Scripts for Data Analysis

This Section describes the R environment showing how data can be analyzed within it. The package aRT is used as a bridge for information exchange with a TerraLib database. This mechanism will allow the R access from TerraLib presented in the next Section.

R is an interpreter for a dialect of S language developed by John Chambers. Therefore, R analysis can be written in a script format. An output from a particular type of a spatial data analysis obtained by using R is illustrated by Figure 1. The left panel displays the data to be analyzed which consists of rainfall measurements taken at 143 meteorological stations located at Paraná state, Brazil. The state borders are also shown in the Figure. Therefore, in this case, data consists of the stations locations as punctual coordinates, rainfall measurements at each locations which are the attributes associated with the points and a polygon defining the state borders. The right panel displays a map of predicted rainfall over the entire state. The predictions were obtained using a geostatistical interpolation method known as *kriging* and, in this case, computed by functions implemented in the package geoR.

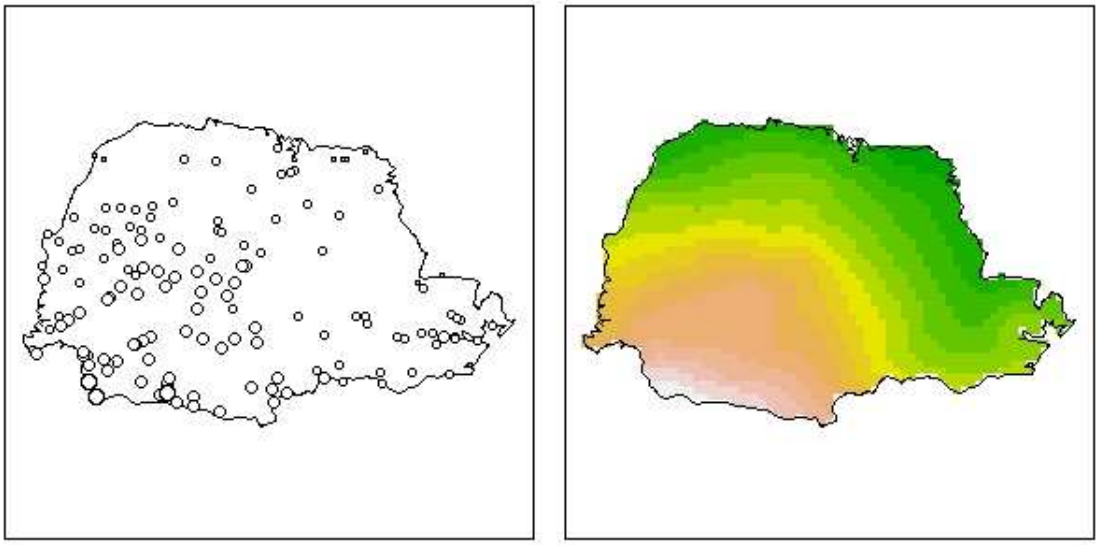

Figure 1. Example of spatial data analysis

For such analysis R needs a way to access the data source whether or not stored in a database. Our objective here is to discuss how to incorporate such analysis into TerraLib. To do so, let's assume that this data is initially stored in a DBMS. Within R we use the package aRT to access the data stored in a TerraLib database. The functions in the package aRT encapsulates TerraLib objects into R variables and can execute spatial and/or temporal queries over the database using TerraLib. Trying to be a transparent interface to the data analyst, aRT manipulates data following the data format defined by the package sp [Pebesma and Bivand 2005]. This package provides data structures for spatial objects and providing an uniform interface for handling spatial data in R.

A typical script to any statistical analysis using aRT can be divided in four basic steps:

1. establish a connection with the database;
2. load data into R, including executing geoprocessing functions if needed;
3. perform statistical data analysis within R;
4. store results in the database.

One schematic example of an R code to perform the necessary analysis which generates Figure 1 starting from a TerraLib database and using the package aRT is shown in Figure 2. Blank lines separate the four steps mentioned above. In the first step, a DBMS

```
conn      = openConn()
db        = openDb(conn, "dbname")
thpoints  = openTheme(db, "themepoints")
thcontour = openTheme(db, "themecontour")

points  = getPoints(thpoints)
contour = getPolygons(thcontour)
data    = getData(thpoints)

raster = krige(points, contour, data)

l = createLayer(db, "lraster")
addRaster(l, raster)
thraster = createTheme(l, "themeraster")
```

Figure 2. An example of an R analysis using aRT

connection is established and the R object `conn` stores the information about the connection. From this a connection to a particular database is opened. The relevant data for this example involves two geometries: points and polygons, with attributes associated to the former. These are stored in two TerraLib's *themes* to which a connection is established allowing for the data transfer. In a second step points, their attributes and the polygon are loaded into R. From this follows the third step where the statistical analysis *per se* is performed. In the example the R function is called `krige()` encapsulates one or a sequence of calls to R functions for geostatistical analysis. The final step assumes the goal here is simply to incorporate the map of predictions to the database. To do so a new TerraLib *layer* is created to store the raster, the raster data is added to the database and a new theme is also created from R allowing visualize the data from TerraLib applications. Notice that an analysis following this format does not need to return any value because all information is exchanged using the database.

Internally aRT uses TerraLib objects in a completely transparent way to R users. The key TerraLib object is the database connection (called `TeDatabase`). An object of this class stores not only a database connection but also meta-data objects referring to the database content. As for the example shown in Figure 2 `db` internally stores a database connection pointer. All objects created from it store a pointer to the same `TeDatabase`. It is also important to notice that `TeDatabase` objects are used in all TerraLib-based GIS to connect to databases.

3. Coupling Specification

There is not a single way to implement an interface to R from TerraLib. In this section we examine some alternatives and justify our choice.

A fast and intuitive way to incorporate a function as described in the previous section would be simply to execute the hole script inside R followed by getting the results from the database. However, this is not an efficient implementation because aRT establishes its own database connection and, after the analysis, the meta-data of the GIS's connection to the database would be inconsistent, forcing a reconnection which takes computing time.

A second alternative is to read the data from the database, use aRT converters to build R objects in memory, and then send them to R for the analysis. This solution is efficient in terms of executing, but it would imply in a further work to TerraLib developers.

aRT uses TerraLib objects to establish the connection to databases, like any other TerraLib-based GIS. Therefore, we propose a yet another solution which differs from the above in the fact that both applications share the same database connection. Provide they use the same `TeDatabase` object there is no concurrence, which ensures the consistency of the object.

It is necessary to standardize the way on how the R scripts for data analysis are written to be easily incorporated into TerraLib by its developers. In our proposal they must be codified as follows. All analysis script must have a main function, that receives as arguments a database connection and all necessary arguments that can be converted to a string easily. These arguments may contain names of database elements where to load and/or save data, and some numeric values, typically thresholds. For example, the analysis of Figure 2 can be encapsulated in a function as shown in Figure 3. This function gets as arguments the database connection and four other arguments with names of a layer and themes. Once a statistician implements the data analysis, using aRT to data exchange, he or she must convert the function to this format, and then write a documentation about each argument, to be used by TerraLib programmers.

```
tlKriege = function(db, tpointsname, tcontourname,
                    lrastername, trastername) {
  thpoints = openTheme(db, tpointsname)
  thborder = openTheme(db, tbordername)

  points = getPoints(thpoints)
  border = getPolygons(thborder)
  data   = getData(thpoints)

  raster = krige(points, border, data)

  l = createLayer(db, lrastername)
  addRaster(l,raster)
  thraster = createTheme(l, trastername)
  return(invisible())
}
```

Figure 3. Encapsulating commands for an analysis in an R function

One disadvantage of this method resides in the fact that once an R function is

called the execution environment is changed and TerraLib geoprocessing functions can only be executed through aRT. However it is defensible, because one statistician can prototype analysis with the function if he or she has access to it using aRT. Therefore, aRT must implement the call of all geoprocessing functions needed by the analysis.

4. Implementation

For implementing the proposed environment, there are some structural changes needed in aRT package to support the integration. We changed the package to enable the creation of objects using `TeDatabase` from outside R. Also, as this object was not allocated in R, it cannot be removed from memory by the R garbage collector.

A new TerraLib class is necessary for accessing R and execute functions. This class must have three functions, described as follows:

Initialize the environment: initialize R, and load the necessary packages and functions.

Declare a `TeDatabase`: This operation implements the database connection sharing between the two programs.

Execute a command: This function calls an R function, following the format described in the last Section. It returns a string with an error code. If the string is empty, the operation was successfully executed. This avoids any programmer control of R returning values.

For this implementation on the TerraLib side, we built a class named `TeRwrapper`. To load R and execute commands, we used the access provided by MyR library [Andrade Neto et al. 2004], in such a way that TerraLib can be used to call R functions as the one described in Figure 3. An example of a C++ code to execute this function can be seen in Figure 4. As we allocate a `TeRwrapper` object, the R environment is initialized and all functions are loaded. A `TeDatabase` is declared into R using function `declareDatabase`, that receives the object name inside R as additional argument. A string with the function calling is built using data chosen by the user, and then it is executed in R with `execute`. If this function returns an error code, the program analyzes it and reports the error to the user. If not, it simply calls a redraw event to actualize the screen. With this, the integration model is complete, and a graphical description of it can

```
TeDatabase* db;
char* str;
TeRwrapper* Rwrapper;
string error;
...

Rwrapper = new TeRwrapper();
Rwrapper->declareDatabase(db, "db");
asprintf(&str, "tlKrige(db,%s,%s,%s,%s)",
         themepoints.c_str(),  themeborder.c_str(),
         layerraster.c_str(),  themeraster.c_str());
error = Rwrapper->execute(str);
if(error != "") ...

OnPaint();
```

Figure 4. Calling an R function using `TeRwrapper`

be visualized in Figure 5. Both applications share a `TeDatabase` object, which enables the database access. The TerraLib-based GIS has control over R, send commands to be executed and receive error codes. Therefore, it characterizes a close coupling integration [Bivand and Neteler 2000].

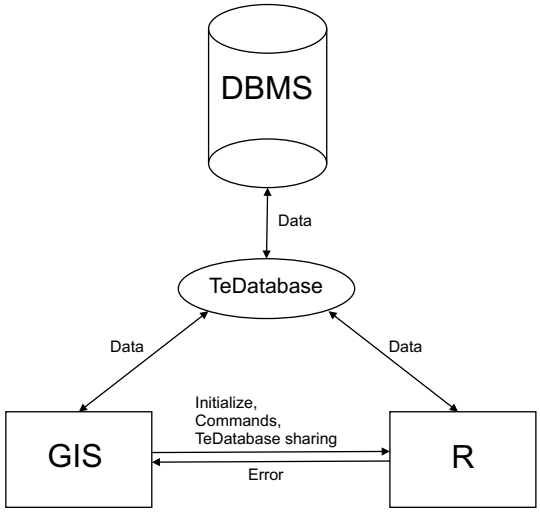

Figure 5. Coupling schema

As initial test for this interface, we developed a C++ program isolated from any graphic interface. It manipulates a `TeDatabase` in the same way of TerraView and aRT. This program receives from the keyboard layers and themes names to be used/generated. Then, a kriging analysis is called, and the result of this operation can be visualized in TerraView program, as shown in Figure 6. Note that aRT alone could execute this operation and generate the same result, and in fact aRT does that, but the focus now is how to use this functionality without any R knowledge.

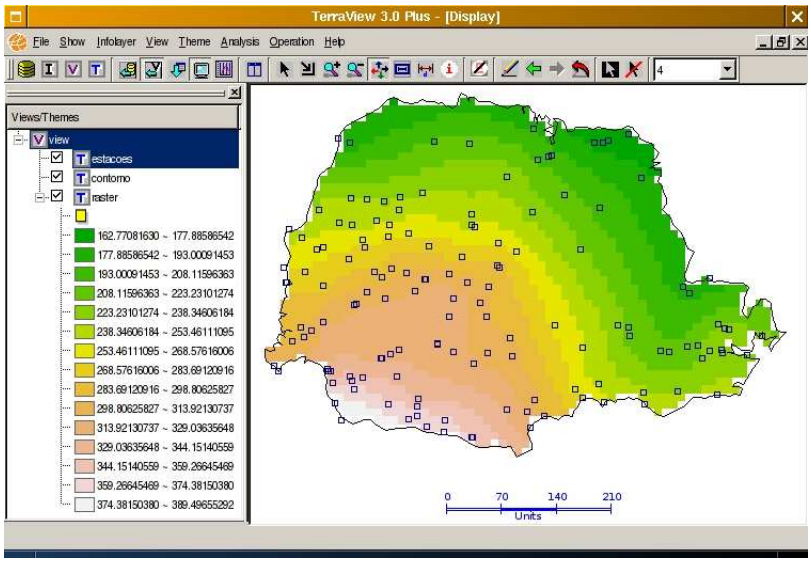

Figure 6. Plotting an R analysis in TerraView

5. Conclusions and Future Work

This article describes an on going project seeking for integrating a GIS toolkit TerraLib and the statistical software R. aRT package enables an integrated analysis environment allowing access from R to TerraLib databases and functions. It gives a powerful analysis environment for statistics professionals, enabling statistical modelling together with geo-processing functions in an efficient manner. Being R manipulation not trivial for the major GIS users, we propose a process and environment for aggregating R statistical functions into TerraLib based programs, in a way that both implementation agents, statisticians and programmers, can work together focusing on their own skills, and a third agent, the final user, can easily use R functions through a friendly interface. Therefore, statistical analysis can now be prototyped writing R functions instead of a TerraLib/C++ environment, and the result can be aggregated to TerraLib.

This environment also follows one of the main TerraLib project purposes: to enable building customized GIS. Only the R functions needed to solve a specific problem will be used in the GIS. If such analysis does not exist in R, it can be prototyped using R and then integrated to the GIS.

Two further developments can be added to the integration process. One is to incorporate the integrated environment into TerraView providing a more friendly interface to the user. The other is to define a documentation language for specify R functions in such a way that it would be possible to generate part of the graphical interface and error handling automatically.

Acknowledges

This work was partially funded by SAUDAVEL project, CNPq grant 552044/2002-4. It is currently developed as part of the activities of the REDE SAUDAVEL. Further details are available at `http://saudavel.dpi.inpe.br/`. This project is been hosted by LEG/DEST/UFPR, with support of TerraLib project developers from DPI/INPE.

References

Andrade Neto, P. R., da Silva, E. S., de Mello, T. B., Carrero, M. A., and Ribeiro Jr, P. J. (2004). myR - uma biblioteca C++ para acesso ao R. In *Resumo dos Trabalhos Apresentados - 16º SINAPE*, page 233, Caxambu, Brazil.

Andrade Neto, P. R., Ribeiro Jr, P. J., and Fook, K. D. (2005). Integration of statistics and Geographic Information Systems: the R/TerraLib case. In *GeoInfo 2005*, Campos do Jordão, Brazil.

Baddeley, A., Turner, R., et al. (2005). *spatstat: Spatial Point Pattern analysis, model-fitting and simulation*. R package version 1.6-6.

Bailey, T. and Gattrell, A. (1995). *Spatial Data Analysis by Example*. Longman. London.

Bivand, R. (2005). spdep: Spatial dependence: weighting schemes, statistics and models.

Bivand, R. and Neteler, M. (2000). Open source geocomputation: using the R data analysis language integrated with GRASS GIS and PostgreSQL data base systems. In *Proceedings of the 5th International Conference on GeoComputation*.

Camara, G., Monteiro, A. M. V., Souza, R. C. M., Vinhas, L., Ferreira, K. R., Garrido, J. C. P., Queiroz, G. R., Carvalho, M. T. M., Casanova, M. A., and Freitas, U. M. (2004). TerraLib: The architecture of an open source GIS library. In *V Workshop on Free Software*, Porto Alegre, Brazil.

Cressie, N. (1991). *Statistics for Spatial Data - revised edition*. Wiley, New York.

Diggle, P. (2003). *Statistical Analysis of Spatial Point Patterns*. Edward Arnold, London, 2 edition.

Getis, A. and Ord, J. K. (1996). Local spatial statistics: an overview. In *Longley ,P., Batty ,M., eds. Spatial Analysis: Modelling in a GIS Environment*, pages 261–277. John Wiley, New York.

Goovaerts, P. (1997). *Geostatistics for Natural Resources Evaluation*. Oxford University Press. New York.

Gómez-Rubio, V. (2005). RArcInfo: Functions to import data from Arc/Info. `http://sourceforge.net/projects/rarcinfo`.

Ono, H. and Murayama, Y. (2003). Development of integrated spatial analysis system using open sources. In *Proceedings of the 7th International Conference on GeoComputation*.

Pebesma, E. and Bivand, R. (2005). sp: classes and methods for spatial data. R package version 0.7-12. `http://r-spatial.sourceforge.net`.

R Development Core Team (2005). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Ribeiro Jr., P. J. and Diggle, P. J. (2001). geoR: A package for geostatistical analysis. *R-NEWS*, 1(2). ISSN 1609-3631. Available for download at: `http://cran.r-project.org/doc/Rnews`.

Rowlingson, B., Diggle, P., et al. (2005). Splancs: Spatial and space-time point pattern analysis. `http://www.maths.lancs.ac.uk/~rowlings/Splancs/`.

Tait, N., Durr, P. A., and Zheng, P. (2004). Linking R and arcGIS: Developing a spatial statistical toolkit for epidemiologists. In *GisVet'04*, pages 33–35.