Integration of Statistics and Geographic Information Systems: the R/TerraLib case

Pedro Ribeiro de Andrade Neto[1]
Paulo Justiniano Ribeiro Junior[1]
Karla Donato Fook[2,3]

[1]Laboratório de Estatística e Geoinformação (LEG)
Departamento de Estatística – Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brazil

[2]Divisão de Processamento de Imagens
Instituto Nacional de Pesquisas Espaciais (INPE)
São José dos Campos – SP – Brazil

[3]Departamento Acadêmico de Informática
Centro Federal de Educação Tecnológica do Maranhão (CEFET-MA)
São Luís – MA – Brazil

{pedro, paulojus}@est.ufpr.br, karla@dpi.inpe.br

Abstract. *This work presents a panorama of GIS integration in Spatial Statistics environments. It highlights the current needs of communities considering such integration. Spatial Statistics is treated in a context focused on the use of computational tools. A review of types of integration is accomplished and a new approach is proposed integrating the statistical software R and the GIS library TerraLib.*

1. Introduction

Software technology is constantly evolving. Computers are becoming faster, and software projects must follow the availability of resources. This scenario favors greater complexity of software projects allowing multidisciplinary programs to combine research in several knowledge areas, like medicine, agronomy and civil engineering, among others.

Geographic Information Systems (GIS) are an example of a multidisciplinary software. GIS comprise knowledge in areas such as cartography, databases and software engineering, among others. They are capable to manipulate several complex georeferenced data structures. In practice, statistical analysis are often necessary to extract useful information from the available data. For instance, methods of spatial analysis are particularly relevant to GIS which, in general, have limited capability to perform such analysis.

On the other hand statistical softwares implement specialist algorithms for spatial statistics needed by GIS. Typically they have a large variety of functions and provide a friendly language for statistical programmers. However, they have limited if none at all access to technologies found in GIS, for instance, complex geographical databases and geoprocessing algorithms.

Integration is technique that combines software components in order to generate more complex systems. This technique is an efficient form of software project, saving time and resources, and it enables to use specialized tools in each area of the wider project.

Tools for statistical analysis of spatial data and GIS are used in areas such as health, meteorology, monitoring environmental and mineral exploration, among others [Druck et al. 2004]. Integration between spatial statistics and Geographic Information Systems (GIS) can increase the individual effectiveness of both sides. There is a couple of statistical programs implementing spatial analysis without any essential component provided by GIS. Example are robust spatial database, spatial models and plotting algorithms [Krivoruchko 2003].

Software integration can be classified according to their architectural characteristics [Zhao 2002], and can be divided in three approaches: the full integration, loose coupling and close coupling [Goodchild et al. 1992, Bailey and Gattrell 1995, Fischer et al. 1996]. An important factor to be considered in an integration process is the level of common knowledge necessary the user to accomplish his work. Integration between two tools that does not request any learning from the user is more transparent, and therefore stronger.

In the full integration mechanism, the statistical analysis tools are incorporated directly in the GIS. This is the case of the incorporation of the geostatistics, K function and kernel density map in GIS SPRING [Camara et al. 1996]; and of the empirical Bayes local and global estimates in the free software and open source TerraView[1]. However, this integration is not the most promising for more sophisticated techniques of spatial data analysis, because the inclusion of a great number of spatial statistics functions in a GIS increases the complexity of the system for common users, and involves a large investment in software programming and maintenance.

In the two coupling approaches, both softwares evolve independently. In loose coupling implementations, data is exported from GIS to the statistical analysis program, and the results are exported for GIS for visualization. Figure 1 displays this integration type. Here, the user is responsible for the whole flow of information, and he/she needs to know who data is manipulated in both programs, and also how to use the programs.

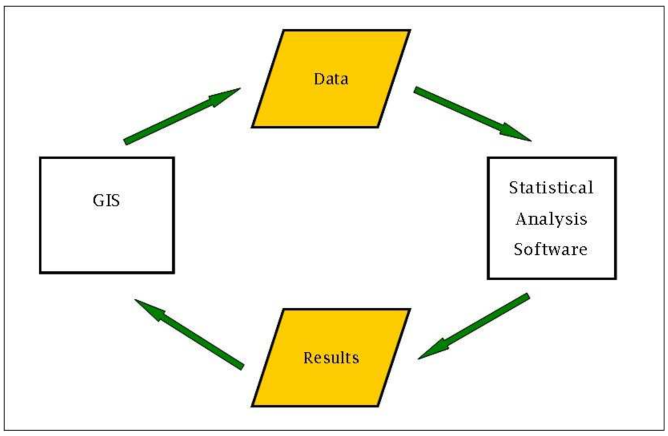


Figure 1. Loose coupling [Bivand and Neteler 2000]

The loose coupling model usually does not require any effort from the programs developers, because it only requires a common data format for information exchanging.

[1]`http://www.dpi.inpe.br/terraview/index.php`

As example of loose coupling we can cite the integration of SPRING with SpaceStat [Anselin 1992, Camara et al. 1996], where SPRING exports data using a vectorial format to be used in SpaceStat. Another example is the coupling of the proprietary GIS ArcInfo and R, the RarcInfo package, that imports and exports files following the ArcInfo format [Gómez-Rubio 2005]. However, this import/export process can generate an overhead when we need a more complex data analysis. As many spatial queries are required during the statistical processing, it is harder for the user to execute such analysis.

In the integration by close coupling, there is a stronger linking between the tools, because the mechanism allows the calling of statistical procedures directly from the GIS, and vice versa. It also involves the writing of a program to facilitate the integration process, reducing the charge imposed to the user [Goodchild et al. 1992]. The mechanism of close coupling is shown in Figure 2.

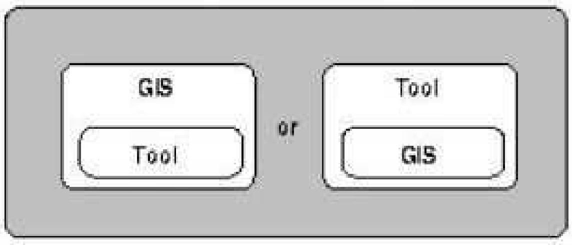


Figure 2. Close coupling [Bivand and Neteler 2000]

An example of close coupling is the integration between S-PLUS (by MathSoft) and ArcView (by ESRI), both proprietary softwares [Bao et al. 2000]. The authors implement an extension for ArcView, including two menus in the user interface. But, internally it works as a loose coupling, except by the fact that the program loads R and sends commands to it directly.

The integration of the open source GIS GRASS (Geographical Resources Analysis Support System) and R is another example of close coupling [Bivand and Neteler 2000]. R can be loaded inside GRASS using the package called GRASS, and GRASS commands are executed in R. One new initiative from R/GRASS coupling is that the user can use a database for some information exchange. But the database stores only attributes, and the user need to know how to manipulate databases.

These close integrations externally are better for the users, because they unify two programs in one, but they still have the same problems as loose coupling: the data exchanging. An interesting approach is proposed by Fischer [Fischer et al. 1996]: a proposal that enables the access directly from a *geographic* database is desirable, because it allows the softwares to share a common database, without loosing higher topologic structures, object identity, metadata and other relationships. None of the above works fit in this description.

In this scope, this article presents an R package implementing a close coupling between R and the geoprocessing library TerraLib, called aRT (R-TerraLib API). This package follows the TerraLib data model, and allows to call TerraLib functions directly, without any context exchange, because TerraLib is a dynamic library, and it is loaded together with the package. It also does not use an external file for information exchange, because all data is stored in a DBMS. The package only requires from the user to know how the TerraLib spatial/temporal model works.

This article is written in the following manner. R and TerraLib environments and their advantages in the integration are detailed in the Sections 2 and 3, respectively. Section 4 presents aRT package, coupling TerraLib inside of R, and Section 5 we conclude and discuss future works.

2. R Project

The statistical program R[2] is an open source tool under GPL. R is a language and environment for statistical computing and graphics, within which statistical techniques are implemented. The R program is a command interpreter, and the interpreted language is referred as a dialect of the award winning S language [R Development Core Team 2005].

R implements a great diversity of statistical methods, which are available in the form of packages. The packages are developed by members of the project, as by the community. There is a set of basic packages supplied with the R distribution and many more are available through the CRAN family of Internet sites[3], which covers a very wide range of modern statistics. All R packages cited in this section are called recommended or contributed, and they are available at CRAN.

Many classical and modern statistical techniques are implemented in R environment. As example of areas and their packages for spatial analysis we can cite:

Point pattern analysis: spatstat package allows to define regions of interest, and makes extensions to marked processes and spatial covariates. Its strengths are model-fitting and simulation. The splancs package also allows point data to be analyzed within a polygonal region of interest, and covers many methods, including 2D kernel densities. ash package has functions for binning points on grids.

Geostatistics: The gstat package provides a wide range of functions for univariate and multivariate geostatistics, also for larger datasets, while geoR and geoRglm contain functions for model-based geostatistics. A similar wide range of functions can be found in the fields package.
The RandomFields package provides functions for the simulation and analysis of random fields. For diagnostics of variograms, the vardiag package can be used. The spatialCovariance package supports the computation of spatial covariance matrices for data on rectangles.

Disease mapping and areal data analysis: DCluster package detects spatial clusters of diseases. spdep package provides functions for spatial autocorrelation for areal data like Moran's I, and functions for fitting spatial regression models.

Ecological analysis: grasper package provides environmental prediction functions using GAM, ade4 has exploratory and Euclidean methods in the environmental sciences, adehabitat allows the analysis of habitat selection by animals, pastecs does the regulation, decomposition and analysis of space-time series, and vegan provides ordination methods and other useful functions for community and vegetation ecologists.

[2]`http://www.r-project.org`
[3]`http://cran.r-project.org`

Among all these packages that manipulate spatial data, is important to cite the sp initiative. sp comes to supply the lack of a common data structure for handling spatial data in R, and provides data structures for spatial objects. It is a new initiative, and the authors hope that packages join the initiative quickly.

R works with data stored in memory, and it can be a problem when we use large datasets. A way to skirt this problem is using an external database access, for example PostgreSQL or MySQL [Bivand and Neteler 2000]. There are R packages enabling database access, as RMySQL and RPgSQL, but all of them require SQL language to work with, because data have to be manually pushed, and the user must have a database modelling knowledge for working with a large amount of tables. It becomes even harder when working with spatial/temporal databases, when we need different tables to store attributes and geometry, and they need to be linked in some way.

3. The TerraLib Library

TerraLib is a library of C++ classes that offers functions and data structures for building customized geographical applications. TerraLib is an open source and free software, and its main objective is to provide a powerful environment for GIS development in a new generation of GIS, once it incorporates space-time support to conventional Database Management Systems (DBMS).

The philosophy of TerraLib development is to use the current mechanisms of the C++ language, for example Standard Template Library (STL), parameterized classes and multi-paradigm programming (generic programming, object orientation and design patterns) [Vinhas et al. 2002]. The geographical data model of TerraLib is a conceptual model of geographical database in which the processing algorithms are written.

A TerraLib database is a repository of information that contemplates the geographical data and their organizational model. TerraLib supports some commercial and public domain DBMS. A DBMS need only to have the capacity to store long binary fields, or an own extension supporting abstract spatial types, for supporting TerraLib. The access to a TerraLib database is implemented in a virtual class called TeDatabase. It contains methods for creating, populating and querying a database. Concrete classes, called drivers, specialize TeDatabase, and each class implements the inherent features needed by one DBMS to support TerraLib functionality. Therefore, TerraLib enables using its geoprocessing functions without care about which DBMS is storing the data. A class diagram of TerraLib supported DBMS's is shown in Figure 3.

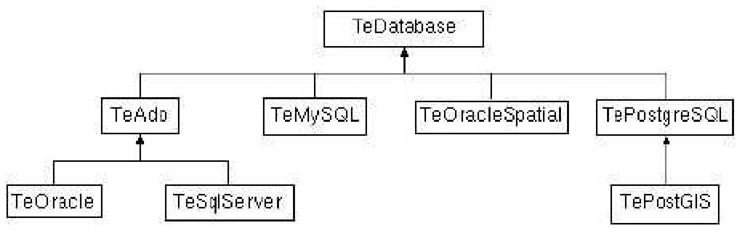


Figure 3. Class Diagram ([Casanova et al. 2005])

The conceptual model of a TerraLib database is composed by several entities.

Here we describe some of them [Casanova et al. 2005]:

Layer: a collection of spatial information located on the same geographical area and sharing attributes. Each layer may contain any type of geometry, and all objects have the same spatial projection. Layers does not store attributes directly, but they can have *attribute tables*. Examples of layers are thematic maps, cadastral maps of geographical objects and raster data.

Attribute Table: a collection of non-geographical data. There is several kinds of attribute tables in TerraLib. Each record of a *static table* refers to one spatial object. *External tables* store information not linked to any geometry. *Temporal tables* manipulate information where attributes and (or) geometry can change with the time.

Theme: built from a layer data, a theme can contain all the layer, or a subset of the layer objects, selected by a temporal, spatial or attribute restriction. It can select one or more layer tables, joining them. Themes are also used in TerraLib-based GIS for visualize and classify the data, and for temporal slicing.

View: a container of themes. It has an associated projection, and then theme data with different projections are converted to the view's projection before drawing. Therefore a view indicates that there is no projection problems when drawing all its themes in a same plotting. Each theme is associated to one, and only one, view.

TerraLib also allows to execute database queries in a proxy way, retrieving the content of a layer or theme in parts, instead of all objects once. This queries can be executed using themes, and therefore they use the temporal/spatial/attribute restrictions of the theme.

4. aRT

aRT (R-TerraLib API[4]) is an R package for integrating R to TerraLib. It implements some classes encapsulating TerraLib objects and functions into R objects, and its main objective is to enable the access to the TerraLib entities in a easy and transparent way to R users. The actual aRT version (0.4-1) implements seven classes, described as follows:

aRTconn: represents a virtual DBMA connection. It can perform some database administration tasks, and manipulates aRTdb objects.

aRTdb: stores a real connection to a DBMA database, and can open/create aRTlayers. Only this object can remove TerraLib objects from the database.

aRTlayer: represents an information layer inside a database. In aRT, each layer manipulates only one type of geometry, and it is capable to execute spatial queries as metrics and relations. Although it cannot manipulate attributes, following the TerraLib model, it can open/create aRTtables, aRTthemes and aRTqueriers.

aRTtable: stores a database table, and data is manipulated using data.frames. Each table has one specific type, following the types established in the TerraLib model. Tables retrieve data to an R user without any attribute restriction.

aRTtheme: created from an aRTlayer, it joins the selected tables, and choose attribute restrictions. It can also create aRTquerier objects.

aRTquerier: encapsulates a database query. If it is created from a temporal theme, then a temporal slicing can be defined. It retrieves data one by one, slide by slide (if temporal), instead of all attributes/geometry once, as layers and tables.

[4]`http://www.est.ufpr.br/aRT`

aRTvisual: defines a colors and style configuration to be used in aRTthemes. It is useful when using aRT with other TerraLib based software to visualize the results stored in a database.

aRT implements the data exchange to the database using the sp format, to facilitate using data analysis from other packages. Actually, aRT is available in Linux and Windows, and it supports connections only to MySQL.

We illustrate the usage of aRT performing a geostatistical analysis on data stored on a database. The database contains a data set with calcium content measured in soil samples taken from the 0-20cm layer at 178 locations within a certain study area divided in three sub-areas. The elevation at each location was also recorded. The first region is typically flooded during the rain season and not used as an experimental area. The calcium levels would represent the natural content in the region. The second region has received fertilizers a while ago and is typically occupied by rice fields. The third region has received fertilizers recently and is frequently used as an experimental area [Capeche 1997].

First a connection with a MySQL DBMS is established at the localhost and user ”pedro”. With the DBMS connection, we can open the database, called ca20.

```
> conn = openConn(user = "pedro", host = "localhost")
```

```
Trying to connect ... yes
```

```
> db = openDb(conn, "ca20")
```

```
Connecting with database 'ca20' ... yes
Loading layer set of database 'ca20' ... yes
Loading view set of database 'ca20' ... yes
```

When the connection is established, some metadata about the layer and view sets are loaded in the TerraLib object. Note that the R object does not need to have the same name of the database. There are two geometric entities (points and regions) stored in two layers which can be listed and opened using `db`.

```
> showLayers(db)
```

```
[1] "lpoints"   "lpolygons"
```

```
> lpoints = openLayer(db, "lpoints")
```

```
Opening layer 'lpoints' ... yes
```

```
> lpols = openLayer(db, "lpolygons")
```

```
Opening layer 'lpolygons' ... yes
```

Plotting of this data is shown in Figure 4 and generated by the following commands:

```
> plot(lpols)
> plot(lpoints, add = T)
```

The first layer contains 178 points and an attribute table (taltitude) with measurements of calcium content and the elevation of each point. The second stores three polygons, representing the sub-regions. To get the data from the table we need to open it from the layer.

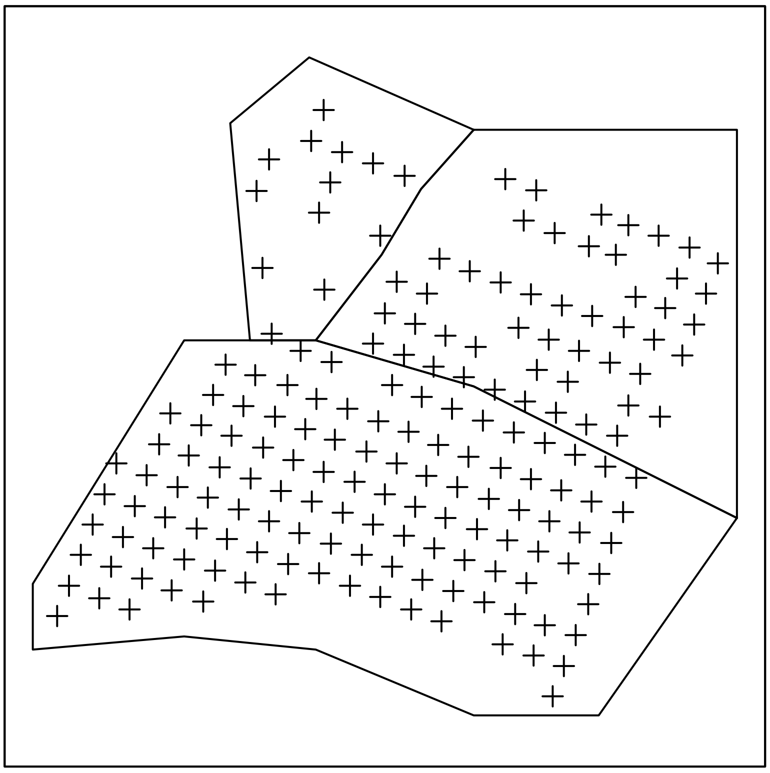


Figure 4. Visualization of the data from the database ca20

```
> lpoints

Object of class aRTlayer

Layer: "lpoints"
Number of polygons: 0
Number of lines: 0
Number of points: 178
Layer does not have raster data
Projection Name: "NoProjection"
Projection Datum: "Spherical"
Projection Longitude: 0
Projection Latitude: 0
Tables:
    "taltitude": static

> tpoints = openTable(lpoints, "taltitude")

Opening table 'taltitude' from layer 'lpoints' ... yes

> tpoints

Object of class aRTtable

Table: "taltitude"
Type: static
```

```
Layer: "lpoints"
Rows: 178
Attributes:
    id: string[16] (key)
    data: real
    altitude: real
```

To load the data inside R we use the layers to get the geometry and choose the table to get the attributes. As we do not want to get any attribute from the layer of polygons, we can read the data using only the layer as argument.

```
> data = getPoints(lpoints, tpoints)
> pols = getPolygons(lpols)
```

As an example of aRT functionalities of spatial queries, we calculate to which polygon each point belongs to. This will be used as a covariate in the data analysis.

```
> pointsid = getID(data)
> polsid = getID(pols)
> areas <- vector("integer", length(pointsid))
> for (i in 1:length(polsid)) {
+     y = getRelation(lpoints, "within", lpols,
+         id = polsid[i])
+     for (j in 1:length(y)) areas[which(pointsid ==
+         y[j])] = polsid[i]
+ }
```

```
Calculating a spatial relation on layer 'lpoints' ... yes
Calculating a spatial relation on layer 'lpoints' ... yes
Calculating a spatial relation on layer 'lpoints' ... yes
```

This data can be added to the database as a new column of the static table:

```
> newdata = data.frame(cbind(id = pointsid, area = areas))
> updateColumns(tpoints, newdata)
```

```
Checking for column 'area' in table 'taltitude' ... no
Creating column 'area' in table 'taltitude' ... yes
Converting 2 attributes to TerraLib format ... yes
Converting 178 rows to TerraLib format ... yes
Updating columns of table 'taltitude' ... yes
```

```
> tpoints
```

```
Object of class aRTtable

Table: "taltitude"
Type: static
Layer: "lpoints"
Rows: 178
Attributes:
    id: string[16] (key)
    data: real
    altitude: real
    area: string[1]
```

As the data consists of three polygons we now create a polygon representing the borders of the area which results from the union of the three original polygons using the function getSetOperation.

```
> border = getSetOperation(lpolygons, "union",
+     id = polsid)
```

From the data recovered from the database we build an object of the class geodata called ca20 which is convenient for using functions in the geoR package for geostatistical analysis.

```
> ca20 = as.geodata(data, data.col = 2, covar.col = 3)
> ca20$covariate$area = factor(areas)

> ml3 <- likfit(ca20, ini = c(100, 500), trend = ~area)
> border = border@polygons[[1]]@Polygons[[1]]@coords
> border = as.data.frame(border)
> loc0 <- pred_grid(border, by = 10)
> loc.area <- rep(NA, nrow(loc0))
> f = function(x) pols@polygons[[x]]@Polygons[[1]]@coords
> l = lapply(1:3, f)
> loc.area[.geoR_inout(loc0, as.data.frame(l[[1]]))] <- 1
> loc.area[.geoR_inout(loc0, as.data.frame(l[[2]]))] <- 2
> loc.area[.geoR_inout(loc0, as.data.frame(l[[3]]))] <- 3
> loc.area <- as.factor(loc.area)
> KC <- krige.control(trend.d = ~area, trend.l = ~loc.area,
+     obj = ml3)
> kc <- krige.conv(ca20, loc = loc0, krige = KC,
+     bor = border)
> georpred <- .prepare.graph.kriging(locations = loc0,
+     borders = border, values = kc$pred)
```

The prediction by kriging methods results in a raster which can be stored in the database.

```
> lraster = createLayer(db, "lraster")

Building projection to layer 'lraster' ... yes
Creating layer 'lraster' ... yes

> addRaster(lraster, georpred)

Initializing the raster ... yes
Adding raster data to layer 'lraster' ... yes
Reloading tables of layer 'lraster' ... yes
```

Finally, we create some themes allowing for direct visualization from TerraView. Views are not important to aRT, therefore there is not a class to encapsulate it, and we only use a name when building a theme.

```
> thpoints = createTheme(lpoints, "points", view = "view")
Checking for theme 'points' in layer 'ca20' ... no
Creating theme 'points' on layer 'lpoints' ... yes
Checking for view 'view' in database 'ca20' ... no
Creating view 'view' ... yes
Inserting view 'view' in database 'ca20' ... yes
Checking tables of theme 'points' ... yes
Saving theme 'points' ... yes
Building collection of theme 'points' ... yes
> thpols = createTheme(lpols, "polygons", view = "view")
Checking for theme 'polygons' in layer 'ca20' ... no
Creating theme 'polygons' on layer 'lpolygons' ... yes
Checking for view 'view' in database 'ca20' ... yes
Checking tables of theme 'polygons' ... yes
Saving theme 'polygons' ... yes
Building collection of theme 'polygons' ... yes
> setVisual(thpols, visualPolygons(color = "black",
+     transp = "100"))
> thraster = createTheme(lraster, "raster", view = "view")
Checking for theme 'raster' in layer 'ca20' ... no
Creating theme 'raster' on layer 'lraster' ... yes
Checking for view 'view' in database 'ca20' ... yes
Checking tables of theme 'raster' ... yes
Saving theme 'raster' ... yes
> setVisual(thraster, visualRaster(terrain.colors(15)),
+     mode = "raster")
```

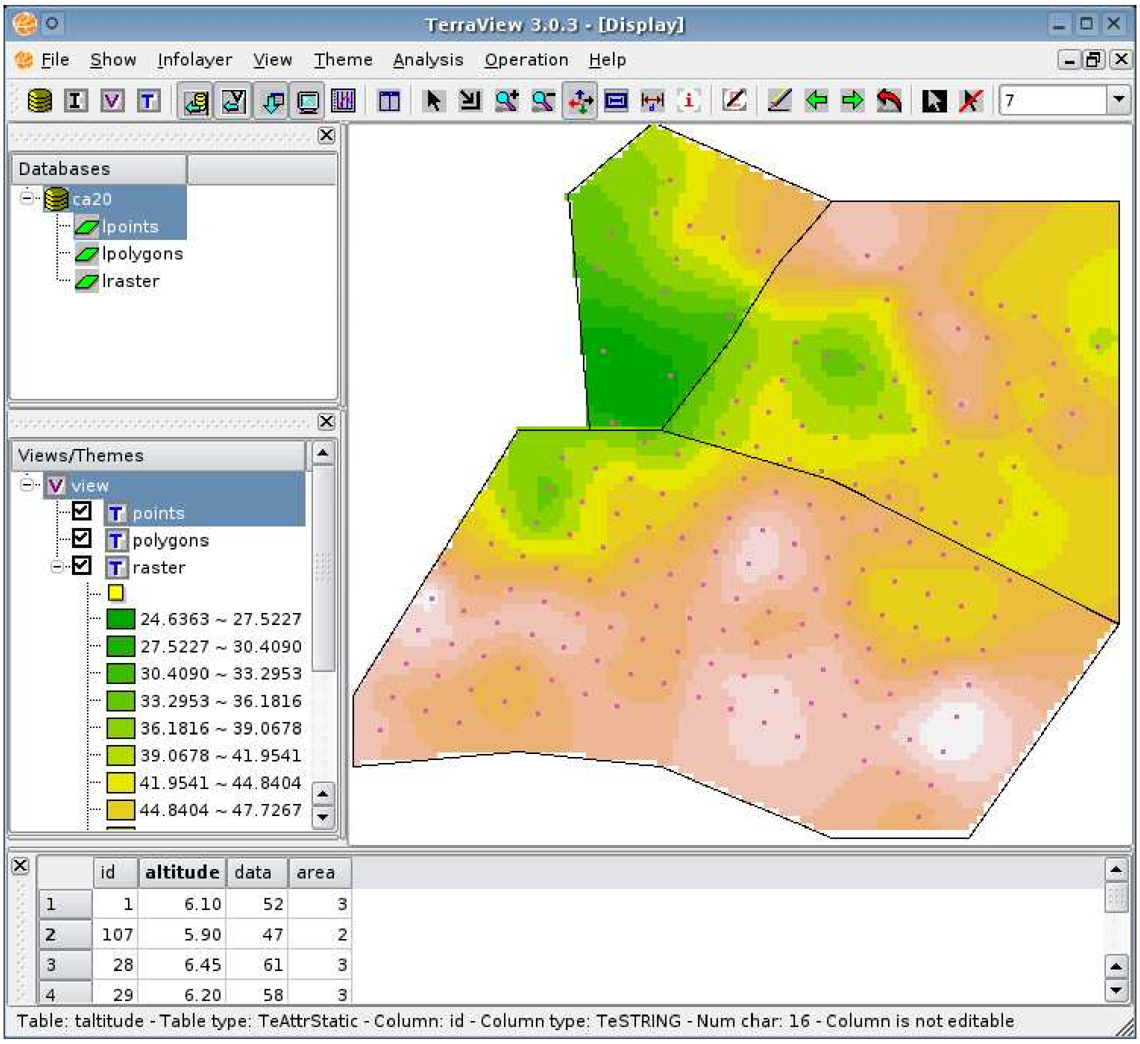
The generated visualization is shown in Figure 5. We can see the static table with the new attribute in the bottom of the Figure.

5. Conclusions and Future Work

This article discuss the need for integration between statistical softwares and GIS. Joining this two technologies strengthens both sides and increases their individual and common effectiveness. The integration can be implemented using full integration or through mechanisms of loose coupling and close coupling.

A new integration model based in a geographic database is implemented. The access is executed through TerraLib with TerraLib components being used directly as R objects. This way we can access the database in a easy manner and execute geoprocessing functions while reading data from the database. The integration increases the robustness of both technologies allowing for execution of more complex tasks and improving the quality of the results.

Being TerraLib an ongoing project, as soon as it presents new geoprocessing models and algorithms access to them can be implemented in aRT keeping the package users up-to-dated with the advances in geoprocessing.



	id	altitude	data	area
1	1	6.10	52	3
2	107	5.90	47	2
3	28	6.45	61	3
4	29	6.20	58	3

Table: taltitude - Table type: TeAttrStatic - Column: id - Column type: TeSTRING - Num char: 16 - Column is not editable

Figure 5. Visualization of the result using TerraView

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 developed at LEG/DEST/UFPR, with support of TerraLib project developers from DPI/INPE. The authors want to express their gratitude to the TerraLib developers Lúbia Vinhas and Karine Reis for all their help throughout the development of this project.

References

Anselin, L. (1992). *SpaceStat tutorial: a workbook for using SpaceStat in the analysis of spatial data*. NCGIA (National Center for Geographic Information and Analysis), Santa Barbara.

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

Bao, S., Anselin, L., Martin, D., et al. (2000). Seamless integration of spatial statistics and GIS: The S-PLUS for ArcView and the S+Grassland links. *Journal of Geographical Systems*, 2(3).

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., Souza, R., Freitas, U., et al. (1996). SPRING: Integrating remote sensing and GIS with object-oriented data modelling. 15(6):13–22.

Capeche, C. e. (1997). Caracterização pedológica da fazenda angra - pesagro/rio - estação experimental de campos (rj). In *Informação, globalização, uso do solo*, volume 26, Rio de Janeiro. Congresso Brasileiro de Ciência do Solo, Embrapa/SBCS.

Casanova, M., Camara, G., Davis, C., Vinhas, L., and Queiroz, G. (2005). *Bancos de Dados Geográficos*. Editora MundoGEO, Curitiba.

Druck, S., Carvalho, M. S., Camara, G., et al. (2004). *Análise Espacial de Dados Geográficos*. EMBRAPA, Brasília.

Fischer, M. M., Scholten, H. J., and Unwin, D. (1996). Geographic information systems, spatial data analysis and spatial modelling: an introduction. In *M. M. Fischer, H. J. Scholten and D. Unwin Spatial analysis perspectives in GIS*. Taylor and Francis.

Goodchild, M., Haining, R., and Wise, S. (1992). Integrating GIS and spatial data-analysis - problems and possibilities. *International Journal of Geographical Information Systems*, 6(5):407–423.

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

Krivoruchko, K. (2003). Using spatial statistics in GIS. In *International Congress on Modelling and Simulation*, Townsville, Australia.

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.

Vinhas, L., de Queiroz, G. R., Ferreira, K., et al. (2002). Programação genérica aplicada a algoritmos geográficos. In *IV Simpósio Brasileiro de Geoinformática*, Caxambu – Brazil.

Zhao, L. (2002). The integration geographical information systems and multicriteria decision making models for the analysis of branch bank closures. Technical report, Faculty of the Built Environment. Sydney, Australia, University of New South Wales.