Software Integration for Multivariate Exploratory Spatial Data Analysis

Jürgen Symanzik[1], Deborah F. Swayne[2], Duncan Temple Lang[3], Dianne Cook[4]

[1]Utah State University, Department of Mathematics & Statistics, Logan, UT 84322–3900
symanzik@sunfs.math.usu.edu
[2]AT&T Labs – Research, Statistics Research Department
dfs@research.att.com
[3]Bell Labs, Lucent Technologies, Statistics & Data Mining Research Department
duncan@research.bell-labs.com
[4]Iowa State University, Department of Statistics
dicook@iastate.edu

Abstract

This paper describes a decade's worth of evolution of integrating software to support exploratory spatial data analysis (ESDA) where there are multiple measured attributes. The multivariate graphics tools we use are XGobi, and more recently, GGobi. The paper is divided into two parts. In the first part, we review early experiments in software linking for ESDA, which used XGobi, different Geographic Information Systems (GIS), and the statistical analysis packages S and XploRe. We discuss applications, capabilities and deficits of these existing links. In the second part we describe GGobi, the descendant of XGobi, which was designed with software integration as a fundamental goal. GGobi lends itself to broader possibilities for linking with other software for richer ESDA.

Keywords

Dynamic Statistical Graphics, Interactive Graphics, Geographic Information System, Visual Data Mining, Multivariate Data.

1 Introduction

Doing ESDA well involves three components: (1) GIS tools for map drawing and operations on spatial information, (2) statistical data visualization software for generating plots of the attribute information, especially to explore multivariate structure, and (3) a statistical analysis package to compute spatial models and perform quick restructuring of data and attributes. Excellent software has been developed for each of these components, but no single tool or package can do the jobs of all three. Integrating software for ESDA makes sense.

The integration of GIS tools and statistical graphics tools depends on the ability to link an element of a display of attribute information to its geographic coordinate, displayed on a map or as a terrain surface. Useful statistical graphics include histograms, scatterplots, parallel coordinate plots, and scatterplot matrices.

Combining statistical plots with geography has been discussed widely in the last 10 to 15 years. Monmonier (1988) described a conceptual framework for geographical representations in statistical graphics and introduces the term *geographic brushing* for the interactive "painting" of map views. Unwin et al. (1990) developed a software system with both map drawing capabilities and statistical graphics. Many software solutions (summarized in citetSCLMM00) have been developed for exploring multivariate spatially referenced data in recent years, but it remains true that no single package can do everything well.

This paper describes an approach to software integration which relies on linking independent tools in a variety of different ways. This approach allows software designers and developers to concentrate on what they do best: geographers create the GIS's, statistical data visualization experts create the statistical graphics software, and experts in languages and algorithms create the analysis environments. Indeed, all these tools already exist, and are constantly being refined and extended. It makes little sense for each set of specialists to attempt to replicate the work of the others, as replication will usually be inferior and always lag behind the original.

We have concentrated on linking software components that enable multivariate graphics to be connected to map views and statistical analysis tools. Few software packages offer any connnection to other systems, and even fewer provide a framework for connecting to arbitrary systems. When communication tools are provided they have typically been added late in the development and are somewhat inefficient and awkward.

In general, linking software is a difficult task. In section 2, we provide case studies of different efforts to link GIS's (ArcView and VirGIS), visualization (XGobi) and statistical analysis packages (S and XploRe). As prototypes, these showed the potential of such linking for ESDA. We describe what the links contributed, some of their shortcomings and the technical challenges they posed. In summary, the links were difficult to develop and hard to maintain, and illustrate that these basic components need to be designed from the outset to support interoperability between each other, and other components which may not yet exist. In section 3, we focus on a collection of different approaches to software integration that we have recently developed in GGobi, a modern descendant of XGobi, that overcome some of the difficulties discovered in section 2. We hope that these will provide the necessary technical infrastructure to allow the kinds of ESDA applications described in 2 to be implemented both easily and efficiently.

2 XGobi and its Links

2.1 Overview of XGobi

XGobi (Swayne et al., 1998) is a data visualization system with interactive and dynamic methods for the manipulation of views of data. A view of an XGobi window can be seen in Figure 2.1. It offers 2D displays of projections of points and lines in high–dimensional spaces, as well as parallel coordinate displays (Inselberg, 1985; Wegman, 1990). Projection tools include dotplots and average shifted histograms (Scott, 1985) of single variables, scatterplots of pairs of variables, 3D data rotations, high dimensional rotations called "grand tours" (Asimov, 1985), and interactive projection pursuit (Huber, 1985). Views of the data can be panned and zoomed. Points can be labeled and brushed with glyphs and colors. Lines can be edited and colored. Several XGobi processes can be run simultaneously and linked for labeling, brushing, and sharing of projections. Missing data are accommodated and their patterns can be examined; multiple imputations can be given to XGobi for rapid visual diagnostics.

XGobi is implemented in the X Window System™, so it runs on any UNIX®system, and runs under Microsoft Windows™or the Macintosh®operating system if an X emulator is used. XGobi has been publicly available since the early 1990's (Swayne et al., 1991), and has been widely used

X Window System is a trademark of MIT.
UNIX is a registered trademark of The Open Group.
Microsoft Windows is a trademark of Microsoft, Inc.
Macintosh is a registered trademark of Apple Computer, Inc.

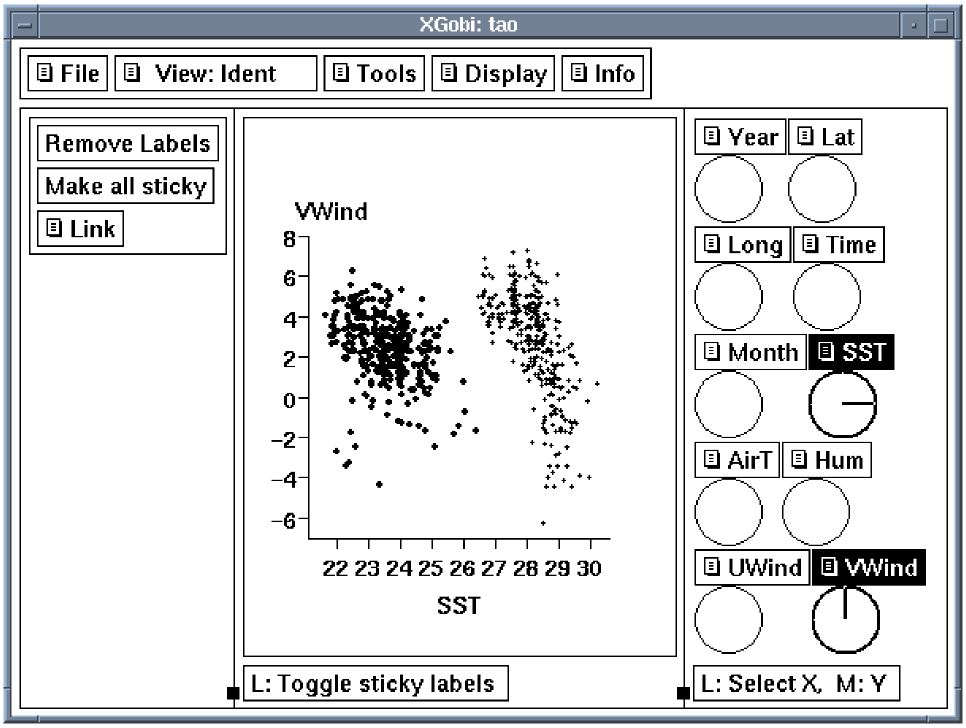

Figure 1: XGobi, showing a scatterplot. The data shown is taken from NOAA Tropical Ocean Atmosphere project, in which measurements are made at buoys in the Pacific Ocean. XGobi is a rich statistical data visualization package, but lacks cartographic display methods and modeling and analysis tools. It is often used in conjunction with other software.

by data analysts in many fields. XGobi can be freely downloaded from `http://www.research.att.com/areas/stat/xgobi/`.

In the remainder of this section, the following projects which linked XGobi to other software will be described, along with the lessons we learned:

- S/XGobi: XGobi's communication with the S statistics environment; one method uses files to exchange data, and the other was our first attempt to use UNIX interprocess communication.
- ArcView/XGobi/XploRe: Links among ArcView, the XGobi visualization software and the XploRe statistics environment using RPC.
- XGobi/VirGIS: Links between XGobi and the experimental virtual reality GIS called VirGIS, again using RPC.
- XGobi/RA_3DIO: Communication between XGobi and RA_3DIO, a virtual reality framework for the design and management of mobile phone networks; using DCE.

2.2 S/XGobi

The designers of XGobi had always intended that it be used in conjunction with other software, especially analytical software such as the S language and statistics environment (Becker et al., 1988). (S is currently available either commercially, as S-Plus, or as an Open Source implementation called R (Gentleman and Ihaka, 1997).) They wanted to be able to reuse standard statistical functionality and also allow users to provide their own methodology, choices for parameters, missing values, etc.

There exists an S (i.e. R or S-Plus) function, distributed with the XGobi software, that allows an S user to launch an XGobi process given S objects as arguments. That function is beautifully simple: the S objects are written out as ASCII files, and a system call executes XGobi with those files as arguments. An XGobi process launched in this way has very limited ability to create S objects directly: after brushing, for example, the vector of point colors can be saved as a file in the S format. The S process has no ability to communicate further with XGobi.

The XGobi authors occasionally explored other approaches that would extend this unsatisfactory relationship. As early as 1991, we used interprocess communication to maintain a live connection between XGobi and S. One of the applications would draw a clustering tree in S, allow the user to click on it to cut the tree and immediately set the point colors in XGobi to show the result. It relied on a second program, also written in C, to gather input from the user and to manage the interprocess communication. It was necessary to assemble each S language command, ship it to S, read back the result and respond accordingly. To make this foolproof, it would have been necessary for XGobi to be fully able to parse S commands and handle errors. Because this was such a daunting task and one that would require continual updates to keep pace with changes to the S language, work on this model was discontinued after the first prototype.

2.3 ArcView/XGobi/XploRe

Our initial foray in the integration of GIS's and multivariate data visualization software was to develop a unidirectional link between the GIS ARC/INFO and XGobi (Cook et al., 1994; Symanzik et al., 1994). We then extended the work using ArcView 2.0, developing a bidirectional link with XGobi (Cook et al., 1996, 1997; Macedo et al., 2000; Symanzik et al., 2000a,b).

Later, the link was extended to include the statistical computing environment XploRe (Härdle et al., 1995), resulting in the ArcView/XGobi/XploRe environment (Symanzik et al., 1998a; Lewin-Koh et al., 1999). Few modifications were required on the ArcView/XGobi side to support this extension while XploRe had to undergo considerable changes since it did not yet support the required communication technology.

Further information on the ArcView/XGobi/XploRe environment can be found at `http://www.math.usu.edu/~symanzik/axx/axx2000/`.

2.3.1 Applications

The ArcView/XGobi link supports one–to–one connections between ArcView and XGobi such as linking geographic location to (multivariate) attribute values and to empirical Spatial Cumulative Distribution Function (SCDF) values. In addition, we explored one–to–two and two–to–one linking to connect variogram–cloud plots (Cressie, 1984; Haslett et al., 1991), spatially lagged scatterplots (Cressie, 1984), and multivariate variogram–cloud plots (Majure and Cressie, 1998) to geographic coordinates. (These plots help the analyst identify spatial dependence patterns amongst the attributes.)

The linking between ArcView and XGobi allows us to simultaneously display spatial locations and concomitant geographic variables within the GIS while visualizing and exploring the corresponding data space within XGobi. The usefulness of the link has been highlighted for several different applications such as satellite imagery, forest health monitoring, precipitation data, and atmospheric science data described in the previously cited main references on ArcView/XGobi. There also exist videos that demonstrate the use of the ArcView/XGobi link (Majure et al., 1995, 1996; Symanzik et al., 1995).

2.3.2 Technical Background

The ArcView/XGobi/Xplore environment uses Remote Procedure Calls (RPCs), the only Inter–Process Communication (IPC) method supported by ArcView's Avenue programming language when this link was first developed. The use of RPCs is a programming technique in which a process on the local system (*client*) invokes a procedure on a remote system (*server*).

ArcView was modified for this application using its built–in Avenue programming language. All of the default ArcView functionality is available, with the addition of several operations that are necessary to handle the link. Specifically, ArcView was modified to do the following: initiate an RPC server and client, initiate and pass data to the XGobi process, brush locations in the map view and instruct XGobi to brush the corresponding points, and process requests from XGobi to brush locations. Detailed technical information on the implementation of the communication mechanism between ArcView, XGobi, and XploRe can be found in Symanzik et al. (2000a).

2.3.3 Achievements and Deficits

The ArcView/XGobi/XploRe environment has been widely used for our own research, in the classroom setting, and by our graduate students. Based on e–mail received over the last few years, many people worldwide have downloaded the sources and have used the combined environment for their own data. It runs on a variety of UNIX platforms, but it does not run under Microsoft Windows, and thus excludes a large user community.

One of the current limitations of the ArcView/XGobi link is the number of points that can be handled. The RPC method requires strings to be passed between ArcView and XGobi, and there is a large overhead in encoding the data, communicating the strings and finally decoding them. With large data, this inefficiency causes problems for linked brushing.

Currently, the ArcView/XGobi link does not allow linking between ArcView and multiple copies of XGobi representing different data, for instance, an SCDF plot and a variogram–cloud plot. This reflects a limitation of XGobi: the rules for linking between XGobi processes are primitive, and it's difficult to link a point in one display to a line or a set of points in another. To address this, either XGobi's linking rules could be made more flexible, or we could add a linking manager to ArcView, which would mediate the linking process and communicate differently with each XGobi process.

The ArcView/XGobi link focuses on ESDA and uses few of the other features of the underlying GIS. ArcView is mostly used to store the data and display additional geographic features that relate to the statistical data of interest. Currently, the geographic brushing in ArcView exists only in what Monmonier (1989) defines as its simplest form: "use a mouse to highlight specific areas on the map." However, another possible extension on the ArcView side could have facilitated more complex types of geographic brushing, e.g., brushing statements of the form "brush all spatial locations that are at most 10 km away from the next city boundary and have no major road within a distance of 1 km".

This project is unlikely to continue. Updates to both XGobi and ArcView have made it difficult to maintain this link across different version combinations. Indeed, ESRI has announced that it will not support Avenue in future releases of ArcView. Given these circumstances, none of the desirable features just listed have been added to the ArcView/XGobi/XploRe environment nor has any major attempt been made to repair the latest versions of XGobi.

2.4 XGobi/XploRe/ViRGIS

The growing interest in GIS's has led to increasingly complex applications that use ever larger and more varied datasets. To be useful to a non–expert user, a GIS should allow the user to explore any virtual world of interest.

The user might, for instance, look around the virtual Alps to decide where to take a skiing vacation (Szabo et al., 1995). Visualization of the scene and all relevant data at the same time is crucial for the ease of use of the system as a whole (Hearnshaw and Unwin, 1994). Such a GIS interface could in fact be one of the prime examples of a post–WIMP user interface (WIMP stands for Windows, Icons, Menus, and a Pointing device) (Coyne, 1995; van Dam, 1997).

We have been interested in linking XGobi and XploRe to software that supports such new features; in particular, we chose the experimental Virtual Reality GIS ViRGIS (Pajarola, 1998; Pajarola et al., 1998). ViRGIS maintains 3D terrain data in vector form (such as surface triangulations), raster data (such as satellite images), and non–geometric data (such as population counts of cities). It allows a user to move through the scene in real–time by means of a standard input device such as a mouse, and to interact with the data in the GIS. Thus far, the ViRGIS interface is a *desktop VR* or video user interface in the classification of Agnew and Kellerman (1996).

The combination of ViRGIS with XGobi/XploRe allowed us to implement and explore new functionality that is conceptually not possible in the ArcView/XGobi/XploRe environment. Details on the XGobi/XploRe/ViRGIS environment can be found in Symanzik et al. (1998b).

2.4.1 Applications

Using data accessible through ViRGIS, we can activate XGobi and XploRe (via XGobi) from within ViRGIS and pass the data into these two packages. This allows us to conduct a graphical exploration of the data in XGobi and a more detailed statistical analysis in XploRe.

The main feature of the XGobi/XploRe/ViRGIS environment is the linked brushing option that combines quite heterogeneous windows and displays. We described in the previous section the use of linked brushing between XGobi and XploRe; in addition, we can also select points from a textual representation in ViRGIS. This textual representation and any linked XGobi window and XploRe display are also linked to the ViRGIS Inventor view of the 3D terrain. Thus, brushing in one window results in all linked windows being updated. As an example, we looked at a data set from 3019 Swiss cities that contained demographics such as the language spoken in that city, ZIP, and population.

2.4.2 Technical Background

The inter–process communication used in this environment is based on the same RPC mechanism as the ArcView/XGobi/XploRe environment. For the implementation of the RPC mechanism on the ViRGIS side, we closely followed an early version of Symanzik et al. (2000a). No major modifications were required on the XGobi and XploRe side.

2.4.3 Achievements and Deficiencies

The XGobi/XploRe/ViRGIS environment was purely experimental. Several interesting problems came up that should be addressed in advance when linking future heterogeneous (2D and 3D) applications. One question was how to translate brushing symbols when linking 2D and 3D displays. Naturally, a circle in 2D relates to a sphere in 3D. But which 3D–object relates to a "+" and "×"? We didn't find an appealing solution. Similarly, the colors of the different tools should correspond: Each package should also use the same set of colors (including the background and annotation colors) so that the results of color brushing can be easily interpreted.

Since ViRGIS was only available for SGI workstations, the potential number of users of the XGobi/XploRe/ViRGIS environment was very limited. Since further development on the ViRGIS side was halted in favor of $\mathrm{RA_3DIO}$, no further development was made on the XGobi/XploRe/ViRGIS environment.

2.5 XGobi/$\mathrm{RA_3DIO}$

$\mathrm{RA_3DIO}$ is a virtual reality framework for the design and management of mobile phone networks and the optimization of antenna positions. $\mathrm{RA_3DIO}$ is based on the research prototype WorldView (Beck et al., 1998). It was implemented to visualize and explore virtual terrains and terrain related themes, especially electro-magnetic wave propagation of transmitters in rural and suburban areas. Microsoft Windows was chosen as the platform for $\mathrm{RA_3DIO}$ because of its wide distribution and its many standard components. Therefore, it runs both on fast graphic workstations and on small Notebook computers. In the remainder of this section, we describe a bidirectional link we developed between RA_3DIO and XGobi (Schneider et al., 2000).

2.5.1 Applications

RA_3DIO can provide a large amount of data for visual exploration in XGobi, e.g., terrain data (polyhedral triangulated data) on its own or spatial data objects handled in RA_3DIO, such as cities with their parameters (city name, spoken language, number of inhabitants, city area, etc.) or an antenna data set (position, direction, height above ground, power, antenna type, carrier frequency, etc.). In one example, the linked environment was used to analyze radiation emissions.

2.5.2 Technical Background

Since RA_3DIO runs under Microsoft Windows and XGobi runs under the X Window System, the particular RPC communication described in the previous sections could not be used for this link. However, there exist several other competing middleware standards. We used the Distributed Computing Environment (DCE) from OSF (Open Software Foundation, now The Open Group).

XGobi had to undergo many changes. The biggest problem for XGobi was the synchronization with the DCE–RPCs. XGobi is a single–threaded application, so only one thread can access internal data structures at a time. In DCE, RPCs can occur at any time. We had to provide synchronization between each RPC thread and the main XGobi thread.

2.5.3 Achievements and Deficits

The link between RA_3DIO and XGobi demonstrated that we could link applications across heterogeneous hardware platforms.

One of the deficiencies of this link is the speed. For a variety of reasons, the "send data" protocol of the link was originally not designed for dynamic data updates. As a result, it is necessary to copy the entire data matrix from RA_3DIO to XGobi even if only one entry has been changed. This results in a superfluous and excessive network load. Additionally, it introduces considerable overhead not just on transmitting the data but also in processing it on the XGobi side of the link. The result is a latency in responding to the user's actions. An interim solution to handle this problem could have been a data mirror residing on the same machine as the XGobi client. A more general solution is to design the communication so that it also supports sending only the changes in the data.

3 GGobi

3.1 Overview of GGobi

GGobi (Swayne et al., 2002) is a direct descendant of XGobi, but it has been thoroughly redesigned. For this paper, the most significant change is GGobi's relationship to other software. These new features will be described in the next section. Readers may also be interested in the following brief description of the changes in GGobi's appearance, portability, and data format, when compared to XGobi. GGobi can be freely downloaded from `http://www.ggobi.org/`.

- GGobi's appearance: GGobi looks quite unlike XGobi at first glance, because GGobi uses a newer graphical toolkit called `GTK+` (`http://www.gtk.org`), with a more contemporary look and feel and a larger set of user interface components. The second change an XGobi user will spot is the separation of the plot window from the control panel: With XGobi, there is in general a single plot per process; to look at multiple views of the same data, we have to launch multiple XGobi processes. In contrast, a single GGobi session can support multiple

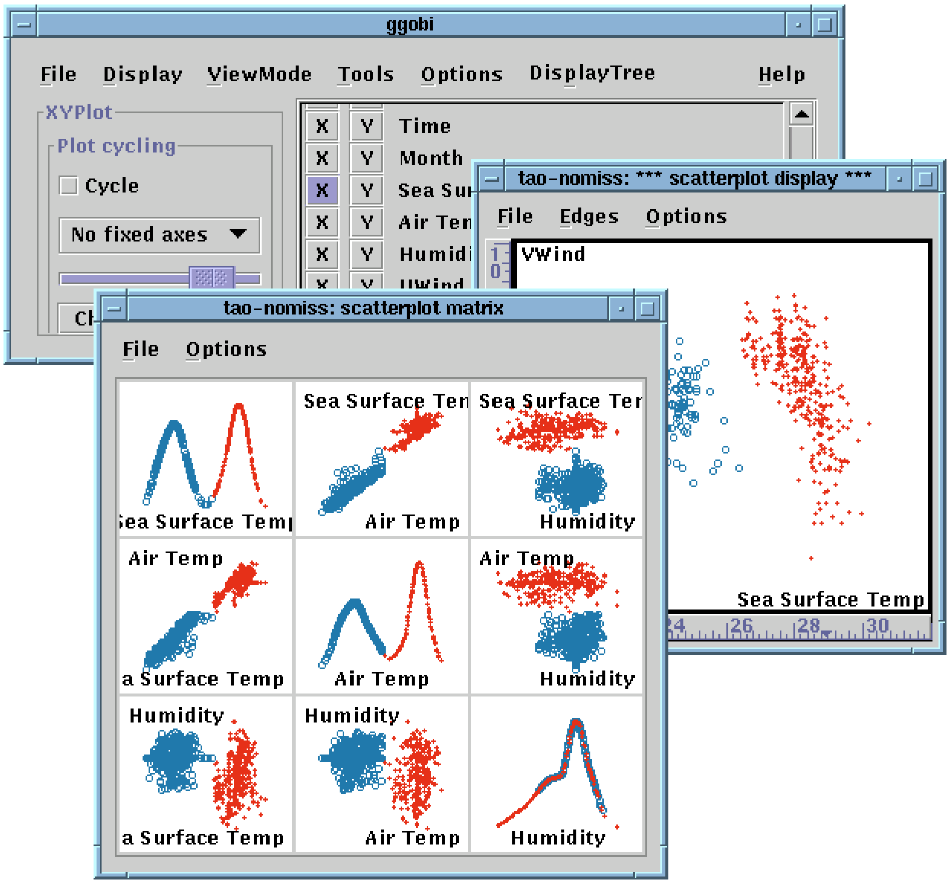

Figure 2: GGobi, a descendant of XGobi, showing the same scatterplot as in Figure 2.1, and adding a linked scatterplot matrix. GGobi has many differences from XGobi, both visible (multiple displays are possible, and the toolkit has a more modern look) and invisible (it has been designed to facilitate software integration).

plots of various types: scatterplots, parallel coordinate plots, scatterplot matrices, and time series plots have been implemented thus far.

Other changes in GGobi's appearance and repertoire of tools include an interactive color lookup table manager, the ability to add variables on the fly, and a new interface for view scaling (panning and zooming).

- Portability: A major advantage of using the new toolkit (`GTK+`) is portability. It originates in the Linux®community, but it has been ported to Microsoft Windows™and Macintosh®OS X. To run the older XGobi on a machine running Windows, it is first necessary to install an X Window System server, but GGobi runs directly under Windows.

- GGobi's data format: GGobi's data format has been extended significantly from that of XGobi. To describe a set of data for XGobi, we have to create a set of files with a common base name,

Linux is a registered trademark of Linus Torvalds.
Microsoft Windows is a trademark of Microsoft, Inc.
Macintosh is a registered trademark of Apple Computer, Inc.

with the data in one file, and other files for the colors, labels, and so on. GGobi continues to support this scheme in a limited way, but its new format uses a single file in XML™, the Extensible Markup Language, which is emerging as a standard language for specifying structured document formats and data exchange.

The use of a single file aids consistency of the different elements of the input, making it easier to validate and maintain. An XML document looks similar to an HTML document, but it allows us to introduce new markup elements. The use of XML in GGobi allows complex characteristics and relationships in data to be specified. For example, multiple datasets can be entered in a single XML file, and specifications can be included for linking them. Using the XML software, GGobi can read compressed files and can read files over the network.

3.2 Technical Background

While GGobi is a stand–alone application, it has been designed and constructed as a programming library and provides direct manipulation, dynamic visualization functionality that can be embedded within other applications. It has a large, but still evolving, Application Programming Interface (API) which developers can use to integrate the GGobi functionality with other code. In this section we discuss GGobi's interoperability philosophy and specifically the different ways by which one can integrate GGobi with other software. At the highest level, there are three different approaches to connecting GGobi with other software.

Embedding GGobi within other Applications: In this approach, we treat GGobi as a programming library and allow its data structures to be compiled into other customized applications. When GGobi is embedded in this way, it can be controlled using programming interfaces from other languages such as Java, Perl, Python and S. When the language is a scripting language, the programming interface includes the use of the language's interactive programming facilities. These interfaces are called *language bindings*.

Extending GGobi: The use of modular plugins allows one to dynamically load code into a running GGobi. Using this approach, programmers can add functionality to GGobi without having to dig deeply into the code, and they can share their extensions easily with others.

Distributed/Remote Access: The client/server architecture allows one to create GGobi as a server process offering a variety of methods to control, query and modify the session. Other applications can invoke these methods, even across different machines. This approach has roughly the same flavor as the linking methods described in Section 2, but CORBA and DCOM are newer, higher-level and more powerful tools than RPC and DCE.

The first two approaches allow the two (or more) pieces of software to be running in the same process, with direct access to shared memory. This makes the communication very fast and flexible. The distributed approach involves inter-process communication and allows heterogeneous hardware and configurations to be used. Different situations will use different approaches, and a single approach is unlikely to work well for all contexts.

3.2.1 Language Bindings

While the API allows GGobi to be integrated with arbitrary applications that can access compiled/object code, we have focused on embedding GGobi within a particular class of applications, specifically interpreted languages. We have developed a complete set of language bindings to GGobi for S (both S–Plus and R), and also the basic functionality and framework for both Perl and Python. These interactive, command–line interfaces to GGobi's functionality allow us to drive and direct the GGobi session in a very different style than an entirely GUI-based (Graphical User Interface) approach.

In addition to providing two simultaneous interfaces (GUI and command–line), this architecture allows the functionality of both the statistical and the visualization environments to be used relatively seamlessly. We can start by reading and processing the data using a powerful and general programming language, creating the appropriate subsets of the observations, transformations of the variables and statistical summaries of the dataset. Then we can display the data using GGobi along with more static graphics displays provided by S. The user can then explore the data both by direct graphical manipulation such as brushing and identification using GGobi's controls, and at the same time via programmatic commands in the programming language. The results from user actions in the GUI can then be queried within S to synchronize the two views (GGobi and S) of the dataset. This is a convenient way to subset graphically and then perform operations such as smoothing or prediction on the subset in the programming language. In the near future, we plan to connect the R and GGobi graphics systems so that we can display plots from both systems on the same canvas, and use R to augment and annotate GGobi's displays.

Thus, the approach of language bindings has many advantages. First, it means that we, as developers, do not have to re–implement (often half–heartedly) all the basic functionality provided in other systems. Second, leaving functionality out of the GUI if it is better suited to a programmatic interface typically simplifies the GUI and provides a more user–friendly and consistent environment for the user. By providing bindings to well–known and commonly used languages such as S, Perl and Python, we avoid inventing *yet another language* that is used only in a single application, GGobi. In the near future, we hope to make some internal changes to the GGobi code that will facilitate automating the creation of language bindings, making this style of interoperability richer for the user and simpler for the developers.

3.2.2 GGobi & `GTK+` Events

The language bindings described above allow the user to manually switch control and focus between GGobi and, for example, S. These bindings allow S to *automatically* respond to different user actions. An S user can associate S functions with different user actions in the GGobi interface, such as identifying a point with the mouse, or moving the brushing region to include or exclude a point. When these actions take place in the GUI, the corresponding S function is invoked and this can update other plots, display summaries on the console, bring up new GUIs, etc.

The user actions that can be customized include low-level events such as button clicks, dragging the mouse across a plot and other toolkit–level actions. Additionally, GGobi defines higher level events such as a point being added or removed from the brushing region, record identification, and other more infrastructure type events such as the loading of a new dataset or the creation of a new display. It is relatively easy to dynamically register an event handler that provides additional functionality and unregister this event handler later in the session. Using this facility, we can customize and extend the GGobi GUI using a high–level language such as S, rather than changing core C code. This modularity or overlaying of functionality makes it simpler for the programmer and also guards against corrupting GGobi. Perhaps most important, this allows us to implement different and novel linking schemes at a very high level.

The RGtk package is an R programming interface to the `GTK+` libraries. It provides a simple and uniform way to associate S functions with `GTK+` events and makes the customization of the GGobi GUI via the S–language possible. Additionally, it also allows us to extend GGobi by building simpler or more customized interfaces. We can programmatically embed GGobi plots and displays as components within other GUIs.

3.2.3 Gnumeric Spreadsheet Plugin

In addition to the language bindings for S, Perl, and Python, we have also embedded GGobi into Gnumeric, the Gnome spreadsheet, using Gnumeric's plugin mechanism. This provides access to GGobi's functionality directly from within the spreadsheet/workbook. To create a GGobi instance, the user of the spreadsheet can specify a range of cells, either manually or interactively. This initiates the usual GGobi control panel and a plot of the specified data in the Gnumeric spreadsheet. The user can then use GGobi as usual. This is a convenient interface that allows us to process, transform and manage data within the spreadsheet before sending it to GGobi, and it can be more familiar and less intimidating than the command–line or programmatic interfaces described in section 3.2.1.

In addition to being able to send data from Gnumeric to GGobi, the GGobi plugin for Gnumeric can also be used in the opposite direction. We can create a dynamic GGobi instance for a given sheet. In this context, events in GGobi can cause updates in the spreadsheet. For example, when GGobi is in "identify mode" (interactively labeling the point nearest to the mouse cursor), we scroll the spreadsheet so that the row corresponding to that nearby point is visible in the center of the sheet. This allows us to see the values of all the variables for that record.

The GGobi plugin can also report events that occur during brushing, in particular when the "brush" moves so as to include or exclude a point in the brushing region. The dynamic GGobi interface provides a Gnumeric function for a cell in the spreadsheet that checks whether the corresponding point is inside the brushing region and returns a 1 or 0 accordingly. As the brush is moved and resized, the cells in the worksheet are updated. This simple information can be used in dynamic computations within the sheet to compute separate statistics for the records inside and outside the brushing region. For example, we can compute the mean and variance of the brushed subset for different columns by simply creating parallel columns formed by multiplying the values in the original cells by the indicator of whether the record is in the brushing region. As we move the brush around, the cells in the spreadsheet are updated and the new means and variances are displayed. In this way, we can readily create dynamically updated numeric displays and format them in easy–to–read ways that complement the dynamic GGobi linking.

3.2.4 Extending GGobi with Plugins

In the previous subsections, we have discussed how we can extend GGobi by embedding it into other applications and environments. In many cases, however, it is more convenient to work within the GGobi application and simply extend it rather than creating a new application. For example, we might want to add new plot types such as boxplots, conditional plots, or graphs and trees. Alternatively, we might want to provide facilities for reading data from different sources such as a database, or a binary or proprietary file format. We might also want to provide additional tools such as a data editor, alternative printing facilities, data summary display, etc. In principle, since the source for GGobi is publicly available, one can modify the code to add such functionality. However, we have provided a cleaner and more modular way to augment GGobi using *dynamically loadable "plugins"*.

There are currently two basic types of plugins. Input plugins are used to read data from different sources such as databases, binary file formats, etc. Regular plugins are intended to provide additional user–level functionality to GGobi, to be accessed using controls that are added to the GGobi GUI; current examples include a data editor and a graph layout plugin. The architecture allows us to create the plugin by writing a few C routines that implement the plugin's functionality. These different routines are invoked by GGobi at different times in the session, i.e., when the plugin is loaded and unloaded, and when the plugin is instantiated and destroyed for a particular GGobi instance. For example when the data editor plugin is instantiated, it adds a item to the Tools menu by which the user can activate the editor. Input plugins are invoked when trying to load data. All plugin routines can basically do whatever they want when they are called and they have full access to the internals of GGobi. They can utilize the GGobi and `GTK+` event mechanism and typically do not have to deal with many of the internal details of GGobi that are not within the API or described in example plugins.

While the basic language for creating plugins, C, provides complete access to GGobi, it is too low–level for many people. Accordingly, we have created three *meta*– or *language*–plugins which allow us to program GGobi plugins directly using R, Java, or Perl. The developer implements an interface in Java, or extends a Perl class, or implements a collection of S functions, and within these provides the desired functionality of the plugin. The plugins can use the GGobi API and any available bindings for that particular language. They can add entries to menus and other GUI components, and provide actions for these controls without having to notify GGobi directly. They behave just like the low–level C plugins and have a very similar structure. Plugins for other programming languages and scientific computing and data analysis systems (e.g., Visual Basic, Matlab/Octave and XLisp-Stat) can easily be added to this framework.

Each time GGobi is started, it reads an initialization or "rc" file and this can include information about the plugins for that user and system. The initialization file is written in XML and can optionally include elements describing the different plugins of interest and their details, such as a human–readable description for the user, the location of the Dynamically Loadable Library (DLL or shared library) containing the code or the name of the Java/Perl class or R function, the names of the routines corresponding to the different plugin hooks, etc.

The plugin mechanism allows developers to enhance GGobi without having to modify the code. It minimizes the need to re–integrate changes into new GGobi releases. It also makes it reasonably easy to develop new plugins by providing a well–defined sequence of steps. Typically, we can use previously written code by merely adapting it to the necessary plugin interfaces. Again, we hope that the ease with which plugins can be introduced provides a focused framework that encourages people to experiment with new ideas and to add less common functionality that might not be worth building from scratch. Also, we hope that it will provide an easy connection to other projects such as ORCA (Sutherland et al., 2000) and other work being done in Java, Python, and Perl.

3.2.5 Automation & Inter–Process Communication

In some contexts, the approach of embedding GGobi within an application (or vice–versa) is not desirable. Instead, allowing an application to communicate with a separate GGobi process, possibly on a different machine, makes more sense. Both CORBA (the Common Object Request Broker Architecture) and Microsoft's DCOM are middleware which allows exactly this style of interoperability. In this framework, we create a server object, such as a GGobi instance, in one process and advertise a set of methods or operations that are supported by the server object. Client applications can then access the server and invoke these methods to control, query, and modify the server. During the life of the server, it can also act as a client and access functionality in other servers to do its job and can even use functionality and data in the original client. This framework allows, for example, objects such as GGobi plots to be embedded within documents, spreadsheets, presentations, etc. and have them dynamically updated.

We have created a basic CORBA interface for GGobi using Orbit, GNOME's CORBA implementation. Many of the GNOME applications (e.g., Gnumeric and AbiWord) use CORBA to share functionality and exploit the concept of component based software and desktop. If there is interest in this distributed approach, we will augment the interface with more of the functionality from the GGobi API. Additionally, we plan to implement a DCOM interface for Windows.

3.3 Potential for ESDA

The multiple ways that open up the GGobi functionality to other packages provide exciting possibilities for ESDA. It could be possible to embed GGobi into a variety of GIS, or embed a GIS in GGobi. A connection from the S language allows the use of the spatial tools package to process spatial information into objects such as variogram clouds to display in GGobi. As the user probes the variogram cloud the associated pairs of spatial locations are highlighted in a GIS. GGobi's database connection gives it access to a rich collection of distributed and diverse data.

We also look forward to using GGobi as part of our research in virtual environments, which we began in the mid-1990s. The VRGobi system uses a virtual environment as a platform for the 3D dynamic and interactive display of statistical graphics. It re-implements a small set of techniques from XGobi into this new domain (Cook et al., 1998). We were especially interested in the potential of VRGobi for ESDA, since it seems natural to display spatial data in 3D. We project a 3D terrain map along the floor of the virtual environment, and those displays seem quite successful. We experimented with adding floating displays of 3D scatterplot clouds above the terrain map, but we would also like have 2D statistical graphics available for data analysis. Displaying 2D graphics in a 3D environment is not always successful, in part because of the lower resolution of the displays. In our current research, we are experimenting with using the 3D environment for displaying the information that is naturally three-dimensional (like terrain maps or tornados), but using 2D displays for plots of statistical data. GGobi is used on a handheld portable PC physically carried into the 3D environment, and it displays statistical plots which are linked to the 3D physical model or terrain. One program is used to drive both GGobi and the 3D simulation; GGobi is embedded in that program and controlled using its API. An example of a spatial application is that we might use GGobi to monitor the changes in atmospheric conditions, such as temperature, humidity, air pressure, wind directions, as a simulated tornado moves across a simulated landscape.

4 Discussion

In this paper, we have described the evolution of methods that allow us to link other software packages to the two statistical graphics packages XGobi and GGobi. The earlier work connected XGobi to different GIS and statistical systems, first using data exchange via files, and later the more sophisticated RPC mechanism that allowed clients to invoke methods as well as exchange data with a server. From the user's perspective, there were issues with consistency of the user interface across the different applications. The high overhead of the data exchange method slowed the links between systems, causing noticeable delays when dealing with large datasets. From the developer's point of view, the programming of the XGobi link required low-level access to the source code and was difficult to maintain and integrate across different releases of the component software. Error handling across systems was never adequately handled. Our most important observation from this past work is that it's exceedingly difficult to integrate software that wasn't designed with that purpose in mind.

Armed with experiences from XGobi and other projects (including R and Omegahat), we designed GGobi to support interoperability from its inception. The user's need for consistency can be dealt with by embedding GGobi functionality into other `GTK+`-based applications. Problems with speed are significantly reduced by having the linked applications in the same process, accessing data structures in shared memory. GGobi's richer data description using XML allows multiple data matrices to be managed, and more sophisticated linking rules to be defined. One can also extend the linking rules using one of the high-level language bindings such as R, Python or Perl. From the developer's perspective, GGobi offers an array of integration and linking methods to choose from: through its API, via `GTK+` events, by adding plugins, or through inter–process communication. They can choose the method that is most suitable for their project and the one that works best with the other software they want GGobi to communicate with.

It is perhaps the extensibility facility provided by the language bindings that offers the greatest potential for both users and developers. We have seen how S has allowed users in the data analysis domain to extend the core computational system, experiment and refine new methodology and generally perform more cutting–edge and innovative analyses. Similarly, we believe that providing this extensible interface to GGobi will encourage greater experimentation and more exploratory usage, both in the traditional and spatial data analysis domains. Having demonstrated the usefulness of the ArcView/XGobi/XploRe environment over many years, it seems obvious that we can create even more powerful environments for ESDA using GGobi on a variety of platforms.

References

Agnew, P. W., Kellerman, A. S., 1996. Distributed Multimedia — Technologies, Applications, and Opportunities in the Digital Information Industry: A Guide for Uers and Providers. ACM Press and Addison–Wesley, Reading, MA.

Asimov, D., 1985. The Grand Tour: A Tool for Viewing Multidimensional Data. SIAM Journal on Scientific and Statistical Computing 6 (1), 128–143.

Beck, M., Eidenbenz, S., Stamm, C., Stucki, P., Widmayer, P., 1998. A Prototype System for Light Propagation in Terrains. In: Wolter, F. E., Patrikalakis, N. M. (Eds.), Proceedings of Computer Graphics International. IEEE Computer Society, pp. 103–106.

Becker, R., Chambers, J., Wilks, A., 1988. The New S Language — A Programming Environment for Data Analysis and Graphics. Wadsworth and Brooks/Cole, Pacific Grove, CA.

Cook, D., Cressie, N., Majure, J., Symanzik, J., 1994. Some Dynamic Graphics for Spatial Data (with Multiple Attributes) in a GIS. In: Dutter, R., Grossmann, W. (Eds.), COMPSTAT 1994: Proceedings in Computational Statistics. Physica–Verlag, Heidelberg, pp. 105–119.

Cook, D., Cruz-Neira, C., Kohlmeyer, B. D., Lechner, U., Lewin, N., Nelson, L., Olsen, A., Pierson, S., Symanzik, J., 1998. Exploring Environmental Data in a Highly Immersive Virtual Reality Environment. Environmental Monitoring and Assessment 51 (1/2), 441–450.

Cook, D., Majure, J. J., Symanzik, J., Cressie, N., 1996. Dynamic Graphics in a GIS: Exploring and Analyzing Multivariate Spatial Data Using Linked Software. Computational Statistics: Special Issue on Computeraided Analysis of Spatial Data 11 (4), 467–480.

Cook, D., Symanzik, J., Majure, J. J., Cressie, N., 1997. Dynamic Graphics in a GIS: More Examples Using Linked Software. Computers and Geosciences: Special Issue on Exploratory Cartographic Visualization 23 (4), 371–385, paper, CD, and `http://www.elsevier.nl/locate/cgvis`.

Coyne, R., 1995. Designing Information Technology in the Postmodern Age. The MIT Press, Cambridge, MA.

Cressie, N., 1984. Towards Resistant Geostatistics. In: Verly, G., David, M., Journel, A. G., Marechal, A. (Eds.), Geostatistics for Natural Resources Characterization, Part 1. Reidel, Dordrecht, pp. 21–44.

Gentleman, R., Ihaka, R., 1997. The R Language. Computing Science and Statistics 28, 326–330.

Härdle, W., Klinke, S., Turlach, B. A., 1995. XploRe: An Interactive Statistical Computing Environment. Springer, New York, NY.

Haslett, J., Bradley, R., Craig, P., Unwin, A., Wills, G., 1991. Dynamic Graphics for Exploring Spatial Data with Application to Locating Global and Local Anomalies. The American Statistician 45 (3), 234–242.

Hearnshaw, H. M., Unwin, D. J. (Eds.), 1994. Visualization in Geographical Information Systems. John Wiley & Sons, Chichester, UK.

Huber, P. J., 1985. Projection Pursuit (with Discussion). Annals of Statistics 13, 435–525.

Inselberg, A., 1985. The Plane with Parallel Coordinates. The Visual Computer 1, 69–91.

Lewin-Koh, N. J., Symanzik, J., Cook, D., 1999. Dynamic Linking of ArcView, XGobi and XploRe for Multivariate Spatial Data: Linked Brushing for Points, Polygons, and Lines. In: Proceedings of the 20th Asian Conference on Remote Sensing, Hong Kong, China, November 22-25, 1999, Volume 1. Joint Laboratory for Geo Information Science of The Chinese Academy of Sciences and The Chinese University of Hong Kong (JLGIS), pp. 575–580.

Macedo, M., Cook, D., Brown, T. J., 2000. Visual Data Mining in Atmospheric Science Data. Data Mining and Knowledge Discovery 4 (1), 69–80.

Majure, J. J., Cook, D., Cressie, N., Kaiser, M., Lahiri, S., Symanzik, J., 1995. Spatial CDF Estimation and Visualization with Applications to Forest Health Monitoring. ASA Statistical Graphics Video Lending Library (contact: dj@research.bell-labs.com).

Majure, J. J., Cook, D., Symanzik, J., Megretskaia, I., 1996. An Interactive Environment for the Graphical Analysis of Spatial Data. ASA Statistical Graphics Video Lending Library (contact: dj@research.bell-labs.com).

Majure, J. J., Cressie, N., 1998. Dynamic Graphics for Exploring Spatial Dependence in Multivariate Spatial Data. Geographical Systems 4 (2), 131–158.

Monmonier, M., 1988. Geographical Representations in Statistical Graphics: A Conceptual Framework. In: 1988 Proceedings of the Section on Statistical Graphics. American Statistical Association, Alexandria, VA, pp. 1–10.

Monmonier, M., 1989. Geographic Brushing: Enhancing Exploratory Analysis of the Scatterplot Matrix. Geographical Analysis 21 (1), 81–84.

Pajarola, R., Ohler, T., Stucki, P., Szabo, K., Widmayer, P., 1998. The Alps at your Fingertips: Virtual Reality and Geoinformation Systems. In: Proceedings 14th International Conference on Data Engineering, ICDE '98. IEEE, pp. 550–557.

Pajarola, R. B., 1998. Access to Large Scale Terrain and Image Databases in Geoinformation Systems. Ph.D. thesis, Diss. ETH No. 12729 Department of Computer Science, ETH Zürich.

Schneider, M., Stamm, C., Symanzik, J., Widmayer, P., 2000. Virtual Reality and Dynamic Statistical Graphics: A Bidirectional Link in a Heterogeneous, Distributed Computing Environment. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2000), Las Vegas, Nevada, June 26–29, 2000, Volume IV. CSREA Press, pp. 2345–2351.

Scott, D. W., 1985. Average Shifted Histograms: Effective Non–Parametric Density Estimation in Several Dimensions. Annals of Statistics 13, 1024–1040.

Sutherland, P., Rossini, A., Lumley, T., Lewin-Koh, N., Dickerson, J., Cox, Z., Cook, D., 2000. Orca: A Visualization Toolkit for High–Dimensional Data. Journal of Computational and Graphical Statistics 9 (3), 509–529.

Swayne, D. F., Cook, D., Buja, A., 1991. XGobi: Interactive Dynamic Graphics in the X Window System with a Link to S. In: 1991 Proceedings of the Section on Statistical Graphics. American Statistical Association, Alexandria, VA, pp. 1–8.

Swayne, D. F., Cook, D., Buja, A., 1998. XGobi: Interactive Dynamic Graphics in the X Window System. Journal of Computational and Graphical Statistics 7 (1), 113–130.

Swayne, D. F., Lang, D. T., Buja, A., Cook, D., 2002. GGobi: Evolving from XGobi into an Extensible Framework for Interactive Data Visualization. Journal of Computational Statistics and Data Analysis (To appear).

Symanzik, J., Cook, D., Lewin-Koh, N., Majure, J. J., Megretskaia, I., 2000a. Linking ArcView and XGobi: Insight Behind the Front End. Journal of Computational and Graphical Statistics 9 (3), 470–490.

Symanzik, J., Griffiths, L., Gillies, R. R., 2000b. Visual Exploration of Satellite Images. In: 2000 Proceedings of the Statistical Computing Section and Section on Statistical Graphics. American Statistical Association, Alexandria, VA, pp. 10–19.

Symanzik, J., Kötter, T., Schmelzer, S., Klinke, S., Cook, D., Swayne, D., 1998a. Spatial Data Analysis in the Dynamically Linked ArcView/XGobi/XploRe Environment. Computing Science and Statistics 29 (1), 561–569.

Symanzik, J., Majure, J., Cook, D., Cressie, N., 1994. Dynamic Graphics in a GIS: A Link between ARC/INFO and XGobi. Computing Science and Statistics 26, 431–435.

Symanzik, J., Majure, J. J., Cook, D., 1995. Dynamic Graphics in a GIS: Analyzing and Exploring Multivariate Spatial Data. ASA Statistical Graphics Video Lending Library (contact: (contact: dj@research.bell-labs.com).

Symanzik, J., Pajarola, R., Widmayer, P., 1998b. XGobi and XploRe Meet ViRGIS. In: 1998 Proceedings of the Section on Statistical Graphics. American Statistical Association, Alexandria, VA, pp. 50–55.

Szabo, K., Stucki, P., Aschwanden, P., Ohler, T., Pajarola, R., Widmayer, P., 1995. A Virtual Reality based System Environment for Intuitive Walk–Throughs and Exploration of Large–Scale Tourist Information. In: Proceedings Enter95: Information and Communication Technologies in Tourism. Springer, Vienna, pp. 10–15.

Unwin, A., Wills, G., Haslett, J., 1990. REGARD — Graphical Analysis of Regional Data. In: 1990 Proceedings of the Section on Statistical Graphics. American Statistical Association, Alexandria, VA, pp. 36–41.

van Dam, A., February 1997. Post–WIMP User Interfaces. Communications of the ACM 40 (2), 63–67.

Wegman, E. J., 1990. Hyperdimensional Data Analysis Using Parallel Coordinates. Journal of the American Statistical Association 85, 664–675.