PART IV

SOFTWARE DEVELOPEMENT

Part IV

Software development

Two GIS add-on tools developed by us have been described in this part of the thesis. ***Chapter 22*** deals with ***LandStat***, which is a new landscape analytical tool for the vector based GIS maps. This is an add-on to *MapInfo Professional5.5* and can perform landscape analyses such as landscape richness, similarity and heterogeity.

MapViewer, discussed in ***Chapter 23,*** is essentially a modified *MapInfo Professional5.5* environment, in which some essential features of GIS have been introduced.

Chapter 22

Chapter 22

MapViewer - *a new add-on tool for MapInfo Professional 5.5*

ABSTRACT

The new software, MapViewer has been developed mainly to introduce some essential features of GIS that are not available in MapInfo Professional 5.5. Several features of MapInfo such as layout, zoom in, zoom out, grabber, etc., have been incorporated in MapViewer, which implements them in an entirely different environment.

MapViewer has been coded in MapBasic 5.5 and Visual Basic 6.0. In this program the MapBasic commands are executed through Visual Basic, using the "integrated mapping" feature of MapInfo.

Apart from customizing the MapInfo environment, what stands out in MapViewer is its 3D rotation of windows and window overlay features.

The 'Window Info' tool of MapViewer lets the user to identify the ". tab" file, and its location. This 'Window Info' tool also performs the functions of the standard MapInfo 'Info Tool'.

MapViewer has been designed incorporating the state-of-the art modules to enhance the power and capabilities of the GIS software MapInfo Professional 5.5.

INTRODUCTION

Huge quantities of information are available today, more than ever before. Data abounds in spreadsheets, sales records, and marketing files. Paper and disk store enormous amount of information on customers, stores, personnel, equipment, and resources. Nearly all of this data has a geographic component.

An estimated 85 percent of all the databases contain some sort of geographic information such as street addresses, cities, states, ZIP codes, or

even telephone numbers with area codes and exchange numbers[1].

Desktop mapping can help one sort through all of this information, and use the geographic components in the data, the results can be displayed on maps, graphs, etc. This lets one see the patterns and relationships in the data quickly and easily without having to pour over the large database.

Using a typical desktop mapping software such as MapInfo, can display the data as points, as thematically shaded regions, as pie or bar charts, as districts, etc. The user can perform geographic operations such as redistricting, combining and splitting objects, and buffering. Also, one can make queries against the data and access the remote data directly from the GIS software.

For example, a GIS software can show which branch store is the closest to the biggest customers with more purchasing power. It can calculate the distances between customers and stores, it can color-code the store symbols by sales volume. What makes it all come together is a visual display of the data on the map.

THE PRESENT SYSTEM - MapInfo environment

MapInfo's building blocks: map layers

Digitized maps in MapInfo are organized as layers. These layers look like transparencies that are stacked one upon another. Each layer contains different features of the whole map in the form of point, polygon, polyline, or text.

For instance a typical geographic map for Pondicherry region could be sorted as district or taluk boundaries, water resources (rivers, lakes, etc,), vegetation (forests, agricultural lands, garden lands, etc.), soil types and so on, and each of these features are stored in a different file called as a 'table'; similar to an individual layer of the traditional McHarg's overlay.

In MapInfo the user begins by opening a table (refers to the file with specific data - either object data or attribute data) and displaying it in a Map

[1] **MapInfo Professional V5.5, 1999.** *User's guide* MapInfo Corporation, New York.

window. For viewing, each of these files/layers can be opened in the GIS environment enabling the user to choose either the political boundaries alone of political boundaries with vegetation or all the different layers together.

In conventional GIS environment these layers are stacked one over the other fitting into the registered X,Y coordinates. By stacking these layers one upon the other, the complete map would be built (Figure 1).

Thus, the map layers form the building blocks of maps in MapInfo. The user can customize the layers in a variety of ways: add and delete layers, or reorder them.

The proposed system - MapViewer

As seen above the digitized maps are organized as layers. Each Table (.tab file) is nothing but a layer of some map. When a new layer is opened in MapInfo it will by default be placed over the layer of the currently active window. Since all the layers are stacked, the user will not be able to distinguish the layers one from the other (Figure 1).

A typical GIS project usually involves vast number of layers. In such situations, it would be difficult to sort and differentiate between one layer and the other. Moreover, if the user wishes to look at each layer in a separate window in a typical MapInfo environment, he would need to open the layers in a single window and clone this window and remove all the layers except the one which is needed (Figure 1). This is quite a laborious process.

In MapViewer each table opens in a different window as shown in Figure 2. However, if the user wants to visualize several layers in one window, he can do so by using the 'add layers' command as usually done in MapInfo.

MapViewer also increases the desk top area for better visualization of map layers. There are many more features which are added to MapViewer like more informative window *info tool, 3D rotation* of *'window layers'*, simultaneous manipulation of all the windows, etc. (Figure 2)

MapViewer also incorporates several features of MapInfo, which have been modified to enhance the user friendliness. Thus MapViewer provides an


MapViewer

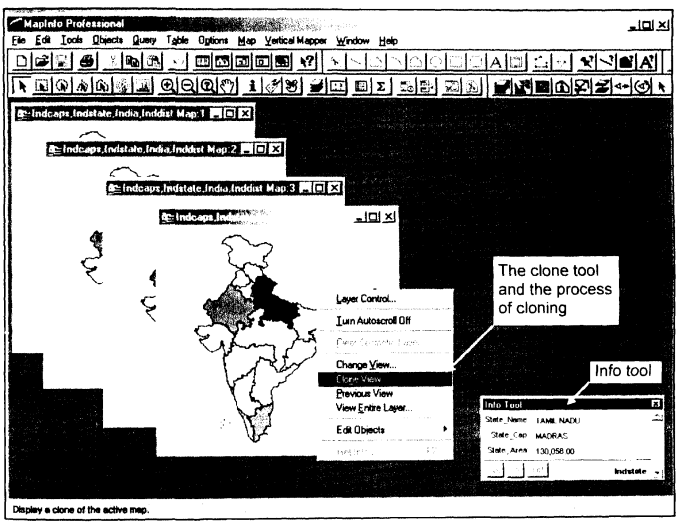


Figure 2 A typical MapInfo Professional 5.5 environment. To open different layers in each window one need to use the clone tool of MapInfo and arrange the windows. Also, note the info tool of MapInfo.

entirely new, comfortable and easy to use environment for GIS professionals to work.

SYSTEM DESIGN

Designing the environment

The appeal of any software lies in its environment: the outward look of the software like the color of the working area, organization of the menus, toolbars, etc.

The MapViewer environment has been designed using Visual Basic. After the splash screen (Figure 3), the user will be right into the MapViewer environment. The starting thing is a Visual Basic MDI (multiple document interface) form, from where the user will work.

Menus

The main menus of the MapViewer are

- File
- Opened Tables
- Options
- Layout
- Windows
- Help

The *File* menu contains some of the common operations like opening the tables, saving, printing, exiting from the software, etc. Using the Windows menu the user can perform various operations on the opened windows like moving, sizing, rotating, etc.,

The *Opened Tables* menu contains the list of tables that are opened. The table that is active will be checked against its file name. The user can also bring a table to the top (i.e., make it active) by checking that particular table's name in the *Opened Tables* menu.

The *Options* menu gives some key features like rearranging the order of the tables, and deleting them. The *Layout* menu helps the user to open layout window and perform all the operations on the layout window. The *Help* menu

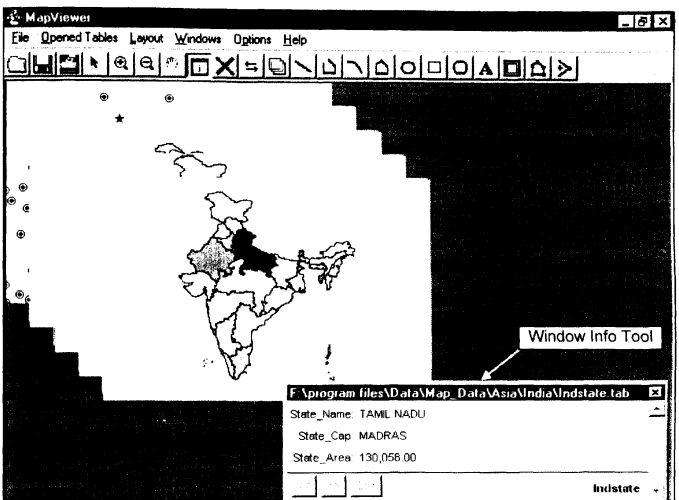


Figure 3 A typical environment of MapViewer. Each table is opened in MapViewer as a separate window which looks like the McHarg's overlay. The enhanced Info tool of *MapViewer*, the 'Window Info' tool showing full path of the table and the attribute information as shown by the 'info tool' of MapInfo.

displays the detailed help features of the software and the *About* MapViewer feature.

Toolbars

The toolbars contain some of the features present in the menus and some new features. The options like *Open, Save*, and *Print* are already available in the menus, but still they are also kept in the toolbar for quick access.

Most of the tools available in MapInfo are incorporated in MapViewer toolbars and some them are modified a little for increased efficiency such as *Select, Zoom In, Zoom Out, Grabber, Info, Label, Ruler,* and *Text.*

Deleting the active window, rearranging the windows with their current size, arranging the windows with their initial size (cascading), etc., are some of the new features available in the toolbar of MapViewer.

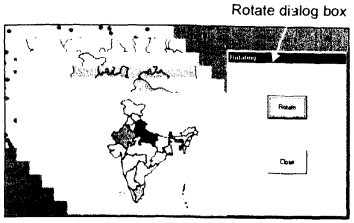
Map Windows

The key feature of MapViewer environment is the *Map Windows.* Here, the MapInfo tables are placed in the Visual Basic form. To be precise a frameless MapInfo with the specified table is placed over the Visual Basic form.

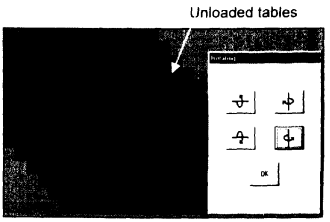
The Visual Basic form is designed in such a way that it gives complete flexibility to the user for manipulating them. In MapInfo, the Map Windows will be having caption bars and frames like any other normal window. This caption bar and the frames decrease visualization of the layers.

In MapViewer, the map windows caption bar and the frame are removed and the window frame has been trimmed so as to enable complete visualization of the table. The MapViewer environment has been so designed that the tables are opened as a group of windows with 12 tables each. Each group is arranged in parallel.

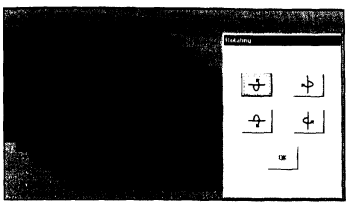
All the opened windows can be manipulated like moving, sizing, rotating (Figure 4), etc. simultaneously. Cascade and tile operations can also be performed. As seen earlier the names of all the tables, which are opened, are added to the *Opened Tables* menu. The user can activate any window by selecting the corresponding table name from this menu.



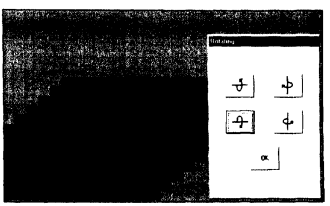
The tables in MapViewer before rotating the window groups



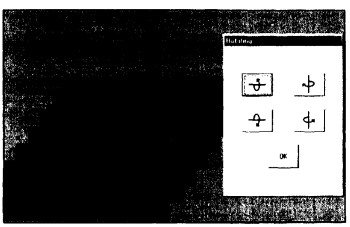
The tables are unloaded while rotating



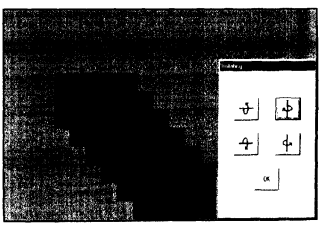
Rotating the window groups upwards



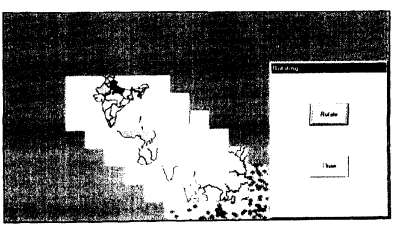
Rotating the window groups downwards



Rotating the window groups rightwards



Rotating the window groups leftwards



The tables in the window groups reloaded after finishing the rotation

Figure 4 Different options of rotating the window groups in *MapViewer* using the tool 'Rotate'.

INPUT DESIGN

Any number of MapInfo tables can be opened and manipulated in MapViewer. A frameless MapInfo window along with the specified table is placed over the Visual Basic window when a MapInfo table is opened in MapViewer. The interaction between MapInfo and Visual Basic is done through the integrated mapping feature of MapInfo.

In MapViewer the input can be sent as the messages given to the software through the selection of various menus and toolbars. These inputs are processed by MapViewer in two ways.

- Through MapBasic statements
- Through Visual Basic statements

In MapViewer the actions that manipulate the MapInfo tables are done using MapBasic statements, which are sent to MapInfo through Visual Basic. Those actions that manipulate the windows are done directly through the Visual Basic statements.

Starting MapInfo

To start a unique instance of MapInfo, it needs to call Visual Basic's CreateObject() function, and assign the return value to a Visual Basic Object variable. For example, if the Object variable is named as "mapinfo" then the following statement launches MapInfo:

```
Set mapinfo=CreateObject("MapInfo.Application")
```

To attach to a previously – running instance of MapInfo which was not launched by a CreateObject() call, Visual Basic's GetObject() function has to be used.

```
Set mapinfo=GetObject( "MapInfo.Application")
```

***Sending Commands to* MapInfo**

After launching MapInfo, text strings that represent MapBasic statements are constructed. For example, if MapInfo has to execute a MapBasic **Open Table** statement, the following string has to be constructed (within Visual Basic):

msg = "Open Table ""STATES.TAB"" Interactive "

This string can be sent to MapInfo by using the **Do** method as follows

mapinfo.Do msg

OUTPUT DESIGN

The output of MapViewer is viewed in the Visual Basic MDI form. The MapInfo tables are placed in the child forms of this MDI form.

Reparenting* MapInfo *windows

After launching MapInfo, one has to use the MapBasic statement **Set Application Window** so that MapInfo dialog boxes and error messages are owned by the client program.

```
msg = "Set Application Window " & FormName.hWnd
mapinfo.Do msg
```

Then, whenever one wants to integrate a MapInfo window into the Visual Basic application, send MapInfo a **Set Next Document** statement, followed by the MapBasic statement that creates the window. For example, the following commands create a MapInfo Map window as a child window of the Visual Basic program.

```
msg = "Set Next Document Parent " & MapFrame.hWnd & " Style 1"
mapinfo.Do msg

msg = "Map From States"
mapinfo.Do msg
```

The **Set Next Document** statement lets one to "reparent" document windows. Within the **Set Next Document,** the hWnd(handle) of a control in

Visual Basic has to be specified. The **Set Next Document** statement includes a **Style** clause, which controls the type of window we will create.

Manipulation of Map Windows

The 2-D and 3-D functionalities involve the manipulation of Visual Basic forms. If the user manipulates the windows with the tables in them, then it will be a time consuming process to move or resize the windows.

So, a more efficient way is followed in MapViewer for the manipulation of map windows. Whenever the user wants to perform any 2-D or 3-D functionalities on the map windows, all the windows are unloaded after storing the size and position of each window and also the names of the corresponding tables in the respective order.

The user can manipulate: resize, move, or rotate the windows as desired. These changes are reflected in the 'template' window frames. On selecting 'OK', the map tables are reloaded into the respective window frames.

Optimization of info tool's output

In MapViewer the output is optimized in many ways to give a better feel and look to the software.

In MapInfo, the Info tool gives the details about the point where the cursor is clicked, if there is some data associated with that particular point. But it does not give the details about the table to which the information is associated.

MapViewer has an Info tool that is modified to give more 'info' than MapInfo's. This 'Window Info' tool has been mainly modified to give details about the table to which the information is associated with. In this modified info tool, the caption bar contains full path of the table with which the information is associated, including the table name.

CODE DESIGN

In the case of MapViewer the MapBasic statements are executed through Visual Basic. The code has been segregated into different modules instead of

dumping of code inside the MDI form (Figure 5,6,7,8, and 9).

Apart from these modules, each form has its own coding which are used for the physical designing of the forms at run time. The parts of the coding that are common to more than one form are constructed into Visual Basic modules.

Some of the Visual Basic modules used in MapViewer are:

- *Layout*
- *Move*
- *Size*
- *Rotate*
- *Menus*
- *Reorder*

The Layout module takes care of the manipulation of Layout windows like reparenting them to the Visual Basic window, setting the layout units, and all the other functions that can be performed on a normal Layout window in MapViewer.

The Sizing and Moving modules are used to handle the 2-D functionalities of MapViewer. These modules mainly composed of Visual Basic code are used to manipulate the map windows' size and position. MapBasic statements are used only in reloading the maps after the manipulation of windows.

The Rotate module implements one of the essential features of MapViewer – the 3-D functionality. Using this module, the Visual Basic form (map window) is made to rotate 3 dimensionally. In fact, the Map Window is not rotated, rather an illusion is created by changing the size and position of the form (Figure 4).

The Menus module organizes the activities of menus and toolbars of the main MDI form. Its main function is to track the addition and deletion of map windows and update the runtime menu, which contains the names of the tables that are opened. The Reordering module helps in the process of changing the order of the tables that are open, and to update the 'Opened Tables' runtime menu.

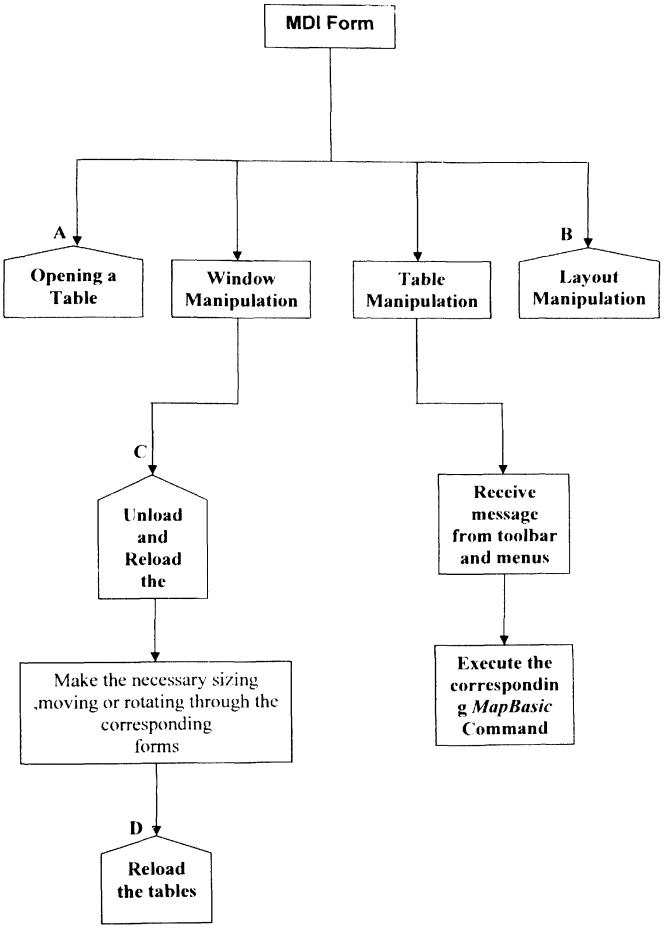


Figure 5 Overall flow of control for MapViewer

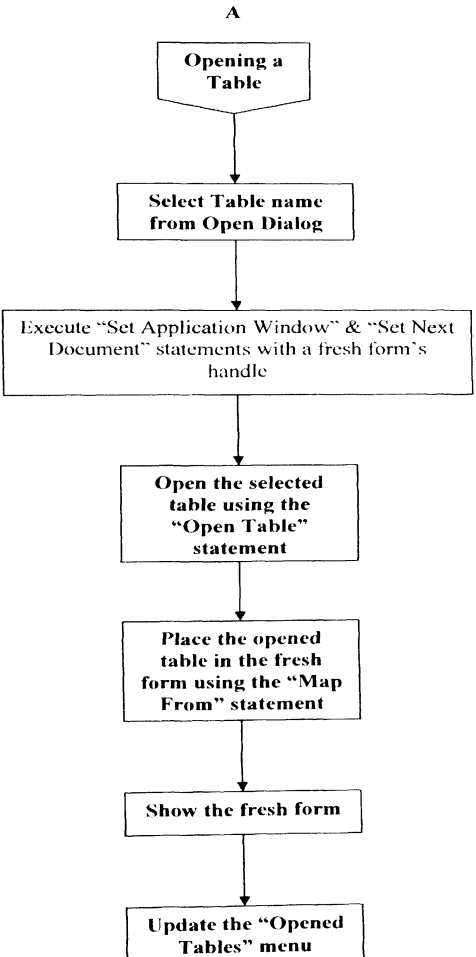


Figure 6 Steps involved in opening a table

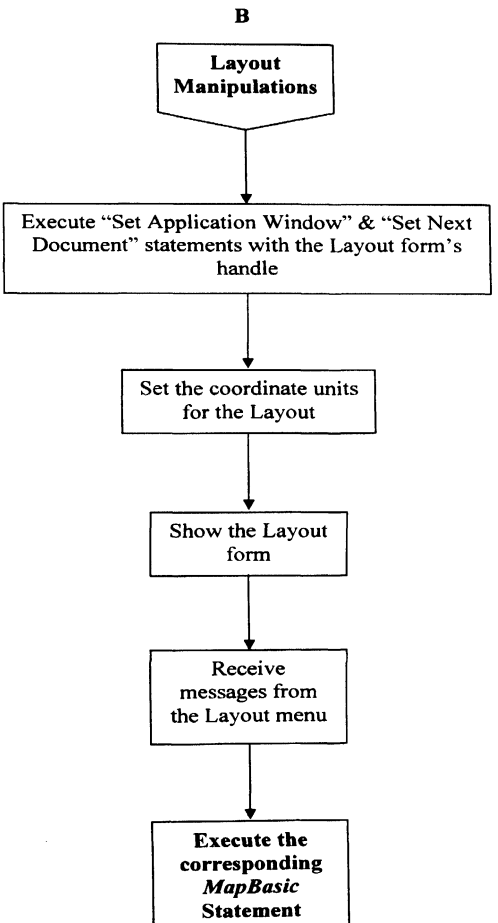


Figure 7 Steps involved in layout manipulations

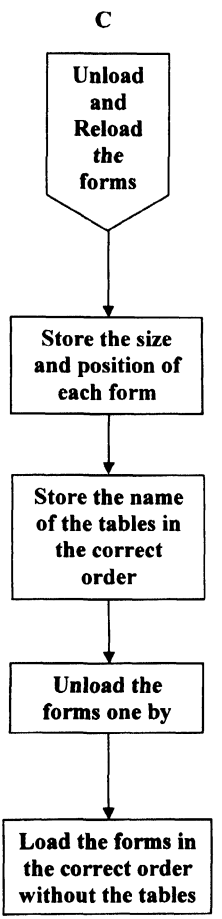


Figure 8 Steps involved in unloading and reloading the forms

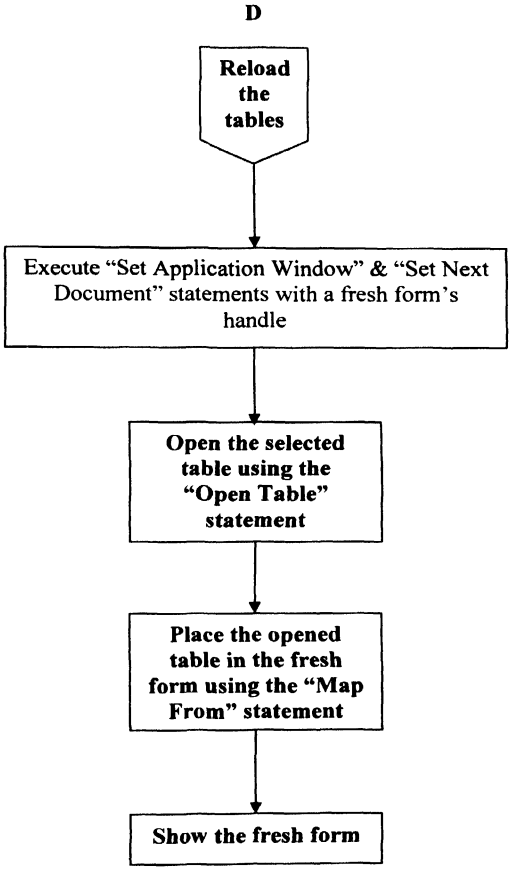


Figure 9 Steps involved in reloading the tables

TESTING AND IMPLEMENTATION

Testing

The system, once developed, has to go through a series of tests in order to ensure that it works the way it is ought to. Some of the routine tests that the software is usually subject to are:

- Test to see if the proposed requirements are taken care of
- Test to see if all the inputs are handled effectively
- Test the system by traversing all the paths and discover any surprises
- Check if the errors and exceptions have been handled properly
- See if the validations of the input data are taken care of
- Test the general robustness of the system

Apart from the above tests, unit testing, function testing, performance testing and integration testing were performed on MapViewer.

Unit Testing

When each of the proposed functionality is completed, it is tested to check whether the desired functionality is achieved completely or not.

Function Testing

Functional test involves testing the system under typical operating conditions, typical input values and for typical expected results. The functional boundary specifies boundary within which the system can function. Three types of functional tests that were done are

- Check if all the documented functions are working
- Check with the maximum input values
- Check to the extent to which it can run properly on giving valid inputs

Performance Testing

Performance tests are conducted to identify the bottlenecks in the system and to fine tune the overall performance of the system. This has been done by

observing the behavior of the system by giving all possible combinations of input values and overloading the system in various ways.

Integration Testing

The system has been finally tested after integrating all the functionalities, since some functions may not work properly when run after integrating.

CONCLUSION

MapViewer is certainly going to enhance the power of the powerful MapInfo. MapViewer adds some new powerful functionalities which can make things comfortable for the GIS professionals.

Some of the features of MapViewer like – Rotation of Window groups, the 'Window Info Tool', and the entirely new user-friendly environment are outstanding when compared with the conventional version of MapInfo *Professional 5.5.*

MapViewer can be very helpful for sorting the haystack of MapInfo files using some of the unique features such as Open tables and 'Window Info Tools.

No software is perfect and MapViewer is no exception. It has some drawbacks.

Even though opening tables in different windows is going to helpful in many ways, there might be some situations where the user wants to open tables in the same window. Then, the user has to do it using the layer control option, which is slightly time consuming.

The 'Window Info Tool' gives the full path of the table with which the information in it associated. But if a particular window has more than one layers or tables, this tool can give only the path of the table, which was opened first, in the particular window.

In spite of these drawbacks, we believe, MapViewer is still going to a powerful tool in GIS applications. These drawbacks will be rectified in the future versions of the software.

Chapter 23

Chapter 23

LANDSTAT - a new landscape analytical tool for the vector based GIS maps *

K.B.Chari, S.A.Abbasi @
Centre for Pollution Control and Energy Technology
Pondicherry University, R.Venkataraman Nagar, Pondicherry - 605 014.

Abstract

This paper details the various stages - developing, testing, and implementation of the *LANDSTAT*, a new analytical tool for performing landscape analyses over the vector map layers of the GIS tool *MapInfo*. *LANDSTAT* (LANDscape STATistical package), coded in MapBasic 5.5, can perform landscape analyses such as landscape heterogeneity, richness and similarity. The state-of-the-art module designing makes *LANDSTAT* efficient and user friendly.

LANDSTAT utilizes the principles of quantifying species richness, diversity and similarity for landscape analysis, in the context of landscape elements (patches) such as rivers, lakes, and landuse patterns as a unit.

LANDSTAT would find application in environmental management, land-use planning, landscape ecology, landscape architecture, urban planning, environmental impact assessment studies, and other related areas.

Introduction

Landscapes are seen as an important scale for many land management and conservation problems. Hobbs (1995) defines landscapes as heterogenous areas of land composed of interacting ecosystems or patches. Landscape ecology aims to study the patterns and processes operating at this scale, and focuses on landscape structure, function, and change.

The understanding of landscape patterns has increased greatly with the

@ Corresponding author.
Ph: (0 413) 655991 - 8 (Ext 311) / 655267 / 655263; Res: 655262 Fax; 655227
Email: prof_abbasi@vsnl.com

* Paper communicated to the Journal *of Landscape Ecology and Urban Planning*

advent of remote sensing and geographic information system technologies. Remote sensing offers the possibility to acquire data on the characteristics of the earth's surface in a relatively easy, repeatable, and analyzable way (Hobbs 1995). Geographic information systems allows us to develop spatially explicit data bases on a wide range of landscape features, which in turn allow quantitative analysis of pattern.

Many computer programs that have been written for landscape analyses were developed for use with raster, or grid cell, data. These days, with the vector based GIS catching-up the popularity, there is a huge need for analytical tools based on vector maps.

Vector maps can support landscape analysis better than the raster maps. As every vector object has a unique coordinate (representing the location), pen size and colour, fill style, area, and perimeter each landscape element - point, line, or polyline - can be identified precisely. The object attribute data would help in calculating the various landscape parameters such as diversity, evenness and similarity measures accurately and precisely. Unlike the raster maps spatial resolution (grain size) of the maps doesn't affect the results obtained in the case of vector images. The software that we have developed, *LANDSTAT,* is a vector-based tool for landscape analysis.

Landscape heterogeneity is frequently examined at two levels - differences in the heterogeneity of landscape units between two landscapes (regional scale) and differences in the heterogeneity within landscapes (landscape level). Various indices have been developed for use on both of these scales and include combinations of estimates of numbers of different landscape units present, areas occupied by different landscape units, and lengths of landscape unit perimeters. Other approaches consider the underlying patterns of plant and plant species richness and variations in evenness. The *LANDSTAT* software is based on the latter approach.

The *LANDSTAT* software

LANDSTAT is a menu-driven and interactive GIS tool for conducting landscape analysis over vector based GIS maps of MapInfo. *LANDSTAT*

utilizes the principles underlying the quantification of species richness, diversity and similarity at landscape level (Table 1). The software is capable of the following:

- estimate the richness measure of a landscape
- estimate the heterogeneity and evenness measures at landscape level
- estimate the similarity of a landscape over two different periods of time (temporal analyses) or between two different landscapes.

LANDSTAT incorporates state-of-the-art modules for calculating: (a) heterogeniety within a landscape (landscape scale) and among two different landscapes through Simpson, Shannon, and Brillouin indices (b) evenness among the landscapes through Simpson, Shannon, and Brillouin indices (c) the similarity among landscapes in terms of qualitative and quantitative measures.

One of the important features of the *LANDSTAT* is the ability to perform 'what if?' analysis - which would be highly useful in quantifying the change in a given landscape due to commissioning of a proposed project. This feature would be highly useful in the disciplines of environmental impact assessment and conservation ecology.

LANDSTAT is PC-based and consists of four main modules: Object search, Heterogeneity, Evenness, and Similarity. The architecture and message flow is shown in figure 1 and 2. Each module consists of several sub-modules and functions.

Table 1 Measurable features of landscape mosaics

Feature	Description
Richness	Number of different patch types in a given area
Heterogeneity	Confound the species richness and evenness in single index
Evenness	Equivalence in numbers (or areas) of different patch types in a mosaic (the inverse of the degree of dominance by one or a few patch types)
Similarity	The likeness or similarity between two different landscapes or the same landscape over different periods of time

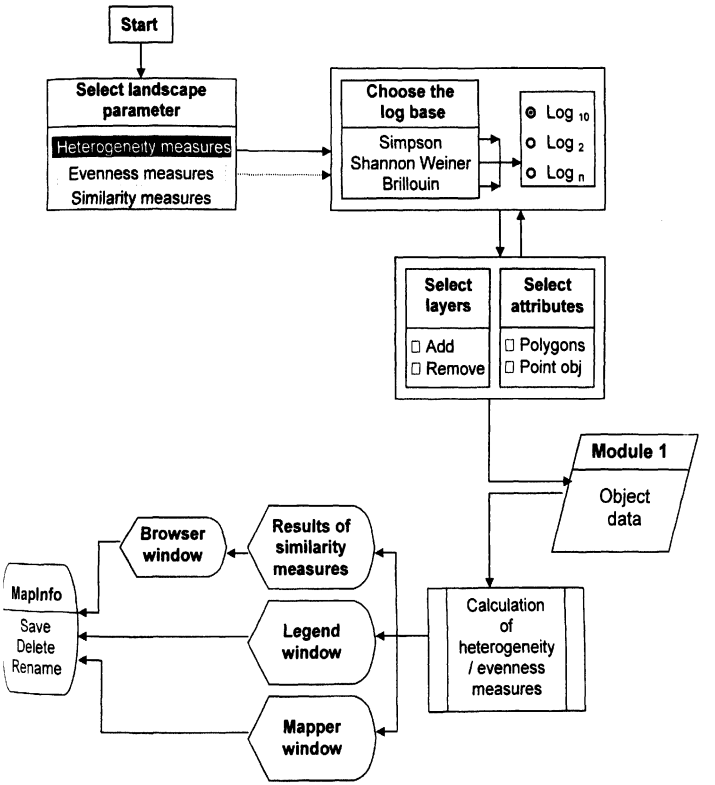


Figure 1 Data flow / steps invoved in Module2 - Heterogeneity measures, and Module3 - Evenness measures

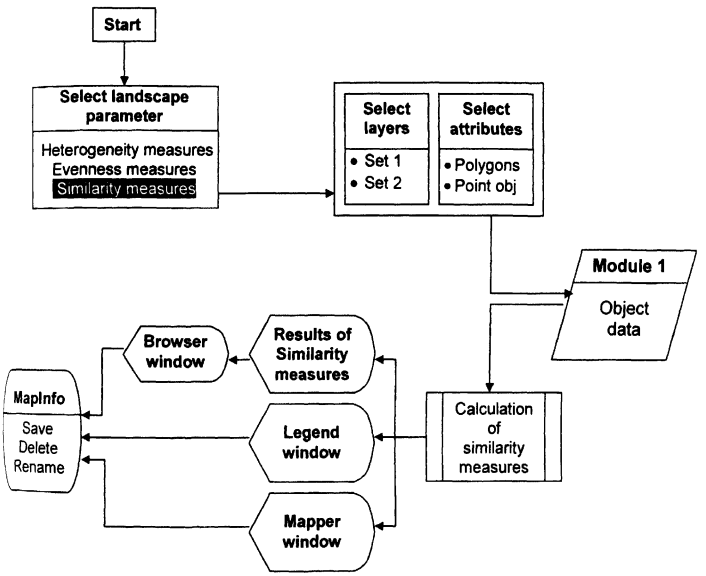


Figure 2 Data flow / steps invoved in Module 4: Similarity measures

Module Design

Module 1: Object Data Search (ODS)

This module forms the essential core of *LANDSTAT,* the input of which would be used for analyses by the subsequent modules. ODS has been designed for collecting the object data for the attributes selected (point objects, polygons) from the selected one or more active map layers by the user. The ODS returns the following object data information: the object types (point and polygons) with unique pen size and colour, fill style, area and perimeter and the object counts.

Module 2: Heterogeneity Analysis (HA)

Heterogeneity is an important, but difficult to measure feature of the most landscapes. It is a complex multiscale phenomenon involving the size, shape, and composition of different landscape patches and the spatial and temporal relations between them (Hobbs, 1995).

The module 2, Heterogeneity Analysis (HA), quantifies the landscape heterogeneity indices using the input data from ODS Module. This module calculates three diversity indices, named after Simpson, Shannon-Wiener and Brillouin (Krebs 1987).

The user has the following options for measuring the landscape heterogeneity:

i) select one of the three log bases: $\log_n$, $\log_{10}$, or $\log_2$, for each of the indices; the default log base being $\log_{10}$

ii) select the layers for which the heterogeneity need to be calculated

iii) select the object types: point objects or polygons or both

Upon finishing the calculations, *LANDSTAT* displays the results as 'results window' which will further lead to a 'browser window'. The browser window can be saved, deleted or renamed using the standard MapInfo commands. Also the *LANDSTAT* will display a 'mapper window' with the map objects, if they are mappable, and a 'legend window' showing the objects types identified and used for calculations.

(a) Simpson's index of diversity

This is a non-parametric measure of diversity, proposed by Simpson (Krebs 1987). The following equation suggests that the diversity is inversely related to the probability that the two individuals picked at random belong to the same species. This can be equally applicable in the case of landscape diversity too, if one considers the landscape elements (points and polygons) as individual landscape elements.

(i) for the infinite population

$$1\text{-}D = 1\text{-}\sum (p_i)^2 \quad \ldots\ldots 1$$

Where 1-D = Simpson's index of diversity

p_i = Proportion of the individuals of objects *i* (points / polygons) in the selected map layers.

A few of the modifications of the Simpson's index are as follows.

(ii) for the finite population

$$1\text{-}\hat{D} = 1\text{-}\sum_{i=1}^{s} [n_i(n_i\text{-}1)) / N(N\text{-}1)] \quad \ldots\ldots 2$$

where $1\text{-}\hat{D}$ = Simpson's index of diversity

n_i = number of the individuals of objects *i* (point, polygons) in the selected map layers

N = Total number of objects in the selected layers = $\sum n_i$

(iii) Williams and McArthur modification of Simpson's index

$$1 / D = 1 / (\sum p_i^2)$$

Where 1 / D = Simpson's reciprocal index

p_i = Proportion of objects *i* in the landscape

Simpson's index (1-D) ranges from 0 (low diversity) to almost 1 (1 - 1/*s*). The reciprocal of Simpson's original formulation (1/D) varies from 1 to *s*; where *s* represents the observed total number of objects present in the selected layers.

(b) Shannon-Wiener index of diversity

This index which measures the order contained in a system, has been one of the most popular measures of species diversity. The following equation of Shannon-Weiner has been applied for measuring the heterogeneity at landscape level:

$$H` = - \sum_{i=1}^{s} (p_i) \log_2 p_i \quad \ldots\ldots 3$$

Where $H`$ = Shannon-Wiener index of diversity

s = Number of species

p_i = proportion of the total sample belonging to the i^{th} object.

The theoretical maximum value for the Shannon - Wiener measure is log(S), and the minimum value (when N >S) is log[N/(N-S)], Where S, represents the number of object types present in the selected map layers.

(c) Brillouin's Index

In situations, where the data that are obtained nonrandomly from a population or data comprising an entire population, the following information-theoretic diversity measure of Brillouin is appropriate (Zar 1999):

$$H = 1 / N \log ((N!/n_1!*n_2!*n_3!) \quad \ldots\ldots 4$$

Where H = Brillouin's Index of diversity

N = total number of individual objects in the sample

n_1 = total number of individuals belonging to object 1

n_2 = total number of individual objects belonging to object 2 (etc.)

Unfortunately, MapBasic can perform factorials for only objects less than 100. Thus *LANDSTAT* has been programmed such that it skips this particular diversity index when the object count exceeds 100.

Module 2: Evenness measures

One of the most common approaches for measuring the evenness (or equitability) has been to scale one of the heterogeneity measures relative to its maximal value when each species in the sample is represented by the

same number of individuals. This can be attributed for the landscape as each landscape element (points / polygons) represented by the same numbers.

The evenness module gets the object data input from module 1. The diversity indices and object count are utilized in this module for calculating evenness of the Simpson, Shannon-Wiener, and Brillouin indices of diversity. The data flow in this module is presented in Figure 2.

All evenness measures usually range from 0 to 1.

(a) Evenness measures for Simpson's index of diversity

(i) For the infinite population

$D'_{max} = 1/S$ 5

Where D'_{max} = Maximum possible value for Simpson's index.

S = total number of objects in the map layers

V'(Evenness)= D / D'_{max} 6

Where D = observed diversity index.

(ii) Evenness for finite population:

$N/S = I+J/S$ where J *and* S are two integers and $J< S$

$D'_{max} = I [2J + S(I-1)] / N(N-1)$

$D'_{min} = [(S-1)(2N-S)] / N(N-1)$

Where S = total number of the object types

N = total number of objects.

I = integer value of N/S.

J = remainder of the individuals.

Evenness = $(D-D_{min}) / (D_{max}-D_{min})$ 7

(b) Evenness measures for Shannon-Wiener index of diversity

$H'_{max} = -S(1/S\log_2 1/S)$ 8

$= \log_2 S$

Where H'_{max} = maximum possible value for the Shannon-Wiener index of diversity

S = total number of the object types

Evenness $J' = H' / H'_{max}$ 9

Where J' = Evenness measure

H' = Shannon-wiener index of diversity

(c) Brillouin measure of evenness

$N/S = I+J/S$

$H_{max} = 1/N \log [N! / (I!)^{S-J} (I+1)^{J}]$

Evenness = H / H_{max} 10

Where H_{max} = maximum possible value for the Brillouin's index for N individuals in S object types.

Module 3 Similarity measures

Often, landscape analyses involve measuring the similarity or the likeness among two different landscapes or the same landscape at different periods. This kind of analysis helps in evaluating the temporal change in landscapes.

Qualitative Measures

The qualitative measures of similarity, also called *binary* similarity coefficients, utilize the presence-absence data of an object (point, polygon) given the spatial coordinates.

(a) Coefficient of Jaccard

$S_J = a / (a+b+c)$ 11

Where S_J = Jaccard's similarity coefficient

a = number of objects in Map A and Map B (joint occurrences)

b = number of objects in Map B but not in Map A

c = number of objects in Map A but not in Map B

(b) Coefficient of Sorenson

$S_S = a / (a +b + c)$ 12

Where S_S = Sorenson's similarity coefficient

a, b,c = as defined in the equation 11

(c) Simple matching Coeficient

$S_{SM} = (a + d) / (a+ b + c + d)$ 13

Where S_{SM} =Simple matching similarity coefficient

a, b, c, = as defined in the equation 11

d = number of objects absent in both Map A and Map B (zero - zero matches)

(d) Baroni - Urbani and Buser Coefficient

$S_B = (\sqrt{ad} + a) / (a + b + c + \sqrt{ad})$ 14

Where S_B = Baroni - Urbani and Buser similarity coefficient

a, b, c, d = as defined above

The range of all similarity coefficients for binary data generally would range between 0 (no similarity) to 1.0 (complete similarity).

Quantitative measures

(a) Percentage similarity

This measure also called as Renkonen index, is calculated by standardizing the sample data as *percentages* so that the relative abundance's all sum to 100 in each sample. The index is then calculated as:

$P = \sum \text{minimum } (P_{Ai}, P_{Bi})$ 15

Where P = Percentage similarity between Map A and Map B

P_{Ai} = Percentage of objects *i* in Map A

P_{Bi} = Percentage of objects *i* in Map B

This index ranges from 0 (no similarity) to 100 (complete similarity).

The actual Renkonen formula uses only the items present in both the lists. This we felt would not reflect the 'change' when it comes to the dissimilarities. This is obvious from the above table. The MVSP utilizes the actual renkonen formula. We have modified Renkonen formula by including all the objects present in both the lists and then comparing the objects present in both. Thus the results of LANDSTAT and MVSP doen't seem to match.

(b) Morisita's Index of Similarty

$C\lambda = (2 \sum^{n} X_{ij} X_{ik}) / [(\lambda_1 + \lambda_2) N_j N_k]$ 16

Where $C\lambda$ = Morisita's index of similarity between Set j and Set k

X_{ij}, X_{ik} = number of individual objects *i* in Map j and Map k

$N_j = \sum X_{ij}$ = Total number of individual objects in Map *j*

$N_k = \sum X_{ik}$ = Total number of individual objects in Map *k*

$\lambda_1 = \{\sum^n [X_{ij}(X_{ij} - 1)]\} / [N_j(N_j - 1)]$

$\lambda_1 = \{\sum^n [X_{ik}(X_{ik} - 1)]\} / [N_k(N_k - 1)]$

The morisita's index of similarity measures between 0 (no similarity) to 1.0 (complete similarity). This index was formulated for the object counts in a map and not the abundance estimates.

(c) Horn's Index of Similarity

$R_O = \{\sum [(X_{ij} + X_{ik}) \log((X_{ij} + X_{ik})] - \sum (X_{ij} \log X_{ij}) - \sum (X_{ik} \log X_{ik})\} / \{[(N_J + N_K) \log(N_J + N_K)] - (N_J \log N_j) - (N_K \log N_K)\}$ 18

Where R_O= Horn's index of similarity for samples j and k

X_{ij}, X_{ik} = number of individual objects *i* in Set j and Set k

$N_J = \sum X_{ij}$ = Total number of individuals in sample j

$N_K = \sum X_{ik}$ = Total number of individuals in sample k

Input design

Input protocol is a major consideration for effective performance of any software. The proposed system gets its input from the vector map layers of MapInfo.

The user has an option to choose the object types: point or polygon for the calculation of landscape diversity. The software counts the select objects of different sizes (pen size, fill style) and stores it in the corresponding variables.

The well - designed input of *LANDSTAT* serves four purposes: controls overflow, reduces redundancy in recording data, increases clerical accuracy, and allows easier checking of data.

Output design

The primary consideration in the output design is arranging the output in a

form convenient to the user. Custom dialog boxes are used to display the result and the user has provisions to view the table data in the Browser window. The Object styles can be viewed using the Legend Window of MapInfo.

Testing

The *LANDSTAT* software was tested for quality assurance. The testing has been conducted in three phases as follows.

Unit testing. The developer and supervisor to developer did the unit testing in order to trace out bugs in each part of the code. It was done during the code development. It included testing each function and procedure after its development is done.

Module testing.This test has been performed for each module, during which each case has been tested thoroughly in order to discover pitfalls. The project has emphasized development in class phases and hence this testing was very important. It has been done by the developer and the supervisor to developer.

System testing. A team was formed comprising four senior people of the organization to test the system thoroughly. The team, besides discovering some loopholes also gave suggestions on the future enhancements. For checking the accuracy and precision of statistical calculations, the object details (object count and object types) were fed into a DOS based MVSP (Multi-Variate Statistical Package). The results were compared (Table 2). The package could successfully calculate heterogeneity, evenness and similarity measures.

Application of *LANDSTAT*- an example

Preamble

A typical garden-land area in an Indian suburb (~2.91 km^2), which is now

being increasingly developed and populated, has been taken for exploring the application of LANDSTAT in GIS.

The study area (figure 3) was extensively surveyed for (a) vegetation (b) developmental features such as roads, buildings, playgrounds, water tanks etc., and (d) topography. These details were mapped in several layers.

The analog / paper maps were scanned with HP Scanjet 6300 colour scanner. All the subsequent processing for GIS was done with the tool *MapInfo Professional v5.5*. The landscape features were analysed using the LANDSTAT software.

The results obtained by the LANDSTAT were compared with the DOS based Multi-Variate Statistical Package (MVSP). The object count and abundance of the symbols and polygons were fed into MVSP for calculating the heterogeneity, evenness, and similarity.

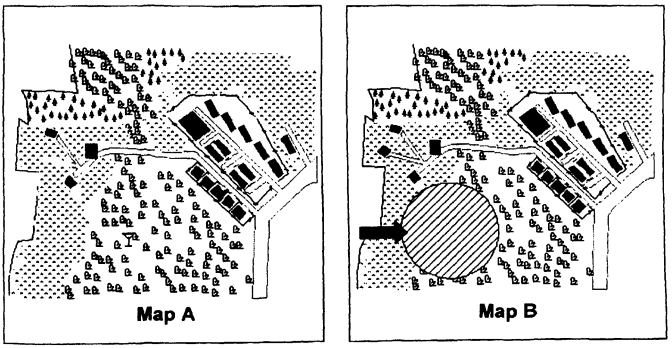


Figure 3 Map A represents a segment of the study area which has been used in the analysis of the landscape diversity of the region using LANDSTAT software (Table 2). Map B represents hypothetical scenario where a portion of the landscape has been modified (indicated by the arrow). The similarity of the landscapes represented by Maps A and B has been calculated by the LANDSTAT software (table 3)

Table 2 Landscape heterogeneity and evenness values calculated by the *LANDSTAT* software for the Map A is shown in Figure 3. The results obtained by the *LANDSTAT* for the landscape diversity is compared with the results obtained by using the Multi-Variate Statistical Package (MVSP). (MVSP is a DOS based package meant for ecological statistics for the numerical data input. The landscape diversity indices were obtained by feeding the object types and the number of objects into the MVSP package).

	Heterogeneity		Evenness	
Index	***LANDSTAT* Values**	**MVSP Values**	***LANDSTAT* Values**	**MVSP Values**
Simpson (1-D)	0.54706	0.549	0.65042	0.6500
Shannon- Weiner				
Log 10	0.450354	0.450	0.532902	0.533
Log e	1.03698	1.037	0.532902	0.533
Log 2	1.49604	1.496	0.532902	0.533
Brillouin				
Log 10	0.43245	0.432	0.51100	0.511
Log 2	0.99742	0.994	0.51100	0.511
Log e	1.43540	1.434	0.51100	0.511

Table 3 Similarity measures between the two landscapes, Map 1 and Map 2 as calculated by the LANDSTAT software and MVSP.

Similarity measures	**LANDSTAT Results**	**MVSP Results**
Jaccard	0.750	0.750
Sorenson	0.857	0.857
Simple Matching Coefficient	0.778	0.778
Percentage similarity	67.6539	92.593@
Morisitta's index	1.17919	Option not available

@ The actual Renkonen formula uses only the items present in both the lists. This we felt would not reflect the 'change' when it comes to the dissimilarities. This is obvious from the above table. The MVSP utilizes the actual renkonen formula. We have modified Renkonen formula by including all the objects present in both the lists and then comparing the objects present in both. Thus the results of LANDSTAT and MVSP doen't seem to match. However the *LANDSTAT* results reflect the actual change.

Discussion

The results obtained by using LANDSTAT matches very closely with that of the MVSP. Thus, it can be said that LANDSTAT can be used in calculating the heterogeneity, evenness and similarity very effectively.

CONCLUSION

LANSTAT has been developed as a comprehensive and user friendly tool for calculating the landscape heterogeneity , similarity measures and evenness measures for the vector map layers of MapInfo.

LANDSTAT can be highly useful in environmental impact assessment (EIA) studies, landscape architecture and management, urban planning, forestry, and conservation ecology.

Generally an EIA study tends to identify *the change that a proposed project would likely make to the environment, or the change that has been already made to the environment, by a commissioned project.* Usually the assessment is done of the impact on several individual components of the environment such as surface water sources, ground water, ambient air, flora etc. But it is now being increasingly realized that impact assessment should be done at the level of effected landscapes in an integrated manner.

For instance, if a water resource project such as a dam were commissioned, the dam would submerge several landscape features such as forests, rangelands, habitats etc., thereby altering the entire landscape drastically. A much diverse landscape would gradually change into a less diverse landscape where the water body would dominate. To measure such changes qualitatively and quantitatively, concepts of landscape ecology are being used in tandem with GIS. By using *LANDSTAT* change in the landscape features along a period could be identified and quantified easily.

Urban planners, landscape architects would find ***LANDSTAT*** highly reliable, precise and easier to measure several features of the landscape. The likely changes in the landscape on the implementation of a proposed plan / project can be quantified using the similarity and diversity measures.

In forestry management and conservation ecology, the software can be helpful in identifying the potential zones for conservation.

References

Hobbs R.J., 1995. *Landscape ecology*, Encyclopaedia of Environmental Biology, pp:417 - 428.

Krebs C.J., 1987. *Ecological Methodology*, Harper & Row, Publishers, New York.

MapBasic V5.5, 1999a. *User's guide* MapInfo corporation, New York

MapBasic V5.5, 1999b. *Reference Manual* MapInfo corporation, New York

MapInfo Professional V5.5, 1999. *User's guide* MapInfo corporation, New York

Zar J., 1999. *Biostatistical analysis*, Prentice - Hall International, Inc. pp:663