Datamology Company

A venture of The Janina Ratkowska Group

VisiCube

The Data Microscope

Version 0.5

Copyright Notice

Trademarks

Acknowledgements

Ernie Anderson invented, designed, and developed *VisiCube*.

Erin Thomas tested *VisiCube*, packaged it for distribution, and developed the website from which it is distributed.

Chip Hartney assisted in the design of *VisiCube* and wrote this manual.

Barbara Fruzyna supported us all in these efforts.

Dedication

VisiCube is dedicated to Janina Ratkowska, the late wife of Ernie Anderson, sister of Barbara Fruzyna, and inspiration for The Janina Ratkowska Group.

Contact Information

Learn more about *VisiCube* at www.visicube.com.

Learn more about Datamology at www.datamology.com.

Write us at info@visicube.com.

Summary of Contents

Table of Contents

Conventions Used in This Manual

logos	New words or terms are displayed in an italicized 12-point proportional font (Times New Roman).
`visicube.exe`	Named files (including programs) and folders are displayed in a 10-point fixed font (Courier New).
Help	Buttons are referenced by their labels in a bold 10-point proportional font (Arial).
www.datamology.com	Hyperlinks are displayed in a blue font and are underlined. These are active links that can be clicked to access the named location on the Internet.
Navigator	Names of the components of VisiCube are displayed in an italicized 12-point proportional font (Arial).
Layer Trivariate Data	Window and *SynchroView* names are displayed in an italicized 12-point proportional font (Times New Roman).
Chapter 1	Cross-references are displayed in a blue font and are underlined. These are active links that can be clicked to access the named location in this manual.

About Version 0

It's Free

Version 0 of *VisiCube* is, and always will be, free. It can be obtained directly from our website and it can be passed among people freely.

It's Limited

Version 0 is fully functional, but lacks features that will be available in later versions of the product. We welcome your suggestions as to how the product can be improved and we are dedicated to providing a complete and bug-free product. However, no technical support, other than this manual, is provided with Version 0.

About this Manual

While every effort is being made to provide a complete and informative manual, we are not there yet. Our current efforts are more focused on the product, itself. So, for the time being, this manual is incomplete. But, we promise to remedy that soon.

Chapter 1 Why VisiCube?

We think *VisiCube* is great. We know that sounds like a sales pitch. It is. But we think it important to communicate to you our confidence in our product. In this chapter, we expand on this simple statement and attempt to explain the value of *VisiCube*. In the chapters that follow, we'll abandon the sales effort and simply provide you the most accurate and complete documentation we can. In the end, we must…and will…let you decide whether *VisiCube* is right for you.

Overview

VisiCube is a data analysis software program. Data analysis is the critical step in research between data collection and publication. However, *VisiCube* does not do the analysis. It is, after all, up to you to truly analyze your data. Only you have the domain expertise and the uniquely human capabilities of organization, decomposition, synthesis, generalization, induction, proposition, inference, deduction, rationalization, thought, and much more. *VisiCube*, as an important partner in that process, provides powerful facilities to manipulate and present your data in a concise and clear manner even when that data is complex. Specifically, statistically meaningful visualizations are utilized to present your data because visuals are most easily processed and understood by the human brain. And, just as importantly, *VisiCube* works with your data at its most granular level giving you access to the individual datum as well as summarizations of that data.

The basic approach taken with *VisiCube* is to break your set of data into meaningful subsets and then to generate visualizations using logical combinations of these subsets. The number of ways this can be done is potentially astronomical, even for small amounts of data. *VisiCube* enables you to move through this universe of possibilities quickly and effectively. The visuals are generated automatically by VisiCube using mathematically correct techniques, leaving you to do the actual data analysis.

The Need

Data Abundance

Computers have given us the ability to electronically capture and retain an amazing amount of information. It is said that we now live in the information age…a time in which the amount of data at our fingertips is enormous and growing exponentially.

But, as noted above, capturing that data is only the first step along the research path that leads to conclusions and new knowledge. The power of the computer must also be brought to bear on the second step along that path…the step of analysis.

Datamology

We apologize for the mix of Latin and Greek, but it is in the spirit of these learned traditions that we use the term *datamology* to describe the activity of studying information. The Greek word *logos*, which is the root of the English word *logic*, means study. The Latin word *datum*, which is the root of the English word *data*, means information in its most common usage.

Literally, however, *datum* means gift or present. We believe that it is in the true study of information that one discovers, and comes to understand, these gifts. Our target audience, therefore, is comprised of those people who are serious about the study of information. We think of these folks as datamologists.

The Opportunity

Supercomputer Availability

The cause of the information age is, more than anything, the computer. And, as remarkable as it may have seemed just a few years ago, supercomputers are now available to anyone in the world in the form of standard desktop (or laptop) personal computers. It is with such a supercomputer that you can pursue datamology. You now have the processing power to churn through vast amounts of data very quickly and the display power to present that data in high-quality visualizations (again very quickly).

Unaddressed Needs

At the great risk of oversimplification, it is our view that existing software options for datamologists are limited and lacking in some fundamentally important ways.

Chart generators, which provide a plethora of chart types, are included with many software applications. Pie charts and bar charts with three-dimensional effects can be seen everywhere. But such charts are often not much more than decorations that correlate with the data and are only capable of showing the obvious. Such applications are woefully inadequate for datamology because they provide only (over)simplified data presentation tools. They lack data manipulation and analysis capabilities.

Statistical software applications, while providing powerful computational facilities, are often extremely mathematical in nature and prove difficult to use to the average datamologist. Statistics is a highly complex science requiring great care in its application. Non-experts are often left to utilize statistical methods by rote without the ability to understand the applicability of particular methods to their data. Further, these applications are oriented toward the reduction of a set of data to individual statistics, in which the nuances of the data are lost, as opposed to giving the datamologist access to the entire set of data in its most granular state. Put another way, they are very good at statistical analysis (when applied correctly), but found to be lacking in the more general act of data analysis.

Multidimensional data analysis applications, including online analytical processing (OLAP) tools, lean in the opposite direction. While on the right track, they are generally geared toward business use and, therefore, tend not to provide the in-depth analytic features required by datamologists.

The Solution

The Datamology Company

We at Datamology are dedicated to the needs of datamologists. (Again, datamologists is our term for those who are truly serious about data analysis but who are not necessarily statisticians or computer experts.) We are mathematicians with a strong orientation to academic research and have many decades of experience in custom and commercial software. Our previous software projects covered

the entire range of data processing including data management, data reporting, and data warehousing. Using these attributes and experiences, we are determined to provide the most effective software application for datamology that we possibly can. It is our one, and only, goal.

VisiCube

VisiCube is our software application for datamologists. It has been built to meet the following general goals (which we have determined to be most important to datamologists):

- It must facilitate serious, detailed, and thorough analysis of your multivariate data.
- Its use must be intuitive to reduce your need for expertise in computers and software.
- It must allow you to model your data accurately.
- It must present your data in a statistically correct and meaningful manner without requiring expertise in statistics.
- It must present data with the most powerful techniques available (which are visual) to enable you to analyze your data efficiently and accurately.
- It must be proactive in leading you through your analysis activities.

Since these goals do, at times, conflict with each other, we can't claim that *VisiCube* is the perfect tool for datamologists. In fact, we are always open to suggestions as to how it can be improved. However, we believe it to be a quality product which addresses these goals well.

The Data Microscope

It can be counter-intuitive but, in order to fully understand large amounts of data, it is important to be able to study each and every part of that data. Statistical analysis and statistical modeling are important techniques in the study of data. The theory of statistics lets you reduce unmanageable masses of data to models that can be used to make predictions about the underlying phenomena. But statistical modeling is a very sophisticated science (beyond the full understanding of most non-mathematicians) and it is easy to apply it incorrectly. You can be misled because the data is not there to speak for itself…it is lost inside the model. What is needed is a tool that allows you to explore and study your data without reducing it to mere statistics. That tool is a data microscope, a tool that allows you to look at the data as a whole as well as examining small portions of that data.

VisiCube, as a data microscope, gives you an alternative to understanding your data statistically. It uses some statistical concepts to help you orient yourself within your data, but all of the data is visible…you always see reality. This visibility of the data is another important way in which *VisiCube* acts as a data microscope. Everything is visual. More importantly, these visualizations are graphical in nature because graphs provide the best mechanism to present large amounts of data in an efficient and concise manner. However, mathematically, there are only a very few ways to graph data clearly and correctly. Instead of embellishing such graphs with extraneous structure having nothing to do with the data, *VisiCube* uses them as primitive building blocks for constructing higher-level visuals having far greater power to illuminate and clarify the data.

But It's Really Different

In addition to the more obvious ways in which *VisiCube* differs from other datamology software applications, there is an extremely important, though somewhat subtle, way in which *VisiCube* is truly unique. Even though it is a Windows application in that it runs in the Windows operating system, *VisiCube* is not a Windows application in regards to how you interact with it. We have gone to extraordinary lengths over many years to develop an interface which is both more powerful and more efficient than the standard Windows interface. It is truly a new approach to interaction between the software and the user of the software. And, although the interface is not software itself (i.e., *VisiCube* is a datamology application regardless of its interface), it does greatly determine the usability of that software. In *VisiCube*, we are proud to introduce our user interface because we think it a significant leap forward in the efforts to make software both simple and powerful.

Non-Modal

More than anything, *VisiCube* can be characterized as non-modal. *Modality* is a term used in the computing industry to define your ability, within a software application, to move between similar types of components of the application (especially windows). A modal dialog is one in which you must interact with the software in an inflexible series of discrete steps in order to accomplish a given task. Such a dialog freezes the rest of the application until the current window is dismissed in an acceptable manner. A non-modal dialog is one in which you are free to jump between windows without penalty. The standard Windows paradigm is, more often than not, one of modal dialogs. In fact, this is not unique to Windows applications. It is common throughout the software world.

Put bluntly, modal dialogs are used because they are easier (and, therefore, cheaper) to create. If a software application forces you to complete a complex task through a pre-specified series of steps, it becomes a relatively simple matter to manage those steps. If, on the other hand, it allows you to get to the desired end via any of the many possible paths to that end, the complexity of that management task increases exponentially. In other words, modal dialogs are generally used…but not for your benefit (except indirectly in terms of the price of the product). They are used for the benefit of the company that created the software.

And, if you think modal dialogues are not common, ask yourself how often you have been faced with a situation in which you wanted to proceed in a manner which the application did not allow. Perhaps you wanted to take a look at another window to get some information that is required by the current one. Perhaps you realized that you had forgotten to make a related change before embarking on the current task. Perhaps you simply wanted to back up a few steps without having to do so one at a time or, worse, having to start over. If you think hard, you'll find the number of such experiences to be enormous. In a modal dialog, you have to either proceed in the pre-defined manner or cancel the task you had set out to accomplish. Only after you had then taken care of the related task could you then safely return to the original task. But, even then, you probably had to start from scratch. Put another way, ask yourself how many **OK** and **Cancel** buttons you have seen. These are the classic identifiers of a modal dialog.

Developers of modal software have long recognized the shortcomings of modal dialogs and have taken some steps to alleviate the pain associated with them. For example, they utilize tabbed windows in which logically related "windows" can be compressed into a single physical window. Of course, these still have their own limitations since they are still locked into the modal paradigm. For example, you can still only see one of the tabs at a time. And, typically, the tabbed window itself is

part of a modal dialog. The effect of using the tabbed window is not seen until you complete your interaction with it and apply the changes. We believe that such efforts are ill-advised. If it is acknowledged that modal dialogs are problematic, then they should not be perpetuated.

We at Datamology have taken the time and effort to make *VisiCube* non-modal in all areas except those that require modal dialogs (such as object creation dialogs in which partial creation would be meaningless and confusing).

Navigation

Each of the non-modal dialogs of *VisiCube*, which we call *SynchroViews*, is accessible through our menu-like navigation system. However, this navigation system does not have the modal constraints of a normal menu system. The standard Windows menu is one in which you traverse a hierarchical organizational structure, beginning at the top level of that hierarchy, one level at a time until you find and select the desired entry in the menu. Being modal in nature, such a menu does not allow you to jump directly to anything except a top-level entry in that menu.

Further, since we utilize the concept of non-modal dialogues consistently throughout the application, we have no need of embedding some dialogs in the menu. Most Windows applications, in an effort to keep the main working window available at all times while presenting a modal dialog, embed those dialogs in the menu. It is in such dialogs that you are given the ability to define how your work in the main window should be changed or altered. Of course, being modal, you cannot see the changes until you have dismissed the dialog. And, if you don't like the results, you must traverse your way back through the menu and try again.

Since *VisiCube* is not restricted by the use of modal dialogs, all work is done in the main work space and the menu system reverts to a more intuitive and natural navigation system which is unencumbered by embedded dialogs. This navigation system, which we call our *Navigator*, gives you the ability to select any *SynchroView* directly. The *Navigator* displays all of the *SynchroViews* in an organized hierarchy. You simply click on the one that you would like to access without having to work down through many levels of modal lists.

Synchronization

Non-modal dialogs, being comprised of multiple related windows, present a special challenge to the software developer. If the user of the software is allowed to move between these windows, each with its own distinct set of controls and capabilities, the software must decide what to do with the various potential inputs. The inputs of one window must either be ignored by the other windows (probably leading to extreme confusion) or be utilized by them. It is this latter approach to which *VisiCube* adheres. Specifically, all of the windows of a dialog are synchronized at all times. For this reason, we call such non-modal, synchronized dialogs *SynchroViews*.

You will also see that the windows of a *SynchroView* are not provided with any means of exit. There is no need. If they are applicable, they are displayed. Of course, you need not utilize each of the windows and have the ability to arrange and resize them as you wish. But exiting from a single window has no meaning in a *SynchroView*. When you select a different *SynchroView* (via the *Navigator*), the new *SynchroView* entirely replaces the old one and all windows of that old *SynchroView* are automatically exited.

Automatic Save

Further, you might notice that there are no **Save** buttons in *VisiCube*. (Nor are there any of the even more dreaded **SaveAs** buttons!) In our opinion, one of the great nuisances of modern software is its dependence on you to manage the files in which your work is saved. We still provide complete control over the folders in which files are stored, the organization of those folders and the files within them, and the names of those folders and files. You still have the ability to copy, delete, rename, and move your files. The difference is that you no longer have to explicitly tell the software when to save your work. Instead of the paradigm adopted by most software applications in which it is assumed that you do not want to save your work, *VisiCube* adopts the opposite paradigm. We assume you want to save your work and do so for you automatically and constantly. Because of this, you cannot lose your work.

Automatic Continue

Finally, *VisiCube* remembers what you were doing when you leave. Upon starting *VisiCube*, you will automatically be placed back in the same *SynchroView*, working on the same file (with all of the changes which were automatically saved), as when you left it last. There is never any effort required to continue your work across invocations of *VisiCube*. It assumes that continuation is desired and automatically makes this possible.

Chapter 2 Getting Started

In this chapter, we explain how you install *VisiCube* on your computer and then use it.

Obtaining VisiCube

VisiCube is available from Datamology as a single installation program. It can be directly downloaded from our website or, in the case of Version 0 only, can be copied legally from any other person who already has the installation program. This single program, named `VisiCubeInstall.exe`, is utilized to completely install *VisiCube* and all related files on your computer. When obtaining the installation program, whether by download or any other means, be sure to note the folder in which it is saved on your computer.

Download

The installation program can be downloaded directly from our website at www.visicube.com. Follow the instructions there to download the installation program to your computer.

Installing VisiCube

Once you have obtained the installation program, named `VisiCubeInstall.exe`, you must execute it on your computer to install *VisiCube* and configure your computer for its use.

Starting the Installation Program

To start the installation program, do the following:

- Start Windows Explorer or My Computer
- Navigate to the folder that contains the installation program.
- Double-click on the name of the installation program.

Completing the Installation

Once the installation program begins, follow its instructions. When the installation program completes, *VisiCube* is ready for use on your computer.

You need not alter any of the default settings used by the installation program. But, if you choose to do so, the following information is provided to assist you in making those choices.

Installation Folder

The installation folder is the folder into which the *VisiCube* program (which is not the same as the installation program) and its related files will be placed. This can be any folder on your computer, but we strongly recommend use of a folder under the `Program Files` folder which already exists on your computer.

Preparing Yourself

Once you have installed VisiCube, it is ready for use. But, are you? We strongly recommend you take a few minutes to prepare yourself to use it efficiently and productively.

Gather data

Before using *VisiCube* to study data, you should locate the desired data and make it available to *VisiCube*. Generally, this requires you to move the applicable data files to your computer or place them on a removable disk that can be read by your computer.

Read this manual

To use *VisiCube* successfully, you should read this manual. It describes the concepts upon which *VisiCube* is built and includes detailed instructions about the use of *VisiCube*. Note that the online help system is more granular in nature. It describes, in detail, the use of each *SynchroView* but presupposes an understanding of the information presented in this manual. Specifically, we encourage you to do the following:

- Read *Chapter 3* (*VisiCube Concepts*) in its entirety to understand the mathematical and informational foundations upon which *VisiCube* stands.
- Read *Chapter 4* (*VisiCube Usage*) in its entirety to understand how to use *VisiCube* efficiently and productively.
- Read *Chapter 5* (*VisiCube Organization*) as needed to clarify how the *SynchroView*s are related to each other and how they can be used to accomplish your goals.
- Use the Glossary as needed to understand the terminology used in *VisiCube* and this manual.

Using VisiCube

In this section, we describe the basic operation of *VisiCube*. As will be described later, *VisiCube* uses some components and paradigms that are not common in other Windows applications, but it is a standard Windows application in terms of how it is invoked and controlled. Therefore, some of the following material is likely to be obvious if you are an experienced Windows user.

Opening VisiCube

Assuming a standard installation, any of the following methods can be used to open *VisiCube*.

Via Windows Desktop

- Double-click the *VisiCube* icon

Via Windows Start Menu

- Click the Windows **Start** button
- Click the **Programs** entry
- Click the **VisiCube** folder entry

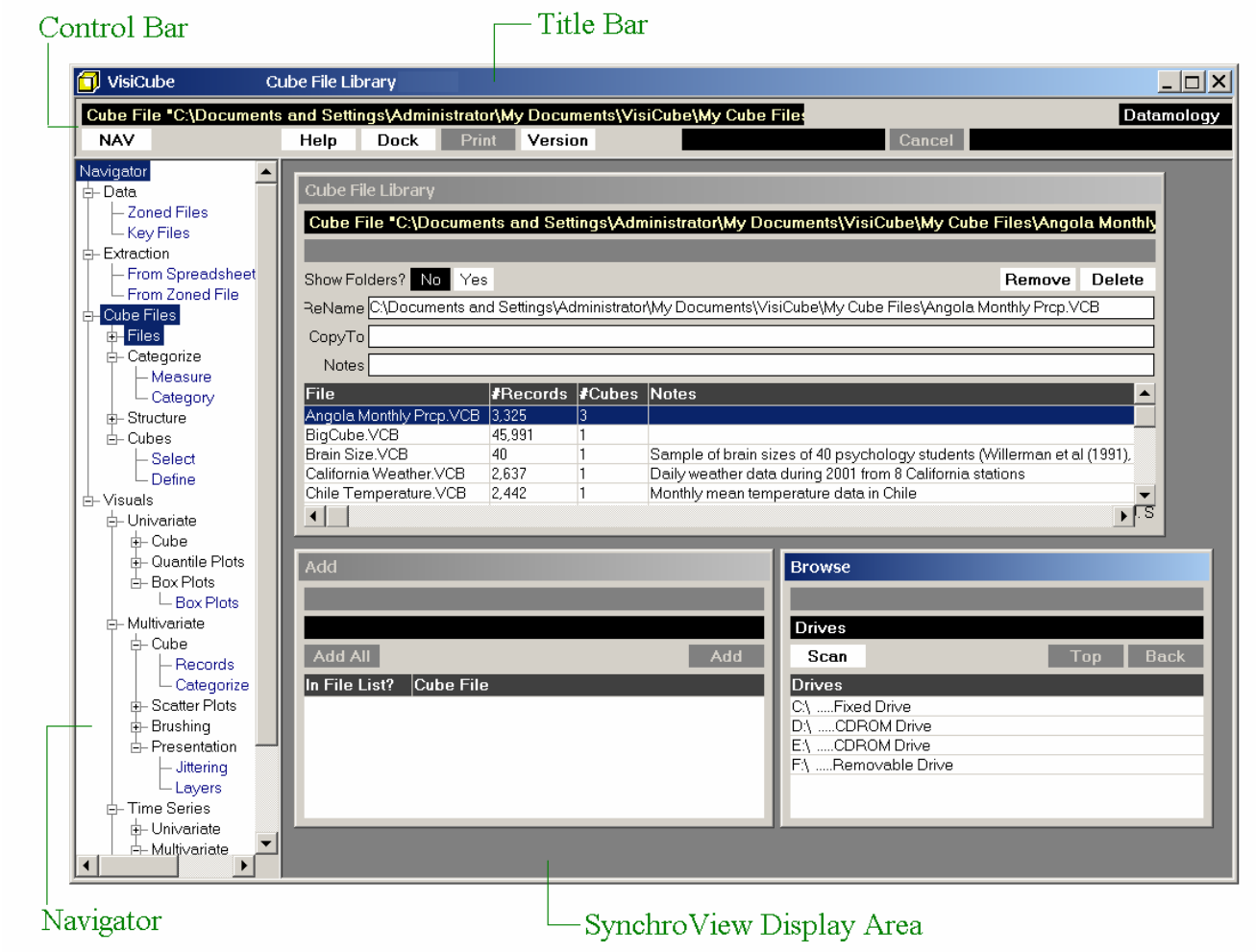
- Click the **VisiCube** program entry

Via Windows Explorer

- Navigate to the *VisiCube* installation folder (which was determined during installation)
- Double-click the *VisiCube* program name (`visicube.exe`)

Orienting yourself

VisiCube, as it appears on your monitor, is comprised of four distinct sections, each having its own unique location and purpose.



Title Bar

The title bar is the horizontal bar across the very top of *VisiCube*. It contains the following components:

- On the left is the product control icon. It can be clicked (left-click or right-click) to expose the product control menu from which *VisiCube* can be adjusted or closed.
- Next to the product control icon is the "VisiCube" product title.

- In the middle is the name of the current *SynchroView*. (*SynchroView*s are described below.)
- On the right are the product control buttons. They can be left-clicked to adjust or close *VisiCube*.

Control Bar

The control bar is the grey horizontal bar at the top of *VisiCube* just below the title bar. Within this area are the components you use to control *VisiCube* (independent of the current SynchroView) and is comprised of two rows of components.

The first line of the control bar is informational in nature and contains the following components:

- On the left is the name of the cube file currently being processed by *VisiCube*. (Cube files are described later in this manual.) This name can be toggled between its long form (including its path) and its short form by clicking on the name.
- On the right is the "Datamology" company name.

The second line of the control bar contains the following components:

- On the left is the **NAV** button which switches the visibility of the *Navigator* between its hidden and visible states. (The *Navigator* is described below.)
- In the middle are the control buttons which are available to you whenever *VisiCube* is waiting for you. (I.e., when *VisiCube* is not working.)
    - The **Help** button can be left-clicked to display information which will assist you in using the current *SynchroView*. Note that this information does not, as a rule, describe the interactions and relationships between *SynchroView*s nor the use of *VisiCube* as a whole. These are higher level concepts that are described in this manual. (Of course, this manual can be displayed online, along with *VisiCube*, by using the Acrobat Reader that is available at no charge from Adobe Systems.)
    - The **Dock** button can be left-clicked to rearrange the current *SynchroView* into its default layout.
    - The **Print** button can be left-clicked, when there is a printable window available in the *SynchroView*, to cause the contents of that window to be printed on your printer.
    - The **Version** button can be left-clicked to display descriptive information about *VisiCube*.
- On the right is the control button, which is available to you whenever *VisiCube* is doing certain time-intensive work for you such as generating a plot. Note that, while the work is being done, status messages are displayed on each side of this button. The message describes the work being done and its progress.
    - The message field to the left of the button displays the type of work that *VisiCube* is undertaking.
    - The **Cancel** button can be left-clicked to stop *VisiCube* from completing the work. If The *SynchroView* can be displayed without the work being completed, it is. Otherwise, you are returned to the prior *SynchroView*. I.e., the one from which you

chose to access this *SynchroView*. And, if there is no prior *SynchroView*, which is the case when you first open *VisiCube*, you are returned to the Cube File Library SynchroView.

- The message field to the right of the button is a progress indicator. It describes the progress of the work (usually in terms of a count that decreases to zero) and is updated constantly while that work is in progress.

Navigator

The *Navigator* is the vertical pane (whose visibility is optional) on the left. It fulfills the purpose of a menu in that it provides you access to the various *SynchroView*s of *VisiCube*. However, it differs from a common menu in that it is always available (should you decide to keep it visible) and provides, with a single click, direct access to any *SynchroView* from any other *SynchroView*. Its visibility is controlled by the **NAV** button in the control bar.

The *Navigator* is a hierarchical tree, with collapsible nodes, in which the *SynchroView*s of *VisiCube* are organized into logical groups. The *SynchroView*s are the end nodes of each branch of that tree. The groups into which the *SynchroView*s (and other groups) are organized are higher level nodes of that tree. You may click any *SynchroView* node to directly access that *SynchroView*. Alternatively, you may click any group node to access the *SynchroView* in that group which you most recently accessed.

SynchroView Display Area

The *SynchroView* Display Area is the large rectangular pane in the lower right. This is the space in which the current *SynchroView* is displayed. It is here that you do your work by interacting with the windows of that *SynchroView*. The particular *SynchroView* to be displayed in this area is selected in the *Navigator* as described above.

The term, *SynchroView*, is utilized to indicate important aspects of this display area. First, it is a view of your work. That work is organized into one, or more, windows where each window is responsible for a relatively simple and independent portion of that work. Second, those windows are synchronized with each other. Any change made in one window of the *SynchroView* is immediately reflected in the other windows in a logically consistent manner. You may move between the windows at will.

Interacting with VisiCube

In popular language, the *user interface* (UI) of a computer product is the place at which the user believes s/he interacts with the software. In reality, this interaction is two-way in nature. The person communicates with the software through input devices such as the keyboard and/or mouse. The software communicates with the person through output devices such as the monitor and/or printer. Practically speaking, the vast majority of this interaction is dictated by visual prompting on and through the monitor. Therefore, the UI for a software application typically refers to the method in which the application communicates with the person through the monitor. Such a UI is typically comprised of many distinct types of elements and is used most effectively by a person that understands that interface.

As noted in the previous chapter, *VisiCube's* UI is different from many Windows applications. To use *VisiCube* effectively, you should understand the elements of its UI and how to utilize them. This

section explains those elements and the paradigm behind them so that you can interact with *VisiCube* efficiently.

Elements of the VisiCube Interface

The following discussion describes elements in terms of their colors to help clarify the functionality of the individual elements. The colors mentioned are those displayed when the Windows Standard color scheme is utilized on your computer.

Buttons

Buttons are elements which, when "pushed", cause a pre-specified action to take place. The applicable action is described by a label on the button. The label appears in a bold font.

The availability of a button is determined by context. If the button is available, it is activated by moving the mouse over it. An active button can then be pushed via left-click of the mouse. The condition of a button is indicated by the following:

- Unavailable buttons use light grey text on a dark grey background.
- Available buttons use black text on a white background.
- Active buttons use white text on a dark blue background.

When an active button is pushed down, the colors change to black text on yellow background. When the button is released, the applicable action is taken.

Toggles

Toggles are complex elements providing a mechanism for state selection. In a toggle, one (and only one) state is always selected. Each of the toggle states is represented by a flat box whose label indicates the matching state. The labels appear in a regular (non-bold) font. The states are always preceded by text that describes the toggle…typically in the form of a question…to clarify the choice to be made.

Like buttons, a toggle state becomes active when the mouse is moved over the state. At that time it can be selected via left-click of the mouse. The condition of a toggle state is indicated by the following:

- The selected state uses white text on a black background.
- Each unselected state uses black text on a white background.
- An active state uses white text on a dark blue background.

When an active state is selected, that state is selected and the previously selected state is unselected.

File Name Boxes

File name boxes are informational elements in which a file is identified. They appear as flat rectangular elements with light yellow text on a black background and have two modes. In one mode (the default), they identify the file with a full path name. In the other mode, they identify the file with only the file name. The mode is changed by left-clicking on the element.

Information Boxes

Information boxes are elements in which a message is displayed. They appear as flat rectangular elements whose coloring indicates the type of message as follows:

- Messages indicating success or normalcy use light blue text on a dark blue background.
- Messages indicating a problem or unexpected result use light red text on a black background.

If there is no message to display, the box is grey.

Lists

Lists are complex elements appearing in tabular form with rows and columns. The purpose of a list is to provide you a mechanism to choose one object from a set of many. The top row of the table contains column headings which describe the objects in the list. Such headings use white text on a charcoal black background. Each object in the list is represented in one row of the table.

The condition of an object is indicated by the following:

- The selected object uses white text on a bark blue background.
- Each unselected object uses black text on a white background.

When an object is left-clicked, the object is selected and the previously selected object is unselected.

Text Boxes

Text boxes, elements in which you can enter text, are white with a thin black border. The text that appears in these boxes is black. To utilize such a box, enter the desired text and then hit the **Enter** key. If you exit the box without hitting the **Enter** key, your changes will not be utilized by *VisiCube*.

Scroll Bars

Scroll bars are complex elements with sculpted components which provide you with a mechanism to adjust the applicable display. All components of a scroll bar are grey. The arrow keys at each end of the scroll bar can be left-clicked to cause the display to scroll a small amount. The slide can be moved via drag-and-drop to adjust the display to the desired position. Finally, you can left-click in the slide area, outside of the slide itself, to cause the display to scroll a large amount.

Colors of the VisiCube Interface

Most of the colors utilized by *VisiCube* are dictated by the Windows color scheme which is currently in effect on your computer. For a few components, however, the colors are dictated by *VisiCube* explicitly. (Generally speaking, these are the colors used in plots.)

Windows-managed Colors

Although you may utilize any color scheme you choose, we recommend use of one of the following as they have been found to be most compatible with the explicit *VisiCube* colors. Note that these are all "classic" color schemes:

- Windows Standard
- Plum
- Slate
- Storm

- Rainy Day
- Marine
- Teal
- Desert
- Brick

VisiCube-managed Colors

Though not intended to be a definitive list of explicit color settings in *VisiCube* (because we reserve the right to make changes to this list as we deem necessary), what follows is a list of the colors that we manage explicitly:

- All colors in all picture boxes in all plot windows.
- Non-coordinated info bars use white text on a black background
- Coordinated info bars, which are used across a set of windows in a *SynchroView*, use light blue text on a black background.
- File bars use light yellow text on a black background.
- Dataset bars use yellow text on a black background.
- An active toggle uses white text on a black background.
- *SynchroView* nodes in the *Navigator,* when not highlighted, use dark blue text. Group nodes use black text.
- List box column titles and highlighted columns use charcoal gray highlighting as background.
- Message bars, when displaying a message in response to an action, use light red text on a black background (to indicate a warning) or light blue text on a dark blue background (to indicate success).

Adjusting VisiCube

Assuming yours is a standard installation, the following methods can be used to adjust *VisiCube* as it appears on your monitor.

Minimize

VisiCube can be minimized with the left-most product control button or through the product control menu. Minimization removes *VisiCube* from the active desktop in Windows. This does not, however, close *VisiCube*. To resume using *VisiCube*, you must un-minimize it. (See below.)

Un-minimize

VisiCube can be un-minimized (from a minimized state) using the standard Windows mechanism for doing so. Typically, this is accomplished by clicking on the *VisiCube* task icon in the Windows Start bar.

Maximize/Restore

VisiCube can be toggled between full-screen and shared-screen modes with the middle product control button or through the product control menu. (If *VisiCube* is currently in full-screen mode, it can only be restored to shared-screen mode. If it is currently in shared-screen mode, it can only be maximized to full-screen mode.) In full-screen mode, no other applications are viewable and *VisiCube* cannot be resized or moved. In shared-screen mode, other applications are viewable and *VisiCube* can be resized or moved.

Move

When in shared-screen mode, *VisiCube* can be moved around the active desktop in Windows. This can be accomplished by dragging the title bar or through the product control menu (with the assistance of the arrow keys).

Resize

When in shared-screen mode, *VisiCube* can be resized in the active desktop in Windows. This can be accomplished by dragging a side or corner of *VisiCube* or through the product control menu (with the assistance of the arrow keys).

Closing VisiCube

Assuming yours is a standard installation, the following methods can be used to close *VisiCube*.

Via product control button

- Left-click the **x** button.

Via product control menu

- Left-click the “Close” option in the menu.

Chapter 3 VisiCube Concepts

In this chapter, we explain the concepts upon which *VisiCube* is built. It is important that you understand these concepts in order to use *VisiCube* effectively and productively.

Mathematics

Much of datamology is built on mathematics and, though you don't need to be a mathematician to use *VisiCube*, you do need to understand some mathematical concepts as they are used throughout the product as well as this manual.

Set Theory

In large part, *VisiCube* is based on set theory. This is a mathematical discipline, developed by Georg Cantor in the early 1870s, in which (among other things) sets and their properties are defined rigorously.

Sets of Members

A set is simply a collection of any number of members. (A set containing no members is an empty set.) The members of a set can be anything at all and they need not be the same in any way. Sets are often represented by listing their members inside a pair of braces and separated by commas. We utilize that method in this manual. For example,

```
{Joe, Mary, Fred}
```

and

```
{Mary, Helen}
```

are two sets of people and Mary is a member of both sets.

Another important way of representing a set is to describe it (inside the braces) with a rule which can be applied to a large list of potential members. For example,

```
{Gender = Female}
```

could be used to define a possibly large set of people which would be awkward to list individually.

Set Operations

Set operations are those operations which, when applied to one or more sets, produce a new set. VisiCube deals with the *union* and *intersection* operations.

The union of two or more sets is a set consisting of those members that are contained in any one of the sets. Unions are indicated by the connective "OR". For example, the union

```
{Joe, Mary, Fred} OR {Mary, Helen}
```

produces the set

```
{Joe, Mary, Fred, Helen}
```

The intersection of two or more sets is a set consisting of those members that are contained in each of the sets. Intersections are indicated by the connective "AND". For example, the intersection

{Joe, Mary, Fred**}** AND **{**Mary, Helen**}**

produces the set

{Mary**}.**

Subsets

Set A is a subset of set B if every member of A is also a member of B. For example, the set **{**Joe**,** Mary**}** is a subset of the set **{**Joe**,** Mary**,** Helen**}**. However, the set **{**Joe**,** Mary**}** is not a subset of the set **{**Joe**,** Fred**,** Helen**}**.

By the above rule, any set is a subset of itself. Thus, **{**Joe**,** Mary**}** is a subset of **{**Joe**,** Mary**}**.

Sets of Sets

A member of a set can itself be a set. An example of a set of sets is the set **{ {**Joe**,** Mary**}**, **{**Helen**,** Fred**} }**.

Ordered Sets

In general, the members of a set have no particular *order*. For example, the set **{**Joe**,** Mary**}** and the set **{**Mary**,** Joe**}** are the same set. A set in which the members are ordered is called an ordered set. Ordered sets are represented by displaying their members inside a pair of brackets. For example, **[**Joe**,** Mary**]** and **[**Mary**,** Joe**]** are each ordered sets and they are different from one another.

Multivariate (or Univariate) Data Analysis

In statistics, the term *variate* is used to describe a variable (an attribute whose value changes). Multivariate data analysis is analysis done on more than one such variable. Univariate data analysis, obviously, describes analysis done on a single variable. *VisiCube* supports multivariate data analysis (and, therefore, also univariate data analysis).

Data

Data records are *ordered* sets of values. A value can be either *qualitative* (such as the color *red*) or *numeric* (such as the number *56.5*). The sets **[**red**,** 56.5**,** 125**]** and **[**black**,** 67.8**,** 165**]** are examples of data records. It is permissible for a data record to have one or more *missing* values. The data record **[,** 47.6**,** 202**]** has its first value missing. Missing values are handled automatically in *VisiCube*, as will be described later.

VisiCube works with *sets* of data records (thus, sets of sets). It will always be the case that every data record in such a set of data records has the *same* number of values. Because the data records are ordered sets, it makes sense to talk about a *particular* value of the entire set of data records, say the 2nd value.

It is therefore meaningful to *name* the 2nd value. For example, it might be named "length". Similarly, the 1st value might be named "color" and the 3rd value might be named "weight". These names would apply unambiguously to every data record in the set. The *general* entities we are here naming are called the fields of the data record.

Any set of data records with which *VisiCube* is working can be imagined as a table with the individual data records being the rows and the fields being columns with identifying names. In fact, this is how *VisiCube* displays them. In this document, we will sometimes treat a set of data records as a table of rows and columns (displaying all the field values), and sometimes as a set of members such as **{***R1***,** *R2***,** *R3***, …}**.

Measures

In data analysis, it is often useful to think of the data fields as being either independent or dependent fields. The dependent fields are those whose values depend on the values of the independent fields. For example, when collecting rainfall data, it is common to record the location at which the measurement was made, the date on which it was made, and the amount that was measured. In such a case, the location and date would usually be considered to be independent fields and the amount would be considered dependent. Stated simply, the amount measured depends on the location and date of the measurement.

In *VisiCube*, we use the paradigm of measurement and refer to the dependent fields as measures. I.e., measures are the fields whose values are the result of a measurement.

Categories

Following the measurement paradigm further, all data fields which are not measures simply describe (or categorize) the measure(s). We refer to these fields as categories. In the rainfall example noted above, both location and date would be considered categories. In fact, they are independent variables. However, it should be noted that categories may include fields that are not independent variables with regard to the measurement. For example, the rainfall data could include another field in which the size of the state in which the measurement is taken is recorded. That size does not affect the measurement in any way. It is simply additional information about the measurement. It is, therefore, considered a category which can be used in later analysis.

Cubes

Multidimensional Hypercubes

The categories of your data can be used to define your data in a multidimensional manner. A multidimensional hypercube (or, simply, cube) is a conceptual object which exists in a large Cartesian space bounded by all dimensions of the cube. Each point in this space can be considered a cell in the cube which contains the applicable measures. The dimensions of the cube are the aspects through which you choose to view those measures. In our rainfall example, you would likely choose to view the measures in a two-dimensional manner through location and date. The coordinates of a particular cell are the dimensions. For example, Station #1 and December 25, 2002, would be the coordinates of a single cell in the location-by-date cube. If the rainfall was measured at that station on that date, the amount would be stored in that cell. In fact, if a second measurement (i.e., a second reading of the measuring instrument) were made at the same location on the same date, a second value would be stored in that cell.

Chapter 4 VisiCube Usage

In this chapter, we explain how you can use *VisiCube* to productively analyze your data. This is not meant to be a comprehensive review of all of the *VisiCube* facilities and capabilities. Instead, it is our intent to provide you with an understanding of *VisiCube* to facilitate your usage of it. Please read *Chapter 5* (*VisiCube Organization*) for details about the specific capabilities of *VisiCube*.

General Approach

VisiCube usage consists of the following basic steps. First, you define the sources of the data that you wish to analyze. Then, you capture that data into a proprietary cube file. Next, you organize that data into structures which facilitate analysis. Finally, you analyze that data through manipulations and visual presentations of those structures.

Locate Your Data

VisiCube must be able to access and interpret your data before you can proceed with analysis of that data. In defining your data, you tell *VisiCube* where that data is, how to access it, and how to interpret it.

To make your data available to *VisiCube*, the file must reside on a drive that is accessible to your computer. Such drives include, for example, your hard drive, floppy drive, network drive, or CD-ROM drive. Then you must direct *VisiCube* to the location of that file and describe the file (both its type and the structure of the data within the file).

By doing so, you have defined a new data source to *VisiCube*. That data source will now be listed in your data library and can be used in the creation of cube files.

Capture Your Data

Once you have defined all of the necessary data sources, you can capture that data (or any part of it) in any number of cube files. A cube file is the essential entity upon which *VisiCube* acts. Each cube file is completely self-defining and, therefore, can be moved between *VisiCube* installations at will. Stored inside a cube file are the cube data itself and any number of cubes built upon that data.

To create a cube file, you select the desired data source (or sources) and extract the desired data from those sources. The extracted data, along with a default cube for that data, are stored in the new cube file (which you name and place on your computer at any desired location).

Organize Your Data

Although it is not necessary, it is often desirable to further organize your cube data. Specifically, you can generate additional categories and measures in the cube data which are based on other values stored in the cube data. You can also define additional cubes. (Remember that *VisiCube* automatically creates a single, default cube for your cube data.) In defining these additional cubes, you can organize your data in myriad ways. It is useful to think of each cube as a separate view of your cube data or some subset of that data. It is likely that you will want to view your data in different ways. Cubes provide you that ability.

Analyze Your Data

Finally, you may analyze each cube in any cube file through the various visualization facilities provided by *VisiCube*. Each facility provides a different technique for examining your data and is completely dynamic. Each change to a visual is immediately reflected in the applicable *SynchroView* and remembered by *VisiCube*. Visuals are utilized to give you the best possible perspective on your data and allow you to analyze that data with minimal effort.

Saving of Changes

VisiCube uses a paradigm in which your work is always saved. I.e., you never instruct *VisiCube* to save your work. It is done automatically in all *SynchroViews*. Work can never be lost.

You also have the ability to copy objects, such as a cube file, at any time. This feature can be used to create a backup version of the object before you make changes that you are not sure you will want to save.

The Current Cube File

Most of the *SynchroViews* provide you the ability to manipulate a cube file. Because the assumption is that, no matter which *SynchroView* you are using, you wish to use it on a single cube file, *VisiCube* keeps that same cube file open until you explicitly choose to access a new one. The name of this current cube file is always displayed by *VisiCube* (in the control bar).

When you wish to open a new cube file, you select it from the *Cube File Library SynchroView*. Since all work on the prior cube file has already been saved, there is no intermediate need to define what you want done with that prior cube. The newly selected cube is immediately opened and made available to you.

When closing *VisiCube*, the current cube file is recorded. When you next open *VisiCube*, that same cube file is automatically opened. In fact, you are also returned to the last *SynchroView* you were using.

Clipboard Support

In addition to the standard approach of capturing source data from files on your computer, you may also create cube files based on data contained in your computer's clipboard. (The clipboard is the place where data is stored when you use the **Copy** feature of many software applications.) Specifically, you can use the *Import Raw Data From Spreadsheet SynchroView* to create a cube from spreadsheet data stored in the clipboard.

In fact, it is this specific feature through which spreadsheet data, such as that stored by Microsoft Excel, is supported. Data from a spreadsheet can be copied to the clipboard and then captured in a cube file via this feature.

Chapter 5 VisiCube Organization

In this chapter, we describe each of *VisiCube's SynchroView*s and how they fit into the big picture. Note that many of the specific details about the individual *SynchroView*s are not included here. (They are already available in the online help system.) The organization of this chapter matches that of the *Navigator* to facilitate cross-referencing between *VisiCube* and this manual.

Data group

This group of *SynchroView*s is where you manage the data sources from which you build cubes. In addition to identifying and organizing those data sources, you define the data within those sources.

For the present, *VisiCube* interfaces to the outside world of data sources by two means. The first of these is through *Microsoft Excel*. It is expedient to use *Excel* for manual data input (for which it is a very powerful facility) and therefore *VisiCube* has no machinery for this purpose. Raw data in the form of a spreadsheet table can be pasted directly into *VisiCube* and this table can be used to automatically generate a special re-useable data structure called the **raw data bank**. The second means is by extracting the data from what is called a **zoned file** - a text file with data arranged in columns - essentially a large table.

The **raw data bank** is an internal data structure that is re-used as. Although you can do a lot of things **to** the raw data bank, you can't do anything **with** it. Raw data is used to generate a new (or update an existing) **cube file** and, in *VisiCube*, you do everything with a cube file. You can have any number of cube files. They are the central subsystems of *VisiCube*, but their **foundation** is the raw data bank and the only path from a cube file to the outside world of data sources is through the medium of the raw data bank. The machinery of the raw data bank is explained in the descriptions of the SVs that follow, but the abstract concept of **raw data** (as it is used in the *VisiCube* universe) is explained in this section.

Data records were defined in the last section as **ordered sets** of **values**. Raw data should be always understood as a **collection** of such data records. A collection can, of course, consist of a single record, but in general there will be many records in a collection. It is convenient, as explained earlier, to think of this collection as a table (call it the raw data table) with the individual records being the rows and the record fields being the columns. You can imagine measurements being taken and data entered into these records by some worker. This worker need not be human—it can be an automatic recording device in a spacecraft or weather station for example. What is mandatory is that the data is entered into a record always in the same order.

The data values comprising the fields of the records are of two kinds, and it must always be specified, for each field (that is, for each column of the raw data table), which of these two kinds of data is represented. The two kinds of data are called **categories** and **measures**. If the data values of a field are qualitative, the field is generally considered to represent a category, and if they are numeric, the field would be a measure. Although it is usually clear that a particular field is a category, it is not necessarily true that a numeric field must always be a measure. Categories and measures are treated very differently in *VisiCube*. Each category and measure in a raw data table must be given a name.

Amongst the categories of a record, each name must be unique, and the same is true of the measures. However, a category field may have the same name as a measure field.

Categories that are named **YEAR**, **MONTH** or **DAY** are called **time categorie**s and play a central role in *VisiCube*. In future releases, the additional time categories HOUR, MINUTE and SECOND will be provided for. In any one raw data table, there can be only a single instance of a particular time category (for example, there can be a YEAR and a MONTH category, but not a YEAR and a YEAR category). The record data values for these categories must be numeric and meaningful (as years, months or days) or else they will be treated (automatically) as "missing data". Visualizations of data records that contain time category fields are called **time series** and are automatically presented as line plots.

When a raw data table contains time categories, then there must be a way to handle duplicate time values. In *VisiCube* this is accomplished by using the notion of a **station**. This is an abstraction of a general location. There must be a field—a category—in the raw data record that identifies separate subsets of records. These different subsets are called **stations**. The category that identifies the station is called the **station category**. If time categories are specified then a station category must also be specified. *VisiCube* will always enforce this. You cannot have one without the other. Of course, it is possible that duplicate time values can occur at the same station, but this would be erroneous data and it can be handled as such. An obvious example of an actual station is a weather station recording climatic data. Different weather stations can record data for January 3, 1980 and the data records can be in the same raw data table. Another example of a station is a store in a retail chain, with different stores recording sales data for the same day.

Zoned File Library SynchroView

Although zoned files (text files with data in fixed-width columns) need not be large, they often are. This largeness must be dealt with if they are to be examined successfully by means of a data microscope. It is possible for zoned files to span multiple storage devices, such as CDs or tapes, and a lot of machinery is needed to permit the extraction of a meaningful subset of data small enough to be efficiently visualized. The zoned file subsystem is, above all, a data **management** facility because the *VisiCube* machinery accomplishes its purpose by the successive extractions of zoned files from zoned files and the joining of them into other zoned files. In its current implementation, this subsystem consists of two SVs, the *Zoned File Library SynchroView* and the *Zoned File Extraction Facility SynchroView*. In terms of functionality, these are probably the richest SVs in the product. They are being evolved using an existing very large set of zoned files—the global climatic database of the NCDC (National Climatic Data center) of the NOAA (National Oceanic and Atmospheric Administration). These zoned files can be freely downloaded by anybody (some caveats apply) from their web site at www.ncdc.noaa.gov.

The **Browse** window is used to locate zoned files. Zoned files may be located on a fixed drive or removable media. Files of any type may be looked at by clicking on the file name in the listbox and using the **Open** button. Files opened in this manner are displayed in the **Zoned File** window. Keep in mind that only sequential zoned text files will make sense to look at. If the file opened is not a zoned file, this will become apparent when the **Zoned File** window displays disorganized text, or even what may appear as long sequences of random characters.

Once a zoned file is opened and verified in the **Zoned File** window as a sequential zoned file, the process of defining the data columns may begin. The **Defined Zones** window is used in conjunction with the **Zoned File** window to define and name data columns within the zoned file. The **Defined**

Zones window is used to define columns as category, measure or empty space using the text notation *c#=name*, *m#=name* or *s#*, respectively, where # represents the number of characters contained in a given column. Zone definition moves from left to right and defined zones are highlighted in the **Zoned File** window as column definition progresses.

When a column is defined as a category named *YEAR*, *MONTH* or *DAY* (i.e., *c4=YEAR*, *c2=MONTH* or *c2=DAY*), that column is flagged by *VisiCube* as a time category. As such, when a time category is present, it becomes necessary to define one of the zone IDs (*Z#*) in the **Defined Zones** listbox as the station category using the **Station?** field.

While defining columns in a zoned file, any of the elements in the **Defined Zones** listbox may be redefined; however, changes may not be made in this way after the file's defined zones are added to the library. At any point during the process of defining columns in a zoned file, you may **Clear** the **Defined Zones** listbox and start over. Once all data zones within the zoned file are defined, the **Defined Zones** window is used to **Add** the file to the zoned files library.

When a zoned file's data columns are defined and added to the library, that file is thenceforth listed in the **Zoned File Library** listbox. This window is used to manage defined zoned files, specifically. Removing a file from the listbox using the **Remove** button takes the file out of the list along with the column definitions, but does not remove the actual zoned file. **Delete**, however, deletes the file from the list and the system.

Files in the **Zoned File Library** listbox may be renamed (**ReName**), removed (**Remove**) or deleted (**Delete**). Removing a file does not delete it, but simply removes it from the window listbox. Deleting a file, however, actually deletes it entirely from the system. Notes may be provided for a zoned file, which appear in the rightmost column of the listbox. Notes can be useful for identifying particulars about a given zoned file. Since it is important both to know where files are located and also to conserve space, the option is made available to hide folder paths for files in the **Zoned File Library** window. When folders are shown, every file is listed with its path, when folders are hidden, then just the file name is listed.

Once a zoned file is defined in the library, additional information may be stored in relation to its measures. The **Measures** window is used to define the measure unit names and missing data values for the active library zoned file. Missing data are sometimes indicated by a specific numeric value in zoned files. This numeric value can be defined to individual measures. *VisiCube* will then treat occurrences of this numeric value for a given measure as missing data. Defining a unit name (i.e., Inches, Lbs.) to a measure allows the unit to be displayed in visuals with the measure values. The **Measure** window also provides a means by which unit names and missing data values defined to a measure may be copied to other measures.

Often, many zoned files archived from one data source will use the same structure. *VisiCube* takes advantage of this by using the zone definitions of the active (currently selected) zoned file in the library as a template. So, when the **Browse** window is used to open a zoned file, the template automatically overlays the opened file in the **Zoned File** window. To use the template, simply **Add** the zoned file. If the template will not be used, just **Clear** the template from the **Defined Zones** window and start from scratch.

Once a zoned file is defined in the library, its column definitions may be reused as a template, but not altered. Before adding a zoned file to the library, be sure the data zones are defined properly, or you will have to start over to fix a mistake.

Key File Library SynchroView

A category can have numerous values. Such values are frequently not associated with information that clarifies its meaning. For instance, in the case of weather data, when a category contains hundreds of station numbers, the station numbers do not speak for themselves with regard to their location or elevation. It can be useful to associate each station number with a location, or elevation. Doing so by hand would take a long time and be prone to human error. This is why a facility is included for creating such associations dynamically. Often, such data are created or distributed with zoned file tables that can be used to look up pertinent information in relation to a given category's values.

Key files are, in fact, zoned files. When defining a new key file, the **Browse** window is used to open a zoned file and the **Define Key Table** window used to define the key/value zones from the zoned file. The first 50 lines of the opened zoned file appears in the **Zone File** window. In the **Define Key Table** window the **Key** field is used to indicate the beginning and ending character positions of the key column from the zoned file. The positions are separated by a comma. For example, when the column begins at the first character and ends with the sixth:

```
1, 6
```

The **Zone** field is used to indicate the beginning character position and number of characters in the values column using the same notation. For instance, if the values column begins at the twentieth character position and the data zone is twelve characters wide:

```
20, 12
```

When either the key or value zone is defined in this manner, the **Zoned File** window updates by highlighting the key column yellow and the value column pink. When you are satisfied that the key and value zones are properly defined, you then **Build** the key table. This key table is displayed in the listbox. Use the listbox to verify that the table is without errors, then **Add** the key table to the library.

Files in the **Key File Library** listbox may be renamed (**ReName**) or deleted (**Delete**). Deleting a file deletes it entirely from the system. Notes may be provided for a key file, which appear in the rightmost column of the listbox. Notes can be useful for identifying why a key file was defined and how it is expected to be of use. Since it is important both to know where files are located and also to conserve space, the option is made available to hide folder paths for files in the **Key File Library** window. When folders are shown, every file is listed with its path, when folders are hidden, then just the file name is listed. By default, key files are stored in the My Key Files folder of `My Documents\VisiCube\`.

Once one or more key files are defined, they are used in ***Names/Labels*** to associate a category's values with labels. The active key file is the one currently selected in the **Key File Library** listbox.

Extraction group

This group of *SynchroView*s is where you extract data from one, or more, of your data sources and create new objects. Those new objects may be cubes or other data sources.

There are no facilities for manual data entry in *VisiCube*. Either the raw data already exists or you must use Microsoft *Excel* to build it and then import it into *VisiCube* from there. Currently there are two facilities for importing data into *VisiCube*. The first of these lets you copy the data from a table

of cells in a spreadsheet and then paste them into *VisiCube*. The second is a comprehensive system for acquiring data from sequential *zoned files*. Zoned files are read-only text files in which the lines of text data are organized into fixed width columns. These are widely used for storing large amounts of archived data. The word "zoned" has been coined to distinguish such files from so-called *delineated* files (sometimes called *comma-separated-value* or *CSV* files). CSV files will be supported in the future.

Import Raw Data From Spreadsheet SynchroView

This *SynchroView* initializes new datasets within *VisiCube* by using a spreadsheet application such as Excel as the vehicle for input to initialize the raw data file. Part of this initialization process includes the declaration of fields as measure or category. The raw data file initialized in this manner remains active throughout *VisiCube* until reinitialized.

Data are copied from a spreadsheet application such as Excel, then pasted using the **Create Raw Data File** window. An example file, Brain Size Datafile.XLS is provided with the your installation of *VisiCube* and is located in the folder: `C:\Program Files\VisiCube\SampleFiles\`. Opening this file reveals a data table demarcated by a thick border. Selecting the cells within this demarcation selects the field headers along with the data fields. Copying this selection places the cells into the copy buffer. At this point, the copied data may be pasted into *VisiCube* by using the **Paste** button in the **Create Raw Data File** window. A successful paste of the copied cells reveals a new window, **Specify Field Type**, with field IDs automatically associated for each field header, **Name**, from the copied cells. This window is used define each column as either a category, measure or key. After verifying that every column is defined as desired, the raw data file is initialized by using the **Create Raw Data File** window to **Create** (initialize) the new raw data file.

When data cells are pasted from a spreadsheet, *VisiCube* looks through each field for any non-numeric character. When a field contains a non-numeric character, that field is denied the option of being used as a measure. In the **Specify Field Type** window, the only options available for such a field are **Key** and **Category**, with **Category** set as the default. Fields containing only numeric characters, however, are assumed to be some manner of measure and are given the added option of **Measure** in the **Specify Field Type** window, with **Measure** set as the default. In the event a field of numeric characters is, in actuality, a category using numeric notation, it is simple enough to define the field as a category before creating the new raw data file. Notice that when pasting the cells from the example data file, Brain Size Datafile.XLS, every field is correctly assumed to be a measure except the first.

Once the raw data file has been created with your data, you can go to the ***Names/Labels*** *SynchroView* to rename measures and create meaningful labels for measures and categories. The ***Categorize*** SVs ***Measure*** and ***Category*** can be used to define categories from measures and break down large categories into classifications, respectively. The ***View*** *SynchroView* can be used to see the raw data at a glance and decide on any changes before finalizing the raw data. When the raw data is finalized, create a cube file in the ***Update / Create*** *SynchroView* in ***Cubes***.

Zoned File Extraction Facility SynchroView

Zoned files are often quite large and contain many categories and measures for many stations. The idea is to extract meaningful subsets from such files and investigate them visually for patterns of interest. The effect of attempting to visualize too much information can detract from rather than contribute to the understanding of data. When extracting to a new zoned file, make sure the **Extract**

Zoned File in the **Extraction** window is selected. This will cause an extraction to create a zoned file data subset which will automatically be stored in the zoned file library and selected as active.

When extracting records from zoned files, think about what you want to extract, then use the **Unions** window to describe meaningful subsets through unions. Subsets are described by defining which elements of one or more category to include when extracting records into a new zoned file. For example, in defining a range of weather stations from *320000* to *329999* from category #1 (C1) the union would look like this:

```
C1=320000-329999
```

To obtain records for these stations only for the years *1998* and *2000* from a year category defined as category #2 (C2), another union would thus be defined as:

```
C2=1998,2000
```

A zoned file extracted such becomes the starting point for a new subset. When defining unions, the **Categories** window can be used for reference.

Zoned files are often the result of an archival process. Naturally, such archives will potentially contain thousands of zoned files that are each structured the same. While any one zoned file from the archive may sometimes contain the precise data you wish to explore, this more than likely will rarely be the case. When data need be extracted across a series of zoned files from such an archive, the **Browse** window may be used in conjunction with the **Joined Files** window to select a series of files from which to extract. The **Browse** window is used to find zoned files and **Join** them to the extraction. In the event that a folder contains many such files, the may all be joined (**Join All**). Joined files are added to the **Joined Files** listbox, which may be removed (**Remove**) individually or all at once (**Clear**).

When archived zoned files are stored on a series of removable media such as CDROMS or tapes, extract from them one at a time. Because extracted zoned files are stored on the hard drive, extractions from removable media may be cumulative, using the extracted zoned file from each extraction as the new source file for the next series.

Because raw data is the precursor to a cube file, only the subset of data that will be visualized should be extracted to the raw data bank. Zoned files should be used to, in a sense, store larger data subsets for the purpose of extracting smaller subsets into the raw data bank. When extracting raw data, make sure the **Extract Raw Data File** button is selected in the **Extraction** window.

Depending on the size and count of the source zoned files, the extraction process can potentially take quite some time. The **Extraction** window may be used to **Stop** or **Pause** an extraction. When an extraction is stopped, it is halted altogether and may not be resumed. Just as the **Extraction** window is used to pause an extraction, it is also used to **Resume** an extraction that has been paused.

Cubes Files group

This group of *SynchroView*s is where you manage your cubes. In addition to organizing those cubes, you update and refine those cubes.

The logical concept of a hypercube exists in the form of a cube file. The cube file is automatically generated from the raw data bank. It contains all the raw data from its parent raw data bank, but with each different value of every category indexed. Although there is only one raw data bank (that is why

it is called *the* raw data bank), there can be any number of cube files and these cube files can be saved and exchanged between *VisiCube* users.

The ***Cubes.Update / Create*** *SynchroView* is used to create a **cube file** from the raw data file. Unlike the raw data file, which is an internal file belonging to the *VisiCube* system, cube files can be freely interchanged amongst users of *VisiCube*. All the data records that were in the raw data file are stored in the cube file (but more compactly). The cube file is also provided with a multi-dimensional database that links every distinct value of every category to a chain of those records that contain them. Taken together, this machinery is an implementation of an abstract logical concept called a *hypercube* (hence the name *VisiCube*)**:**

Each category of a cube file has a certain number of possible values. Any record in the cube file contains one possible value in each of its category fields (it is possible that the field has *no* value, but this attribute - "*missing*" - is considered to be a valid value). Each category can be thought of as an *edge*, divided into as many equal parts as there are possible values in that category. If there are 3 categories, then the resulting 3 edges can be thought of as an actual physical *cube*. However, there generally may be more than 3 categories, in which case we say that the resulting set of edges form a *hypercube*. We can't draw it, but this does not matter; the concept of edges divided into possible values is unchanged. Now, the category values of a particular record can be looked at as providing *coordinates* equal to those values in each edge. This set of coordinates defines a particular location in the hypercube. This location may be thought of as a *cell*. There may, in general, be other records having the same coordinates, thus defining the same cell. The chain of record numbers for the records having the same coordinates is the chain linked to that particular cell of the hypercube. It is easily possible that a set of coordinates may not match *any* record. Such a cell is *empty* - it has no record numbers linked to it. Cubes consisting mostly of empty cells are called *sparse*. Sparseness is a common condition, especially if the cube has a large number of dimensions (categories). *VisiCube* does not actually use this full cube, but rather a kind of "fuzzy cube" consisting only of the non-empty cells, so sparseness is not an issue. Each non-empty cell should be thought of as defining a subset of all the records in the cube file.

Files group

The following *SynchroViews* provide you the ability to manage your cube files.

Cube File Library SynchroView

This *SynchroView* is used to manage cube files. Such management includes file selection and manipulation, keeping notes and maintaining a list for commonly used cube files. Also, the raw data file may be initialized from this *SynchroView*.

The cube file selected in the **Cube Files** window is active throughout *VisiCube*. This means that every *SynchroView* that operates on a cube file will use this file. Every such *SynchroView* will list the path and name of the active cube file using yellow text in a black bar near the top of the *SynchroView*'s primary window. To quickly see an instance of this, select the `Colleges.VCB` file in the **Cube Files** window, then visit the ***View*** *SynchroView*.

Copying and renaming cube files is trivial using this *SynchroView*. The **Browse** window is used to navigate to a folder where the active cube file will be copied. For instance, select C:\ in the **Browse** window; notice how the **Copy To** field in the **Cube Files** window updates with this location. Likewise, this field updates accordingly as folders are navigated using the **Browse** window.

Renaming the active cube file is as simple as modifying the text in the **ReName** field of the **Cube Files** window and hitting enter.

When the active cube file is no longer needed in the **Cube Files** window, but may be needed at a later time, it can be **Removed**. **Deleting** a file, however, will remove the active cube file entirely from the system.

When a cube is created from raw data, it is automatically added to the cube file library. The need to manually add cube files to the library occurs when re-adding a cube file removed from the list or accessing a cube file shared by a colleague. The **Browse** and **Add** windows work together toward this end. The **Add** window displays cube files that may be used with your current version of *VisiCube* in the folder currently active in the **Browse** window. If such a file does not exist in this folder, then the **Add** window is empty. A quick way to see this behavior is to browse to the *VisiCube* sample files folder by following this path: C:\,then Program Files, then *VisiCube*, then SampleFiles. The **Add** window will then display a list of sample files automatically included with your *VisiCube* installation. By default, all of these files should already be added to the **Cube Files** window, and thus marked **YES** in the **In File List?** column of each file listed. Files not listed in the **Cube Files** window will be marked **NO**, but can be added individually (**Add**) or enmasse (**Add All**). As a shortcut, the **Browse** window can be used to **Scan** the system for cube files; all that are found are listed in the **Add** window.

Notes for files are viewed in the **Cube Files** window. This is a convenient way to keep track of information pertinent to a given cube file. Many of the sample files provided with your *VisiCube* installation have notes that summarize the data as well as its source and ownership. The **Cube File** window may be widened or scrolled horizontally to see longer notes. Notes for the active cube file may be added or modified by using the **Notes** field. Notes are limited to 200 characters in length.

Click the **Extract** button in the **Cube Files** window to initialize the raw data file with data active cube file. For instance, select `Colleges.VCB`, click the **Extract** button, then go to the ***Raw Data.View*** *SynchroView*. Data from this cube file will be displayed, showing that the raw data file is using data extracted from `Colleges.VCB`.

Categorize group

Additional categories can be created dynamically from existing categories and measures within the raw data bank. The process of defining a category is called categorization. When you categorize a *measure*, you are essentially dividing the range of values of a measure into a set of sub-ranges that can be individually named as the values of the new category. When you categorize a *category*, you are actually creating subsets of values in which a higher-level category is defined in terms of named sets of values from the underlying category. You will quickly notice, the moment you start playing with *VisiCube*, that categories and category values are “the stuff that makes the world go round” and that you are limited in what you can do without them.

Categorize Measure SynchroView

This *SynchroView* is used to create a category by using a measure as the basis for the new category. An example of this exists in the `Brain Size.VCB` cube file, where the **Brain Size** measure is defined to a category of the same name with the values **Small**, **Medium** and **Large**. The effects of this may best be viewed in the ***View*** *SynchroView* of ***Cubes*** by selecting the **Brain Size** category then selecting its values to see how records correspond.

Lets use the `Brain Size.VCB` cube file provided with your installation. Initialize the raw data from this cube file by selecting it in the ***Files*** *SynchroView* of ***Cubes*** and extracting it with the **Extract** button, then return to this *SynchroView*. The measure **FullScale IQ** is automatically selected. While any other measure in the **Measures** window may be selected for categorization, the default is fine for the purposes of this example. In the **Sorted Data** window, select both the listing for the values **90** and **133** from the data column in conjunction with records **34** and **31**, respectively. Three category values are then automatically defined in the **Categorize** window for data ranges *77 to 89*, *90 to 132* and *133 to 144*. Now, one at a time, select each value in the **Categorize** window, top to bottom, and change their values to **Low**, **Average**, and **High** by modifying the text in the **Chg Value** field and hitting return. Then **Apply** the changes. A new category is created called **FullScale IQ** that may be viewed in the ***View*** *SynchroView* in either ***Raw Data*** or ***Cubes***.

Categorize Category or Key SynchroView

This *SynchroView* is used to create a category by using an existing category as the basis for the new category. This can allow a category with numerous values to be simplified into one or more new categories with fewer values. In this manner, category values that span a limited number of records can be coalesced into values that span a greater number of records. An example of this exists in the `Colleges.VCB` cube file, where the states in the **States** category have been coalesced into six regions in the **Regions** category. The effects of this may best be viewed in the ***View*** *SynchroView* of ***Cubes*** by selecting the **States** category, and observing the effect of selecting its various values, then selecting the **Regions** category and observing the same effect.

Structure group

The following *SynchroViews* provide you the ability to supplement the definition of the cube data stored in a cube file.

View Cube SynchroView

This *SynchroView* is used to look at the data records in the active cube file. This *SynchroView* is different from the ***View*** *SynchroView* in ***Raw Data*** in that the only records shown are those that match the selected value of the active category.

Viewing records by category value allows records to be represented as a slice of the multi-dimensional cube file. For instance, make the `Livestock.VCB` cube file active, then return to this *SynchroView*. Selecting **country** in the **Categories** window and then its value **Denmark** in the **Values** window causes the **Records** window to show only the records that contain the **Denmark** value in the **country** category. Likewise, selecting the **livestock** category, then selecting its **Poultry** value causes only records to be displayed that contain the value **Poultry** in the **livestock** category. In this manner, a sense of the cube file and how its data is cross-sectioned by the categories and their values may be obtained.

The **Records** window displays one measure or category field at a time, which is the field selected in the **Selected Record** window. When the active cube file contains a key, the key will be displayed alongside this selected field by default. Viewing the `Colleges.VCB` cube file in this *SynchroView* does a good job of exemplifying this behavior. The reason records are displayed in this manner rather than displaying them all at once is so that more information may be viewed in a compact space.

Change Names and Labels SynchroView

A measure without a unit can be difficult to make sense of. As well, a category value without an informative label can also be difficult to understand. This facility is provided to allow units to be specified for measures and labels for category values. Also, sometimes measures or categories are cryptically named, these may be renamed here as well.

When the names of measures are defined in meaningful ways, then plots created from this information become easier to read and understand. For instance, in a scatterplot, measure names are displayed along each axis. Likewise, when a unit is defined to a certain measure, the unit is displayed in plots alongside the measure name in parenthesis such as to clarify the meaning of the measure values. For instance, when looking at a scatterplot of the `Brain Size.VCB` file the **pixels** label is listed next to its measure name, **Brain Size**, in parenthesis:

```
Brain Size(pixels)
```

In any visual plot that uses categories, the category may be selected in some fashion to visualize measure values in relation to the values of the selected category. For instance, looking at a quantile plot of the `Brain Size.VCB` file will show a set of two, or three, quantile plots depending on whether the **Gender** or **Brain Size** category is selected. In the case of **Gender**, two plots are displayed for the values **Female** and **Male**. These values are displayed above of each plot. When a label is defined to a category value, the label will be displayed instead. The label is used when it may be useful to clarify the nature of the category value. For instance, the `Egyptian Skulls.VCB`, file uses labels for the category values in **Year** so as to specifically identify years to **BC** or **AD**.

The elements of a measure may be updated within the **Measure Names and Units** window. It is as simple as selecting a measure, modifying the text in the **Chg Name** or **Chg Unit** fields and hitting return. The elements of a category are updated between two windows, **Category Names** (for changing the name) and **Value Labels** (for changing value labels). These work in the same fashion.

A key file defined in ***Key Files*** can be used to propagate the label fields of a category containing many values. The active key file's table is listed in the **Selected Key Table** listbox. Keys in this table relate to a given category's values. You propagate a category's labels with the labels from this table by selecting a category and then using the **Selected Key Table** window to **Update** the labels.

Outlier Facility SynchroView

Under construction! We will provide a description of this SynchroView with the next release.

Cubes group

As the number of categories and measures in a cube increase, the number of possible visualizations of the data can become very great. To organize this diversity, any number of **datasets** can be defined for a cube. A dataset is a subset of the set of all records of a cube and is defined in terms of the categories of the cube. It is the datasets that are visualized, not all the data in the cube at once (but it is legitimate to define a dataset as being all the data).

Besides the raw data records and the hypercube machinery (which are automatically generated), the cube file also contains dataset definitions (which you specify). A dataset definition defines a dataset, which is a subset of all the raw data records.

Data visualizations are always of *some* dataset. The particular dataset being used is specified by *some* dataset definition. Thus, the cube file must always have at least *one* dataset definition in order that its

cube can be used. To insure this, a *default* dataset definition is provided when the cube file is created. This dataset definition specifies the dataset consisting of *all* the data records.

If there are only a few records, this default dataset might be all that is ever needed**:** you always work with all the records. In general, you will want to work with different subsets of all the records. You can specify as many dataset definitions as you want, each one of which defines a particular subset of all the records.

You specify a dataset definition in terms of the category fields and their values. An example of a simple specification is **{**Color = Red**}**. This set definition specifies all those data records for which the value of the field *Color* is *Red*. A more complex definition might be **{**Color = Red**}** OR **{**Color = Blue**}**. Another might be **{ {**Color = Purple**}** OR **{**Size = Small**} }** AND **{**Sex = Male**}**.

Depending on the type of visual, datasets consisting of more than about 1,000 data records may result in over-crowding. Some types of visuals will be able to handle much higher densities, but probably 10,000 data records would be a reasonable upper limit - for a single visual. *VisiCube* uses a number of devices to handle crowded visuals conveniently, but the best plan of attack is to break a lot of data down into a lot of visuals ("divide and conquer").

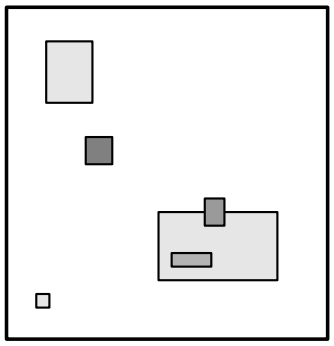
You can define any number of datasets within a cube file, and to help with this you can usually define additional categories if there are many data records. It is the individual datasets that are visualized, not all the data at once, and *VisiCube* makes it very easy (effortless, really) to switch from one dataset to another. *VisiCube's* dataset machinery is one of the reasons why it is called the *Data Microscope* - a small dataset is very like a microscope slide with a tissue section from a source too large to be examined conveniently as a whole.

Keep in mind that the definition of a dataset (in terms of the cube's categories) already provides very specific information about the nature of this subset of records, which reinforces the further information presented by the dataset's visual.

What about really large data sources, such as databases and "flat" files with millions or billions of records? For now, you must be content with extracting what you can of these into Excel and starting from there. Later, *VisiCube* will have facilities for dealing with these data sources directly.

With apologies to Gaius Julius Caesar, *VisiCube* also uses the strategy of "divide and conquer" to bring sense and order to the visualization of data. What are being divided are sets of records (where a "record" has been described earlier in the topic *data records*) and these sets are being divided into *subsets* of records. It is very important that these subsets be meaningful subsets that you can make use of in an interesting and informative manner. It is important because there are a lot more meaningless and useless subsets than useful ones. This may come as a shock to those who have not delved into set theory before, but a set of only 100 records can be divided into more than 100,000,000,000,000,000,000,000,000,000,000,000 different subsets! A set of 1,000 records can be divided into more subsets than there are electrons in the known universe.

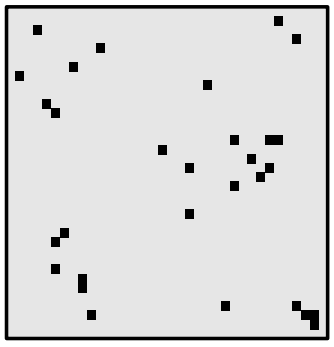
Our starting point will be a hypothetical set of records that is conveniently called the universal set. This might be a huge "flat file" (think of a reel of tape) containing millions or billions of records, or it might be a large relational database containing hundreds of tables (of records). It also might be a little spreadsheet table containing 10 records, but to get the full picture imagine a big universal set that can be represented by the following diagram**:**



The small rectangles represent a few subsets of records (assumed to be meaningful!) out of the infinite number of subsets in the universal set. These subsets represent the first division of the universal set into useful and manageable parts that can be further broken down and examined by the *VisiCube* engine. You can imagine extracting such subsets using SQL (Structured Query Language) or other means. For now, this has to be done outside of *VisiCube*, but this will not be the case in the near future. Imagine, in particular, that you have extracted the subset represented by the colored rectangle and imported it into *VisiCube* as a raw data file. We will call this set of records Subset X.

All the above diagrams are variants of what are called "Venn" diagrams. These are universally used to abstractly represent unions and intersections of sets. Venn diagrams are very powerful because their logic is precisely accurate. However they can distort reality considerably and it may be worthwhile to redraw the previous diagram showing things more as they really are. This new diagram is still not totally realistic (it would take a bigger drawing and lots of work to make it so) but it does illustrate how disconnected real sets are.

Subset X



The Universal Set

This diagram represents Subset **X** as a 36 X 36 matrix with each cell representing one record. This comes to 1296 cells, whereas Subset X actually contains 1302 records, but this is close enough. Assume that the records (each one representing a college) are ordered in the same way as in the cube file, proceeding left-to-right and top-to-bottom within the matrix. The black-colored cells represent a hypothetical Dataset **Z**. No attempt has been made to make the number of records in this subset consistent with Dataset **Z** as shown in the first diagram. This scattering of black squares is **one** of that enormous number of possible subsets of Subset **X** (more than all the electrons in the known universe).

Select Dataset SynchroView

A cube file contains within it a set of records—the same set of records that made up the raw data file from which it was created, but in a more compressed form. You can specify a *subset* of this set of records using the machinery in the **Datasets.Define** *SynchroView*. This specification is called a dataset, although it is actually a *definition* of a dataset (the actual records of the dataset are found amongst the single set of records in the cube file). You can define as many datasets as you wish. This *SynchroView* lists all the datasets defined for the active cube file. It lets you manage this list by either deleting datasets from the list or creating duplicates by taking a **"snapshot"** of the currently selected dataset.

The visual state of a cube is the particular configuration of visuals that were manifest the last time this dataset was used. This configuration or *state* is stored in the dataset and becomes current the next time that dataset is used. When a dataset is first created (by means of the **Datasets.Define** *SynchroView*), every possible visual parameter (such as the particular measures used in a scatterplot matrix) is given a default initial value. This is an extremely important design feature**:** it means that you will be "up and running" with a newly created dataset no matter which *SynchroView* you bring up within *VisiCube*. As you use the dataset (for example, brush a set of points) the *state* of the dataset changes and is always remembered. The dataset *definition*, once the dataset is created, cannot be changed, but the dataset *state* changes constantly as you work with the dataset within *VisiCube*.

The state of a dataset can be "frozen" by taking its "snapshot". Simply press the **Snapshot** button and an exact duplicate of the currently active dataset (its definition and its visual state) will be made using a unique ID and added to the list of datasets of the cube. You should describe the snapshot in the **Notes** text box for future reference. The purpose of taking a snapshot is to freeze its visual *state* (its definition is always frozen) and normally you would not make any changes to this state (presumably you are saving its state for future reference or display), but *VisiCube* will not prevent you from changing a snapshot's visual state (it treats it as any other dataset of the cube).

You can browse the datasets of the cube by clicking the list of datasets. A quantile plot for a selected measure of the cube is displayed to let you visualize the data distribution (of that measure). You have the option of selecting any measure to be used for the quantile plot. Of course, at any point, you could invoke a particular *SynchroView* to actually see that dataset in action. To aid in browsing the list of datasets, you should annotate the list with notes. You can enter a note of up to 200 characters describing any dataset. You can also **Delete** the selected dataset from the cube. The ID numbers of deleted datasets are not re-used later. When new datasets are defined, an always-increasing ID number is assigned to it for the cube.

Define Dataset SynchroView

A cube file contains within it a set of records—the same set of records that made up the raw data file from which it was created. While you can visualize the data from this entire set of records at once, this may not result in clearest view of what is going on in this data if there are a large number of records. In general, you will want to visualize the data from a smaller number of records, a subset of the original set of records. At this point you may want to review the section *Some Set Theory*, presented earlier in the user guide. This *SynchroView* lets you define a subset of the original set of records, in terms of the cube's categories and their values, entirely by selecting and clicking.

The simplest subset consists of all the original records. The *Define Dataset* window of this *SynchroView* displays "**All Records**" when it is first invoked with no other subsets specified. If this

is the dataset definition you want, click the *Add* button and you are done. Since a cube file is created with this particular dataset already appended to it, it may seem unlikely that you would want to define a second such one and append it to the cube file. However, especially if the original number of records is small, you may very well specify many such "all records" datasets because as well as specifying a subset of records, a dataset maintains a unique visual environment within the cube. (See the *Select Dataset SynchroView* for more information.)

You will define a subset of records by first defining what are called atomic subsets and unions. An *atomic subset* is specified as **{Category = Value)**. A *union* is specified as, for example, **{Category = Value1, Value2, Value3}**. The values are all from the specified category. The example shows 3 values. You can specify up to **10** values for a union. It may be that a single atomic subset or a single union is enough to specify the subset you want. In this case, you are done and can add this dataset to the cube's collection. In general, however, you will specify several of these (subsets, unions or both) and then use them to define binary subsets. A *binary* subset is specified as **{subset OR subset}** or **{subset AND subset}**.

As a simple example, using the *Colleges.VCB* cube file, you could define the union **{State = CA, IL, NY}** and the atomic subset **{Region = SouthEast}** and then define the binary subset **{S1 OR U1}**, where **S1** designates the atomic subset and **U1** designates the union. These **"S"** and **"U"** designators are used as IDs in the *SynchroView*'s interface, but the **"{ }"** expressions are expanded in full as you build them up by selecting and clicking.

If the active cube file has no categories, the only dataset you can specify is **"All Records"**. If this cube file has a very large number of records, you always have the option of extracting its raw data file, categorizing a measure, and then updating the cube file with this new category, from which you can then define proper subsets.

Visuals group

This group of *SynchroView*s is where you visualize the data in your cubes.

In *VisiCube*, the visuals are interacting sets of plots and supporting windows (together called synchro-views, as explained in the next section) that let you look at the datasets in a highly dynamic manner. The plots themselves are made up of a few kinds of "primitive" graphs (such as dot plots and scatterplots) but these are built up into more elaborate constructions.

Visuals are the graphical representations of the data in a dataset. By one means or another they must let you see the data and its patterns and relationships as clearly as possible. To do this, *VisiCube* uses only five "kinds" of graphs**:** scatterplots, quantile plots, box plots, dot plots and line plots. All of these are of the flat "2-D" variety and constructed as simply as possible. These graphs are used individually as *primitives* for constructing more elaborate higher-level structures. The design and workability of these structures depend greatly on *VisiCube's* ability to categorize and proliferate meaningful subsets of data.

The landmark book, *The Elements of Graphing Data* by William Cleveland of Bell Labs (Hobart Press, 1994), is the inspiration for many of the graphic ideas used in *VisiCube*. Chapter 4 of this book, *Graphical Perception*, has been and will continue to be used as the authoritative reference for the design of the product's visuals. The chapter section *Pop Charts* should be read by anyone who wonders why *VisiCube* does not provide pie charts and bar charts.

An example of a higher-level structure is the multiway plot, a visual constructed out of the elementary dot plot. A dot plot is actually a simplification of the very widely used (outside of *VisiCube*) horizontal bar graph. The dot plot is a horizontal bar graph with the "bar" missing. If you play with the multiway plot for a few minutes, using the sample `Colleges.VCB` or `Livestock.VCB` cube files, you should see that this visual is an elaborate and dynamic universe in itself, far transcending the dot plots it is made out of. You should also see that substituting a bar chart for the dot plot would greatly diminish its effectiveness, if not turn it into a mess.

There is a major innovation used throughout *VisiCube***:** the absence of the standard tick marked scales. The data area of any graph is always fitted exactly to the range of the data values used in that graph and the minimum and maximum values are displayed at the corners. This is based on an idea of Edward Tufte (*The Visual Display of Quantitative Information*, Graphics Press, 1983). The data area is divided into 4 or 16 equal-sized regions by drawing a faint grid of 1 or 3 pairs of intersecting lines (depending on the size of the graph). This allows the distribution of the data to be immediately evaluated.

Conventional tick marked scales are a holdover from the days (long before computers) when scales were intended to allow determination of data values. The scales are of marginal use for this unless the density of tick marks is so great that the graph looks silly. In practice, taking reasonably accurate measurements off of a printed chart is fussy at best. In *VisiCube*, when looking at charts interactively, totally precise measurements of all data points are always instantly available. The actual data range is certainly of interest and is always displayed precisely in *VisiCube*. The elimination of tick marked scales greatly reduces clutter and has a lot to do with the clean appearance of the product's visuals.

Also reducing clutter is another innovation**:** only 3 or 4 significant digits are ever used in a data value displayed on a graph. A standard "power-of-ten" notation is used to indicate the magnitude of the scale in a separate part of the graph, or in the window containing the graph. This power of ten is always a multiple of 3, indicating *thousands* or *millions* or *thousandths*, etc. The decimal point in the displayed number is adjusted to make this possible. Thus 315,275 is displayed as 315.3 and **{E+3}** and 0.062247 is displayed as 62.25 and **{E-3}**.

The basic colors used in all visuals are black and shades of gray on white backgrounds. Any other color *always* encodes something. Color is too valuable an information-containing device to throw away on mere decoration. When thus used sparingly and meaningfully the effects are sometimes striking (*form follows function*).

One of the most important uses of color in *VisiCube* is to visualize subsets of data. The maximum number of different subsets that can be visualized on any one plot is six. This procedure is called *categorizing a dataset* and a *SynchroView* is devoted to it (this is unrelated to the ***Categorize.Category*** *SynchroView*). The six colors used have been carefully chosen and cannot be changed by the user.

Univariate group

The following *SynchroViews* provide you the ability to visualize your data one measure at a time.

Cube group

Univariate Dataset Records SynchroView

Under construction! We will provide a description of this SynchroView with the next release.

Quantile Plots group

Single Quantile / Box Plot SynchroView

Under construction! We will provide a description of this SynchroView with the next release.

Multiple Quantile Plots SynchroView

Under construction! We will provide a description of this SynchroView with the next release.

Box Plots group

Box Plots SynchroView

Under construction! We will provide a description of this SynchroView with the next release.

Multivariate group

Visuals in the ***Multivariate*** SVs are designed to allow visualization of distribution and relationship between two or more measures. The method employed to accommodate this is the scatterplot. Each of the ***Multivariate*** SVs use the scatterplot as a sort of primitive, elaborating from it when needed. A variety of methods are implemented for making the scatterplot as meaningful as possible to view. Such methods include *categorization*, or grouping of points, *brushing* and *jittering*. Categorization is used in ***Multivariate*** SVs to identify values correlating with existing categories by color or pattern. Brushing is used to select, independently or by range, datapoints of interest so that they stand out in all other ***Multivariate*** SVs. Jittering is used to add noise, or gently scatter, overlapping datapoints in measures with redundant values so the redundancy doesn't hide the true nature of data in visuals. The most elaborate *SynchroView* in ***Multivariate*** is the ***Layers*** *SynchroView*, which allows the values to be studied between three measures at a time.

Because any given point of interest is directly associated with a particular record, the ***Records*** *SynchroView* is provided for looking at a particular record's values. This theme is consistently repeated throughout *VisiCube*. Regardless of the plot type, a *SynchroView* will always be present to allow the records of a value to be looked at in relation to a given data point.

Cube group

Multivariate Dataset Records SynchroView

This *SynchroView* is used to select and look at record values in relation to their corresponding data points in the scatterplot. Clicking on a data point highlights both that data point and its record in the **Dataset Records** window. In turn, selecting a record from the **Dataset Records** window highlights its data point in the scatterplot. When data points overlap, this can sometimes be the only way to highlight a particular data point. The scatter plot used in this *SynchroView* is made active in the ***Scatterplot*** or ***Scatterplot Matrix*** *SynchroView*.

All values of the record highlighted in the **Dataset Records** window are listed in the **Selected Record** window. Records are ordered in the **Dataset Records** window by record number. In this window, a column is displayed for **Record** numbers and the measure or category highlighted in the **Selected Record** window. When a cube has a key, a column for the key is displayed in addition.

Records that miss one or both values for the vertical and horizontal measure used to define the scatterplot will not be listed in the **Dataset Records** window. This is because such a record could not relate to a data point in the scatterplot, thus making it impossible to highlight a data point by selecting a record.

Categorize Dataset SynchroView

Under construction! We will provide a description of this SynchroView with the next release.

Scatter Plots group

Scatterplot SynchroView

The measure values selected from the **Vertical Measure** and **Horizontal Measure** windows define the scatterplot. Vertical and horizontal measures selected in this *SynchroView* stay active throughout the ***Visuals.Multivariate*** sub component, thus making the defined scatterplot itself active throughout the SVs therein.

The main idea behind this *SynchroView* is to look for patterns, or the lack thereof, in the distribution of data points. When a scatterplot reveals a distribution of particular interest, the other ***Multivariate*** SVs may be used to study the relationship of the data. For instance, the ***Brushing*** *SynchroView* may be used to highlight specific data points so that their records stand out in other distributions, and the ***Layers*** *SynchroView* may be used to compare the scatterplot against a third measure.

Scatterplots are explicitly defined by record values. So, each point in the scatterplot relates to a record that contains values for the vertical and horizontal measures. The ***Multivariate.Records*** *SynchroView* is used to look at record values in relation to specific data points.

Scatterplot Matrix SynchroView

This *SynchroView* is used to create and study a scatterplot matrix of up to ten measures. The measures are selected in the **Select Measures** window and then plotted using the **Plot** button. The **All** button is a shortcut to selecting all measures in a cube, unless more than 10 measures exist, in which case, the first ten measures are selected. The **Clear** button clears all selections.

The way to read a scatterplot matrix is by row (horizontally) or column (vertically). One measure is studied at a time in relation to other measures in the matrix. In this manner, the relationship of a single measure may be understood throughout a series of distributions. This effect may also influence which measures will be included or excluded from a particular plot of the scatterplot matrix. The `Colleges.VCB` cube file offers the most measures out of the sample files with which to study this effect.

It may be useful to note that the series of squares rising diagonally from the bottom left to the top right of the scatterplot are each a label for the row and column they join. This label will always provide meaningful information about its measure. For instance, if the measure is jittered, this will be noted, and if a unit is defined, this will also be indicated. Measures trade their horizontal and vertical axis on either side of the diagonal line of labels. This effectively makes the scatterplot configuration on both sides mirror images.

Clicking on a scatterplot in the scatterplot matrix makes it active throughout the ***Visuals.Multivariate*** SVs, including the ***Scatterplot*** *SynchroView*. In this fashion, the ***Scatterplot*** *SynchroView* can be used to see a close-up of a given scatterplot within the matrix. As well, other ***Multivariate*** SVs such

as ***Brushing***, ***Categorize*** and ***Jittering*** can be used to influence the nature of distributions throughout the scatterplot matrix, especially along a given row, or column.

Brushing group

Brushing By Selection SynchroView

Under construction! We will provide a description of this SynchroView with the next release.

Brushing By Intervals SynchroView

Under construction! We will provide a description of this SynchroView with the next release.

Presentation group

Jittering SynchroView

This *SynchroView* is used to add *noise* to one or both measures of a scatterplot in order to reveal overlapping data points. This effect is called jittering. Data points overlap when two or more records contain the same measure values in both measures used in the scatterplot. Adding noise to a measure does not change record values, but randomizes display points in relation to their values. When two or more points overlap, the effect of this randomization introduced by noise disperses visual points along the vertical and/or horizontal axis in relation to their actual values, causing hidden points to shift into view. Jittering introduced in this *SynchroView* persist through the visual plots of every *SynchroView* in the ***Visuals.Multivariate*** sub component.

In the **Jittering** window the scale of added noise may be set from **0** to **4** for the horizontal and/or vertical axis. Initially, **0** will be set, indicating that no jittering is present. The higher the number, the greater the dispersal of visual points along the axis in relation to their values. The `Egyptian Skulls.VCB` cube file does a good job of demonstrating the various distributions created by the different levels of noise. Make sure this cube file is active, then in the ***Scatterplot*** *SynchroView* (***#43***) set the active measures as **M2** and **M4** (**Basibregmatic Height** and **Nasal Height**), then return to this *SynchroView*. Selecting **1** causes overlapping points to disperse tightly clustered around their values, selecting **2** loosens the clustering a little more, and selecting **4** causes the most dramatic dispersal.

Jittering is most useful when many data points overlap. Data become visually misrepresented when only one point may represent a given coordinate shared my a number of records. The noise added to a measure creates a visual distortion of the actual values in such a way that actually clarifies representation by allowing data points hidden in this manner to come into view.

Though this *SynchroView* does not provide means for looking at the records and measure values in relation to a specific data point, other SVs do. *Layer Trivariate Data SynchroView*, *Brushing By Selection SynchroView*, *Brushing By Intervals SynchroView*, and *Univariate Dataset Records SynchroView* each provide a means of viewing records in relation to specific data points.

Every time a scatterplot is redrawn, jittered values will change location on the axis of the jittered measure. This is because these visual points are randomized in relation to the scale of noise introduced each time the a plot is drawn.

Layer Trivariate Data SynchroView

Under construction! We will provide a description of this SynchroView with the next release.

Time Series group

Any data that can be presented in a serial fashion - with the data records sequentially ordered in a meaningful manner - is a candidate for visualization as a **time series**. You can think of the records as being ordered from left to right (horizontally), or from top to bottom (vertically). Each record can be imagined as corresponding to the value of a **time variable**. A time variable takes on the values of a time category of the cube, as explained below. Currently, a time variable can only represent a **date** in *VisiCube*, but in the future its use will be greatly expanded. The reason for initially handling only dates is that each different kind of time variable will be handled in a manner unique to that kind, and the machinery for handling them needs to be built up gradually.

The active cube must have at least one **time category** specified in order for the time series SVs to be entered. A time category is a category having a reserved word name such as **YEAR** or **MONTH** or **DAY**. In this initial implementation of the time series visuals, these are the only time categories handled, and they are further restricted to just two configurations**:** (1) YEAR, MONTH, DAY (all must be specified), and (2) YEAR only specified. These restrictions will be removed shortly, and additional time categories such as **HOUR**, **MINUTE** and **SECOND** will become available. The climate database used as an example requires only the first two configurations, as will be the case for a very large number of possible applications. Time categories *must* be specified at the raw data level and cannot be specified once the cube has been generated. The specification of time categories is described in Section 1 Raw Data.

The simplest time series visualize a single measure. Although this is the simplest situation, it is still far from trivial and a lot of architecture is required to handle all the possible nuances. By deliberately concentrating on the single-measure case, a proper foundation can be constructed before launching into the multiple-measure case (which offers possibilities of visualization and dynamic analysis heretofore untapped by any software). There are two potential difficulties that must be addressed by programs that let you look at time series data and these are as clearly manifested in the univariate case as in the multivariate case. Solving them in the simpler case is an obvious first step in the evolution of the *VisiCube* time series facility. These difficulties are discussed next.

Data records that have a time category must deal with the fact that different records may originate from different sources, but all records must use the same time (included as the time category value of each particular record). *Time* is, after all, the 4th dimension of *space-time*, the coordinate system of our universe. This fundamental nature of time is immediately seen in the climate data, where the data records originate from thousands of different stations, the data input by thousands of different people, speaking innumerable different languages, but all using the same time coordinate. The totally obvious notion of a weather station has been incorporated into *VisiCube* as the abstract concept of a **station** - a fixed point in *space-time* that uniquely identifies any record. Notice that the point in *space* need not be precisely identified as to location**:** the important thing is that it identifies a particular set of records - those associated with this abstract *station*. As one example, a particular *store* in a retail chain (such as *Sears*) would qualify as a station, with nation-wide sales records all identified by the store at which they originated. Another example would be a particular *server* in a network of servers, where internet traffic data is being recorded at the same time throughout the network. In *VisiCube*, time-series visualizations always take place at **some** station. But it should be mentioned at this point that other kinds of visualizations (such as a multivariate scatterplot) are not at all subject to this restriction and can easily represent data from different stations in the same plot.

VisiCube always sorts the data for each station by the values of the time categories before using it. During this process it detects all duplicate and missing time values. While most automatically recorded data (such as point of sale data in a store) will never have duplicate time values, it can easily have missing time values (for instance, a store may be closed on some days). Some kinds of data, especially archived data accumulated from multiple sources, may have many duplicate time values. The NCDC historical monthly rainfall and temperature data, a sample cube for which is shipped with the product, has a large number of duplicate time values at many stations. *VisiCube* indicates both of these conditions in a precise and non-ambiguous manner as a part of the time-scale provided with the visualizations. Details are provided in the explanations of the SVs further on. The present machinery for this purpose is a preliminary for the more dynamic procedures that will be available in the next release. These will be integrated with the visualization of multivariate time series that necessitate sophisticated handling of non-serial time values so that multiple line plots can be properly synchronized.

Probably nowhere in *VisiCube* is the difference between its chart designs and those of pop-charts more dramatic than in the line plots used to visualize time series. Even in the univariate case, where only a single measure is plotted, the information content of the chart is very high. To talk about just one aspect of this content, each time point is provided with an extended grid line and the small spacing between time points (8 pixels) results in a dense gridwork across the chart. The gridwork provides an instant connection between a data point and the scale, but it also serves to carry a great deal of additional information. In the present release, it indicates **missing data values** (these are not related to missing time values) by the use of **blue** lines with no data dot and with interrupted point-to-point connecting lines. In the next release, when more dynamic handling of duplicate time values will be implemented, the grid lines will be used for carrying the additional data dots for multiple-use time points. When the categorization of datasets (such as used in scatterplots) is extended into the time-series facility, it will be the gridwork that will carry most of the additional information load. This gridwork usage is only the "tip of the iceberg" of what the line plot design will evolve into during the coming months, especially after multivariate time series are implemented.

Time series can be very long. Given enough space and processing time, a time series could contain about 2 billion points, but such a long time series would not be practical. The very extensive subset and extraction facilities of *VisiCube* dictate the use of considerably shorter time series. Probably most time series that you might use will consist of less than 10,000 points. Keep in mind that time series always represent a single **station**. This greatly cuts down on the size of any particular time series. The visualization SVs make it very easy to select different stations, as is explained below. The sample climate data cube, which contains monthly rainfall data for three countries in Southern Africa over a 27-year period, has 5,152 time points divided into 113 separate time series for 113 different weather stations. Additionally, no matter how long a single time series is, the maximum number of points in a single time series **plot** is 1,000. Longer time series are broken into plot **sections** of 50 to 1,000 data points.

Univariate group

Univariate Time Series SynchroView

Under construction! We will provide a description of this SynchroView with the next release.

Multivariate group

Multi-Measure Time Series SynchroView

Under construction! We will provide a description of this SynchroView with the next release.

Multi-Station Time Series SynchroView

Under construction! We will provide a description of this SynchroView with the next release.

Multiway group

Multiway Plots SynchroView

This *SynchroView* is very easy to use but it is not always easy to understand what you are seeing. Before reading the following discussion, spend some time in *VisiCube* playing with the multiway plot visual using two of the sample cube files**:** `Colleges.VCB` and `Livestock.VCB`. The meaning of much of what you see, as you select combinations of options, will be obvious but some things less so. Also, you may be unsure of what a "median" is, an important concept in interpreting the multiway plot. When you feel ready, continue reading below.

The multiway plot consists of a multiple number of dot plots. A single dot plot is contained in a panel of the multiway plot. You select one or two categories of the cube file in order to create a multiway plot. The first category defines the levels of the multiway plot. For example, if you select the category *country* from the `Livestock.VCB` cube file, the levels are the 26 countries Albania, Austria, Belgium, etc. With this one category selected, you have a one-panel multiway plot with 26 levels corresponding to the 26 *values* of the category. We will give a couple of examples of one-panel multiway plots before considering the more interesting multiple-panel multiway plot.

To begin with, our single panel contains as many levels as there are values of the selected category. A black dot is positioned horizontally on each level, representing a data value falling in the range specified by the scale at the top of the panel. What is the meaning of this data value? In the case of the *country* panel from the `Livestock.VCB` cube file, there are 130 records in the cube file, 26 records for each of 5 livestock types. There is a single measure in this cube file - a count of animals. Each record records the number of animals of a specified type in a specified country. In our single panel, a data value (for one of the 26 countries) is the sum of the counts of the animals of the 5 different types - the total count for that country. More correctly, the data value is the *sum* of the 5 counts if the SUM option is selected in the *Categories* window. If the AVG option were selected, then the data value would be the *average* of the 5 counts. If the MIN or MAX option were selected, then the data value would be the *minimum* or *maximum* of the 5 counts. These four options represent the accumulation functions for the multiway plot. Now it turns out that the `Livestock.VCB` cube file is particularly simple in that there is precisely one record for every possible *country-livestock* type combination, 26 X 5 = 130 records in all. This is natural since the data is the result of a census of the number of animals of the different types in these countries.

To get another picture, lets generate a one-panel multiway plot from the `Colleges.VCB` cube file, selecting *state* as the category. This cube file has 20 different measures and 3 different categories. We will assume that the measure *Math SAT* is selected and also that the SUM option is selected in the 4-way toggle. There are 51 state *values* for the selected category, so that the panel has 51 *levels*. Now lets look at a particular level, that for the state *NY*. The black dot is at the extreme right of the range so that the data value for *NY* is 40,900 (40.9 X {E+3}). There are 1,302 records in this cube file and

101 of them represent colleges in the state *NY* (You can quickly see this from the *Cubes.View* *SynchroView*). This black dot summarizes all 101 of these records.

There is an important difference between these two examples. In the *Livestock* example, each of the accumulation functions makes sense. In the *Colleges* example, the SUM function doesn't seem to make any sense**:** what is the meaning of the *sum* of the SAT scores? The answer is that it is a valid result but of no use. If the measure **"# new stud. Enrolled"** were selected, then the resulting sums would be meaningful**:** 76,300 new students enrolled in all the colleges in the state of *NY*. Generally, the other accumulation functions always make sense but the SUM function only makes sense if the selected measure represents a count.

Selecting a second category (*livestock type* using the `Livestock.VCB` cube file) yields a 5-panel multiway plot (one panel for each of the 5 livestock types). The animal counts that were summed over all the livestock types in the one-panel plot are here broken down into separate counts for each livestock type. When there are two categories, the **Invert** button is enabled, which lets you select the two categories in the reverse order with one click. If you do this, a 26-panel multiway plot is generated, one panel for each country, each panel having 5 levels of livestock types. In the case of the `Colleges.VCB` cube file, there are 3 categories from which two categories can be selected in three different ways. Together with their inverses, this yields 6 different multiple-panel plots. This cube file has 20 different measures, so there are 120 different multiple-panel plots that can be generated. There are also two ways to order the panels and (simultaneously) three ways to order the levels, resulting in 120 X 6 = 720 different plots. Finally the panels can be expanded or not, so that there are 1,440 different multiway plots with which to examine the `Colleges.VCB` cube file data.

Returning to the `Livestock.VCB` cube file, with the 5-panel multiway plot, we notice a problem due to the use of identical scales for each panel (this is the default case, with Expand = NO). The *Horses* panel data shows almost no variation because the scale has been adjusted to provide for the much greater counts used in the *Poultry* panel. The intent is to allow for comparison between the panels, but this hides the variation in the horse counts for the different countries. Notice that the comparison *between* the panels works fine - it is immediately obvious that there are much greater numbers of poultry than horses. The standard device for handling this situation is to use a *logarithmic scale*. This is a scale that looks somewhat like this**: [10 100 1000 10000].** This "spreads" the data out so that all the variation can be seen. However, there is a price paid for this. It is not easy to read logarithmically scaled data because the brain must decode the distorted positions of the dots in order to truly see the data. Rather than using logarithmic scales, *VisiCube* provides a toggle that lets you *expand* the data in each panel so that it precisely fits the width of the panel. Thus each panel has its own scale. This allows you to see the true data variation in each panel without having to decode the dot positions. You can instantly switch back and forth between individual and common panel scales to compare either from panel to panel, or from level to level.

Using the `Livestock.VCB` cube file with only the *country* category selected (so that there is but one panel), when you select a different accumulation function this panel's data changes. Thus the data for AVG is different than the data for SUM, as would be expected. With both categories selected and 5 panels displayed, this does not happen**:** each of the four accumulation functions yields exactly the same data. What is happening? In the one panel case, you are summing or averaging over a set of 5 records, so that unless each livestock type had exactly the same count, the results would have to be different. In the 5-panel case, you are summing over only a single record in each level of each panel, so that the results would have to be the same. This is because the livestock data, coming from a census, consists of a single record for each *country-livestock type* combination. In the general

situation, this would not be the case. In the `Colleges.VCB` cube file for example, there are many records for each *state-Math SAT* combination, so that AVG and SUM yield different results for each panel.

There are two ways to order the panels in a multi-panel plot. One way (Order = NO) uses the same order for the panels as for the values of the second category in the cube file. A more meaningful way (Order = YES) orders the panels by their medians. The *median* of a set of data values is that specific data value such that as many other data values are below it as are above it (this is slightly approximate if there are an even number of data values). When you select this option, a pink line is drawn down the panel marking the median data value for that panel. This line cuts through the center of one of the dots. You will note that the number of dots to the left of this dot is the same (or almost) as the number of dots to the right of this dot. Ordering the panels by their medians gives the feeling of the panel data "moving to the right" as you work down the panels from left to right (and top to bottom if there is more than one row of panels). The *median* is a much better statistic for this purpose than the *mean* (or *average*) because the *mean* is thrown off by the presence of outliers (data values outside of the normal range of values). Note that, if Expand = NO (all panels use the same scale), then the pink lines marking the medians are displaced increasingly to the right as you work down the panels.

There are three ways to order the levels in a multiway plot. They can be sorted or not, and, if sorted, they can be sorted by their medians or by their values. If not sorted, the levels are ordered from top to bottom in the same way as the values of the first category are ordered in the cube file. If sorted by their medians, the *same* order of levels works for each panel. This may seem somewhat surprising at first (one might think that each panel, having different data at a particular level, would have a different median at that level). Here is what happens (assume that you are using the `Livestock.VCB` cube file with the 26 levels being *country* and the 5 panels being *livestock type*)**:** for each level, the median of the 5 data values (one in each panel) is computed. This results in 26 medians, and these are sorted in ascending order, top to bottom. This order works across all the panels because its computation used the data from all the panels. If sorted by their values, a different situation arises**:** the data of only a *single* panel can be sorted in this manner. What about the other panels? They are forced to follow the order dictated by this single panel and therefore may be somewhat scrambled. The particular panel that is used to control the sort order is the one selected in the *Panels* window of the *SynchroView*.

Chapter 6 An Example

Under construction!

In this chapter, we will provide a complete example with the next release of the product.

Appendix A: VisiCube Requirements

VisiCube can be used on any computer that meets all of the following requirements:

- Windows 98, Windows Me, Windows XP, Windows 2000 or any later version of the Windows operating system.
- 333 MHz, or faster, CPU
- 64 MB, or more, of RAM
- 10 MB, or more, of disk space for the installed product (Additional space is required for temporarily for the installation program.)
- SVGA, or better, monitor which is capable of 1024x768 resolution, or better, and 16-bit color scheme, or better

Appendix B: VisiCube Limitations

The following is a list of the known limitations of *VisiCube*:

- Zoned files can have no more than 100 defined zones.

Appendix C: Frequently Asked Questions

Under construction!

In this appendix, we will provide a list of FAQs with simple answers to each.

Appendix D: Version History

The following describes each release of *VisiCube* and the most important capabilities that were made available with that release.

Version 0.1: Released March 31, 2002

Highlights:

- This is the original release of *VisiCube*! It supports spreadsheet data from the computer's clipboard.

Version 0.2: Released April 13, 2002

Highlights:

- The *visuals categorization system* (which is integrated throughout the multivariate visuals *SynchroView*s) has been largely re-written to simplify its use.

Specifics:

- The *Layer Trivariate Data* *SynchroView* has been rewritten and expanded.
- The *Brushing By Selection* *SynchroView* has been rewritten.

Version 0.3: Released May 7, 2002

Highlights:

- Time series visuals are supported.

Specifics:

- The *Brushing By Intervals* *SynchroView* has been added. (This complements the *Brushing By Selection* *SynchroView*.)
- The *Select Dataset* *SynchroView* has been changed to allow for the creation of snapshots of a dataset and its current visual state.
- The *Define Dataset* *SynchroView* has been entirely rewritten to simplify its use and expanded in functionality.
- The *Univariate Time Series* *SynchroView* has been added to support plots of one measure.
- The *Time Series Dataset Records* *SynchroView* has been added to link line plot points and their corresponding dataset records.

Version 0.4: Released July 27, 2002

Highlights:

- The structure of raw data, and the methods for handling it, has been completely revised to significantly improve performance and reduce the size of cube files.

- Support has been added for source data in zoned files (including keys to the categories in such data).
- The fundamental manner in which time series are handled has been changed to support the new concepts of time categories and time stations.

Specifics:

- The *Zoned File Library SynchroView* has been added to manage zoned files.
- The *Key File Library SynchroView* has been added to supplement zoned files with labels for their categories whose values act as keys to the data.
- The *Zoned File Extraction Facility SynchroView* has been added to support creation of new zoned files as well as cubes based on zoned file data.
- The *View Cube SynchroView*
- *This SynchroView* is used to look at the data records in the active cube file. This *SynchroView* is different from the ***View*** *SynchroView* in ***Raw Data*** in that the only records shown are those that match the selected value of the active category.

Viewing records by category value allows records to be represented as a slice of the multi-dimensional cube file. For instance, make the `Livestock.VCB` cube file active, then return to this *SynchroView*. Selecting **country** in the **Categories** window and then its value **Denmark** in the **Values** window causes the **Records** window to show only the records that contain the **Denmark** value in the **country** category. Likewise, selecting the **livestock** category, then selecting its **Poultry** value causes only records to be displayed that contain the value **Poultry** in the **livestock** category. In this manner, a sense of the cube file and how its data is cross-sectioned by the categories and their values may be obtained.

The **Records** window displays one measure or category field at a time, which is the field selected in the **Selected Record** window. When the active cube file contains a key, the key will be displayed alongside this selected field by default. Viewing the `Colleges.VCB` cube file in this *SynchroView* does a good job of exemplifying this behavior. The reason records are displayed in this manner rather than displaying them all at once is so that more information may be viewed in a compact space.

- Change Names and Labels SynchroView has been changed to allow for the automatic creation of category value labels from key file tables.
- The *Categorize Measure SynchroView* has been changed to automatically categorize a measure and include data visualization.
- Integration of the Univariate Visual *SynchroView*s (including *Single Quantile / Box Plot SynchroView*, *Under construction! We will* provide a description of this SynchroView with the next release.
- Multiple Quantile Plots SynchroView, and *Box Plots SynchroView*) has been improved.
- The *Time Series Dataset Records SynchroView* has been rewritten to simplify its use.

Version 0.5: Released September 20, 2002

Highlights:

- Reformatted and expanded the manual.
- Added outlier support.
- Changed the raw data paradigm to use a single data bank for the current cube at all times and to make its usage completely transparent.
- Changed the interface to better distinguish its components and simplify its use.
- Changed the file creation *SynchroView*s to allow for direct naming of the newly created files and explicit specification of the location in which those files are stored.
- Reorganized the *Navigator* to better match the intended flow of work.
- Added **Cancel** support to each *SynchroView* in which a plot is generated or a computation is done to enable you to terminate a long-running process.
- Throughout manual and product, replaced paradigm of "dataset" with that of "cube" for clarity.

Specifics:

- Removed the *System Settings* *SynchroView* from the *Navigator* since the folders are now specified and remembered in the file creation *SynchroView*s.
- Expanded time series support to include multiple time stations and measures.
- Expanded the *Zoned File Library* *SynchroView* to include more information about each defined zone.

Glossary of Terms

Every effort is made in *VisiCube* to utilize consistent, concise, and meaningful semantics. Understanding the terms listed in this glossary enhances your ability to use *VisiCube* productively.

category

A data field whose values qualify their applicable data records. In terms of measurement, these are the fields which determine or describe the record and, therefore, the measurements made for the record. (See also "measure".)

cell

A point, in the multidimensional space of a cube, in which one, or more, facts are stored.

cube

A multidimensional structure, comprised of cells, into which cube data is organized by *VisiCube*. This is a logical construct, sometimes referred to as a hypercube in other literature, which defines the organization of the data. A cube provides a single view of the cube data in the applicable cube file.

cube data

The raw data that is stored in a cube file and is, in turn, utilized by all cubes and visuals defined in that cube file.

cube file

A file with a proprietary format in which *VisiCube* stores cube data and cubes based on that data. Such a file is created and altered only by *VisiCube* and is fully portable between installations of *VisiCube*.

cube library

The library through which all cubes, in a given cube file, are made available to you.

data

Information which can be analyzed with *VisiCube*. Fundamentally, a set of data values.

data field

A set of data values having the same meaning and type across data records. An individual field is either a category or a measure. (See also "data record".)

data file

A file in which data is stored. Such a file is created and altered by software products other than *VisiCube*.

data library

The library through which all data sources, known to *VisiCube*, are made available to you.

data record

An ordered set of data values pertaining to a single entity, object, specimen, sampling, or observation where each value belongs to a distinct and different data field. (See also "data field".)

data source

A source of data for *VisiCube*. This is a logical construct which identifies and defines the physical source for a single set of related data values. This physical source is typically a data file. The definition of the source includes instructions on how the data can be extracted from the source to create a data table for use by *VisiCube*.

data table

A logical representation of data in which each row is a distinct data record and each column is a distinct data field.

data value

A single value which is either quantitative or qualitative in nature and, generally, cannot be decomposed into smaller meaningful parts. A quantitative value is numeric and can be processed mathematically. A qualitative value is non-numeric.

datamology

Our own term for the in-depth study of information using scientifically and mathematically sound techniques.

datamologist

You.

delimited file

A text file in which each record is a data record comprised of the same data fields. Further, the data fields of each data record are delimited by a specific character. Typically, the delimiting character is a comma or a tab. (See also "zoned file".)

drag and drop

The Windows action of "grabbing" an object on the desktop (by pushing and holding the left button on the mouse) and dragging it (by moving the mouse) to a new location before dropping it in that new location (by releasing the left button on the mouse).

file

A physical storage object on your computer. By physical, we mean that the file is explicitly named and is stored in a particular location on your computer. It can be located, moved, copied, renamed, or deleted by file management software such as Windows Explorer. Such a file is typically created and altered by a particular type of software product (if the format of the file is non-proprietary) or a single software product (if the format of the file is proprietary).

key

A category whose data values are all unique with the expectation that any one of these values can be utilized as a consistent and stable reference to the data record containing the value.

keyed source

A data source which has one, or more, keys defined for it. (See also "non-keyed source".)

library

A *VisiCube* facility in which all logical constructs of a similar type are made available to you for manipulation and management.

measure

A data field whose values quantify their applicable data records. In terms of measurement, these are the fields which are the result of measurements made for the record. (See also "category".)

Navigator

The menu-like component of *VisiCube*, displayed in the lower left pane, which provides you the ability to access any *SynchroView*.

non-keyed source

A data source which has no keys defined for it. (See also "keyed source".)

product control button

One of the standard Windows control buttons located in the upper right corner of *VisiCube*.

product control icon

The cube icon located in the upper left corner of *VisiCube*.

SynchroView

A synchronized set of related windows in which changes in any one window are immediately reflected in all of the other windows. This synchronized set of windows is designed to provide you with a unique capability and is displayed in the *SynchroView* Display Area of *VisiCube*. Each *SynchroView* is accessible from the *Navigator*.

text file

A data file that is comprised of individual bytes, each of which represents a single text character. The characters are stored in the file in their ASCII (American Standard Code for Information Interchange) representation. Such a file is created and manipulated by a text editor such as Microsoft Notepad. In such a file, each carriage return character (represented by the hexadecimal byte '0D') marks the end of a record. The file, therefore, is segregated by these carriage returns into any number of distinct records.

visual

A graphical presentation of a cube.

window

A single component of a *SynchroView*. Each window provides you the ability to accomplish a basic task related to the activity of the *SynchroView*.

zoned file

A text file in which each record is a data record comprised of the same data fields. Further, the data fields of each data record are in exactly the same relative position (or zone). (See also "delimited file".)

Index of Terms and Concepts

Under construction!

Here, we will provide an index to this manual to assist you in locating information quickly.