

Datamology

A venture of The Janina Ratkowska Group

VisiCube

The Data Microscope

Release 0.6

Copyright Notice

Copyright ©2002 Datamology.
All rights reserved.

Trademarks

Datamology, VisiCube, and Data Microscope are trademarks of The Janina Ratkowska Group.

Adobe and Acrobat are trademarks of Adobe Systems Incorporated.

Microsoft, Windows, and Excel are trademarks of Microsoft Corporation.

Acknowledgements

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

Chip Hartney designed *VisiCube* and the website and wrote this manual.

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

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

Summary of Contents	iii
Table of Contents.....	v
Conventions Used in This Manual.....	xi
About Version 0.....	xiii
Chapter 1 Why VisiCube?.....	1
Chapter 2 Getting Started	9
Chapter 3 VisiCube Concepts.....	23
Chapter 4 VisiCube Usage	47
Chapter 5 VisiCube SynchroViews.....	51
Chapter 6 Examples.....	65
Appendix A: VisiCube Requirements	67
Appendix B: VisiCube Limitations	69
Appendix C: Frequently Asked Questions	71
Appendix D: Version History	73
Appendix E: Sample Cube Files.....	77
Glossary of Terms.....	79
Index of Terms and Concepts	87

Table of Contents

Summary of Contents	iii
Table of Contents	v
Conventions Used in This Manual.....	xi
About Version 0	xiii
Chapter 1 Why VisiCube?.....	1
Why Is It Needed?	1
Overview.....	1
The Need.....	1
The Opportunity.....	2
The Solution.....	3
What Makes It Different?	4
Non-Modal.....	4
Direct Navigation.....	5
Window Synchronization.....	6
Accurate Visualization.....	6
Proactivity.....	7
Chapter 2 Getting Started	9
Obtaining VisiCube	9
Download.....	9
Installing VisiCube	9
Starting the Installation Program	9
Completing the Installation.....	9
Uninstalling VisiCube.....	10
Preparing Yourself.....	11
Gather data	11
Read this manual.....	11
Using VisiCube.....	11
Opening VisiCube.....	11
Orienting yourself.....	13

Table of Contents

Interacting with VisiCube	15
Adjusting VisiCube	22
Closing VisiCube	22
Chapter 3 VisiCube Concepts	23
Data	23
Data Facts	23
Data Types	23
Set Theory	25
Sets of Members	25
Cardinality	25
Ordered Sets	25
Missing Members	26
Sets of Sets	26
Set Operations	26
Subsets	26
Rules	26
Subset Definition	27
Statistics	27
Mean	27
Median	27
Quantile	28
Interquartile Range	28
Adjacent Value	28
Taxonomy	29
Experimental Data	29
Study	29
Phenomena	30
Observations	30
Surveys	30
Indicators	30
Records	34
Data Files	35
Data Sources	35

Data Source Library	35
Cubes.....	36
Example	36
Hypercubes	37
Dimensions	37
Cells	38
A Better Example.....	39
Measures and Measurements	40
Plots.....	40
Plot Capacity.....	40
Quantile Plots.....	40
Box Plots.....	41
Time-Series Plots	42
Scatter Plots	43
Cube Files	44
Master Cube	44
Subcubes	46
Visuals.....	46
Extraction.....	46
Chapter 4 VisiCube Usage	47
General Approach	47
Define Your Data Sources	47
Capture Your Source Data	47
Prepare Your Data For Analysis.....	48
Create Perspectives Of Your Data	48
Analyze Your Data Visually.....	48
The Current Object	49
Jittering	49
Time-Series Visuals	50
Chapter 5 VisiCube SynchroViews.....	51
Organization (in the Navigator).....	51
Manage Data Sources SynchroView	51
Change Zoned Source SynchroView	52

Table of Contents

Manage Cube Files SynchroView.....	52
Add Cube File SynchroView	53
Define Measure SynchroView	54
Define Dimension SynchroView	54
Classify Dimension SynchroView	54
Describe Dimension SynchroView	55
Manage Cubes SynchroView.....	56
Add Cube SynchroView	57
Brush Records SynchroView	58
Separate Dimension SynchroView	58
Limit Measure SynchroView	59
View Univariate Records SynchroView	60
Define Box Array SynchroView	60
Define Quantile/Box Plot SynchroView	61
Define Quantile Array SynchroView	61
View Multivariate Records SynchroView	61
Define 2D Scatter Plot SynchroView	62
Define 3D Scatter Plot SynchroView	62
Define Scatter Matrix SynchroView	62
View Time-Series Records SynchroView	63
Define Multi-Measure Time-Series Plot SynchroView	63
Define Multi-Group Time-Series Plot SynchroView	64
Chapter 6 Examples.....	65
Appendix A: VisiCube Requirements.....	67
Appendix B: VisiCube Limitations.....	69
Limitations of VisiCube	69
Limitations of Version 0	69
Appendix C: Frequently Asked Questions.....	71
Appendix D: Version History	73
Version 0.1: Released March 31, 2002	73
Version 0.2: Released April 13, 2002	73
Version 0.3: Released May 7, 2002	73
Version 0.4: Released July 27, 2002.....	73

Version 0.5: Released September 20, 2002	74
Version 0.6: Released January 4, 2003	75
Appendix E: Sample Cube Files	77
Brain sizes and intelligence	77
Three species of iris	77
Increased incidence of malignant melanoma	77
Egyptian Skulls	77
American Colleges	78
California Weather Data	78
Nigeria Temperature Data	78
Glossary of Terms	79
Index of Terms and Concepts	87

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 and keys 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 an italicized blue font and are underlined. These are active links that can be clicked to access the named location in this manual.

The Elements of Graphing Data

Publication titles are displayed in a black font and are underlined.

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.

See [Appendix B \(Limitations\)](#) for a description of the specific limitations of this free version of the product.

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.

Why Is It Needed?

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, datamologist is our term for one who is truly serious about data analysis but who is not necessarily a statistician or computer expert.) 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.

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.

What Makes It 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 the software itself (i.e., *VisiCube* is a datamology application regardless of its interface), it does greatly determine the usability of that software. And usability is important.

In his book, *About Face: The Essentials of User Interface Design* (IDG Books Worldwide, 1995), Alan Cooper, the designer of Visual Basic, describes the difficulty everyday people have with software as follows:

“Saying that someone is ‘computer literate’ is really a euphemism meaning that he has been indoctrinated and trained in the irrational and counter-intuitive way that [computers] work, and once you have been properly subverted into thinking like a computer nerd, the obvious ridiculousness of the way [it] presents itself to the user doesn’t seem so foolish.”

We at Datamology have taken these words to heart, as well as many of Cooper’s suggestions. 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).

Direct 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, hierarchical list. You simply click on the *SynchroView* node that you would like to access without having to work down through many levels of modal lists.

Window 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.

Accurate Visualization

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 an authoritative reference for the design of our visuals. The section named “Pop Charts” should be read by anyone who wonders why *VisiCube* does not provide the typical pie charts and bar charts found in many software products. We strive to present data in only the most statistically and mathematically appropriate manners and, therefore, avoid the use of common chart types and techniques which tend to be used mindlessly elsewhere.

For example, you will never see two-dimensional data graphed in three dimensions. Not only is it physically impossible to accurately present three-dimensions on a two-dimensional medium (such as your monitor or paper from your printer), the third dimension has no relevance or meaning. And, if it has no meaning, then its presence can only confuse the data being presented. Further, we strive to construct even the most complex graphics as simply as possible to let you see the data and its patterns as clearly as possible.

Another difference between *VisiCube* and other graphing software is the absence of the standard tick-marked scales. The data area of a graph is generally fitted exactly to the range of the data values used in that graph with the minimum and maximum values are displayed at the boundaries. This is based on an idea presented by Edward Tufte in his book, The Visual Display of Quantitative Information (Graphics Press, 1983). In such graphs, the data area is often divided into a grid of equal-sized regions which allows the distribution of the data to be immediately evaluated. Dense tick-marked scales are often provided in other products to allow estimation of specific data values. But, this is not necessary with *VisiCube* because determination of precise values can be immediately accomplished by simply clicking on the data point of interest.

Another innovation, one which eliminates a potential cause of confusion, is the restriction of data values to statistically significant digits. Generally, scales use values with a maximum of four

significant digits and a standard “power-of-ten” notation which indicates the magnitude. This power-of-ten is always a multiple of 3 indicating thousands, millions, thousandths, or the like. This allows for more direct and accurate understanding of individual values without the confusion of insignificant digits.

Finally, unlike many other graphical software packages, VisiCube explicitly handles duplicate observations of a single measurement. When such occurs, each duplicate is reported and a visualization technique is utilized to make the duplication apparent to you.

Proactivity

Although interactivity was once thought to be great, it must be recognized that the value of interactivity is limited and excessive interactivity is counterproductive. The result is software in which everything is controlled by the user. What is needed, instead, is proactive software which takes care of the common and mundane tasks of computing automatically. We have gone to great lengths to make *VisiCube* proactive.

Automatic Save

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

VisiCube also remembers what you were doing when you leave. Upon restarting *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.datamology.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.

If you run the installation program for a release of the product which you have not previously installed, the product is simply installed on your system. However, if you run the installation program for a different build of a release of the product which you have already installed on your system, the new build replaces the old one automatically.

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.

Destination Folder

The destination folder is the folder into which the *VisiCube* folder will be placed. The *VisiCube* folder will be named “VisiCube 0.6” and will contain the *VisiCube* program (which is not the same as the installation program) and its related files. 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.

Rebooting Your System

You do not need to reboot your system in order to use *VisiCube* after the installation completes.

However, if your system contained an older version of the Microsoft Installation (MSI) Engine than is utilized by the *VisiCube* installation program, it will have been automatically upgraded and you will be prompted to reboot your system once the installation is completed. You should do so to ensure that the MSI Engine is correctly installed on your system.

Uninstalling VisiCube

If, at any time, you wish to remove *VisiCube* from your system, you may do so by running the uninstall utility that is installed with the product. This utility can be invoked with either of the methods described below. Once invoked, follow the instructions presented by the utility.

You will typically wish to uninstall a release of *VisiCube* after you have installed a new release of the product and determined that it is satisfactory for your purposes. Note, however, that it is not necessary to uninstall old releases of the product. It is simply recommended to ensure that you are using the most recent version of *VisiCube*.

Via the Windows Control Panel

- Click the **Windows Start** button
- Click the **Settings** entry
- Click the **Control Panel** entry
- Click the **Add/Remove Programs** entry
- Click the **VisiCube** program entry
- Click the **Remove** button

Via the VisiCube Program Folder

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

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 \(VisiCube SynchroViews\)](#) as needed to clarify how the *SynchroViews* 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 methods described below can be used to open *VisiCube*.

When *VisiCube* starts, you will be presented with the *Version* window in which you are informed that you are using Version 0 of the product. You may close the window explicitly or simply begin working in *VisiCube*. As soon as you take any action outside of that window, it is closed automatically for you.

Via Windows Desktop

- Double-click the *VisiCube* icon

Via Windows Start Menu

- Click the Windows **Start** button
- Click the **Programs** entry
- Click the **Datamology** folder 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.

Control Bar

Title Bar

The screenshot displays the Datamology's VisiCube software interface. The window title is "Datamology's VisiCube View Multivariate Records". The main area is divided into several sections:

- Control Bar:** Located at the top, it includes a "NAV" button, "Help", "Dock", "Version", and "Cancel" buttons.
- Title Bar:** The top-most bar of the application window.
- Navigator:** A tree view on the left side showing the hierarchy of data sources, cube files, measures, dimensions, cubes, records, dimensions, measures, and visuals. The "Visuals" section is expanded to show "Univariate" and "Multivariate" options.
- Selected Record:** A table showing details for "Record 24":

ID	Name	Value
D1	Variety	Setosa
M1	SepalLength	5.1
M2	SepalWidth	3.3
- Cube Records:** A table showing the records used in the plot:

Record	Variety
24	Setosa
25	Setosa
26	Setosa
- Scatter Plot:** A plot titled "SepalLength X SepalWidth" showing the relationship between SepalLength (X-axis, ranging from 4.30 to 7.90) and SepalWidth (Y-axis, ranging from 2.00 to 4.40). The plot displays a scatter of data points, with a small box highlighting a specific point at approximately (5.1, 3.3).

Navigator

SynchroView Display Area

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 product title: “Datamology’s *VisiCube*”.
- In the middle is the name of the current *SynchroView*. (*SynchroViews* 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 component:

- The name of the object (data source or cube file) currently being processed by *VisiCube*. (Data sources and 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.

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 *SynchroViews* 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 **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 *SynchroViews* 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* contains a hierarchical tree, with collapsible nodes, in which the *SynchroViews* of *VisiCube* are organized into logical clusters. The *SynchroViews* are the end nodes of each branch of that tree. The clusters into which the *SynchroViews* (and other clusters) 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 cluster node to access the *SynchroView* in that cluster 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 distinct 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.

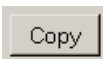
Elements of the VisiCube Interface

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 following is a button which provides “copy” functionality:



The availability of a button is determined by context. An available button can be pushed via left-click of the mouse when the mouse is over the button. When the button is released, the applicable action is taken. The availability of a button is indicated by the following:

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

Text Boxes

Text boxes are complex elements in which you enter freeform text.

There are two components of every text box:

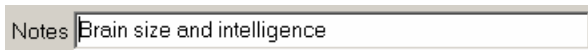
- On the left is a label which describes the text to be entered. This label uses black text on the background color (typically light grey) of the component in which the text box occurs.
- On the right is the field in which the text is to be entered. The text is black and the background color describes the state of the entered text as follows:
 - A white background indicates the text that is displayed is the same as that which has been saved by *VisiCube*.
 - A yellow background indicates the text that is displayed has yet to be saved by *VisiCube*. The color is utilized to warn you about this unsaved change.

The use of these text boxes is dependent on whether you are in a modal dialog or not. When in a modal dialog, simply enter the desired text and it will be utilized when you complete the dialog. Otherwise, when interacting with *VisiCube* through a standard *SynchroView* (which is not modal), there is a danger that you will type the desired text but never tell *VisiCube* to utilize it. The saved value is not updated on a keystroke-by-keystroke basis, so you must inform *VisiCube* when you are done. This is facilitated through the following mechanisms.

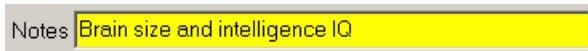
Text boxes come in two flavors. The first accepts only a single string of text without embedded carriage returns. This is a single-line text box. The other accepts multiple lines of text separated by carriage returns. This is a multi-line text box. The difference is whether the **Enter** key is recognized as part of the text input. Multi-line text boxes are accompanied by a companion **Apply** button which is to be clicked when the typed text is complete. Since single-line text boxes do not accept the **Enter** key, it is used as a shortcut to apply the typed text. I.e., when using a single-line text box, you type the desired text and hit the **Enter** key.

To make you aware that there is text which has yet to be utilized, as soon as you start entering text in non-modal text boxes, the background changes from white to yellow as a warning to you. Upon hitting the **Enter** key or the **Apply** button (whichever is applicable), the typed text is utilized and the background is changed back to white assuming the entry is valid. If it is not, an appropriate error message is displayed in the window.

Consider a “Notes” field that contains the value “Brain size and intelligence”. When the value is first displayed in a text box, it would appear as:



After appending “IQ” to the end of the value, it would appear as:



After hitting the **Enter** key, the background would again return to white.

The **Tab** key can be used to move the cursor between text boxes in a single window. But note that use of the **Tab** key does not cause the entered text to be utilized by *VisiCube*. Instead, it acts just like use of a mouse click to move to another field.

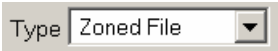
Dropdowns

Dropdowns are complex elements which provide you a mechanism for state selection. In a dropdown, one (and only one) state is always selected.

There are three components of every dropdown:

- On the left is a label which describes the state for which a selection is to be made. This label uses black text on the background color (typically light grey) of the component in which the dropdown occurs.
- In the center is a text field in which the selected state is displayed. This text field uses black text on a white background.
- On the right is a button, with a picture of an arrow pointing down, which can be pushed to expose a list from which the desired state can be selected.

The following is a dropdown in which the “Type” state is defined. It is currently set to “Zoned File”.



File Name Boxes

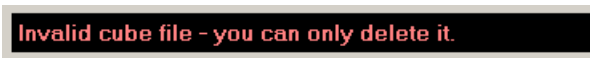
File name boxes are informational elements in which a file is identified. As seen below, they appear as flat rectangular elements with light yellow text on a black background and have two modes.



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:

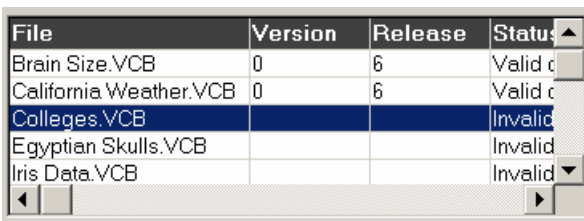
- Messages indicating success or normalcy use light blue text on a dark blue background.
- Messages indicating a problem or unexpected result use light red text on a black background.

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

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.

In the following example, which is a selection list element, horizontal and vertical scroll bars are provided (along the bottom and right edges, respectively):



File	Version	Release	Status
Brain Size.VCB	0	6	Valid c
California Weather.VCB	0	6	Valid c
Colleges.VCB			Invalid
Egyptian Skulls.VCB			Invalid
Iris Data.VCB			Invalid

Selection 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 or more objects from a set of many.

File	Version	Release	Status
Brain Size.VCB	0	6	Valid c
California Weather.VCB	0	6	Valid c
Colleges.VCB			Invalid
Egyptian Skulls.VCB			Invalid
Iris Data.VCB			Invalid

The top row of the table contains column headings which describe the objects in the list. Such headings use white text on a charcoal 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.

Single-select lists

A single-select list is a selection list in which only one entry may be selected. When an entry in a single-select list is left-clicked, it is selected and the previously selected entry (if not the same) is unselected.

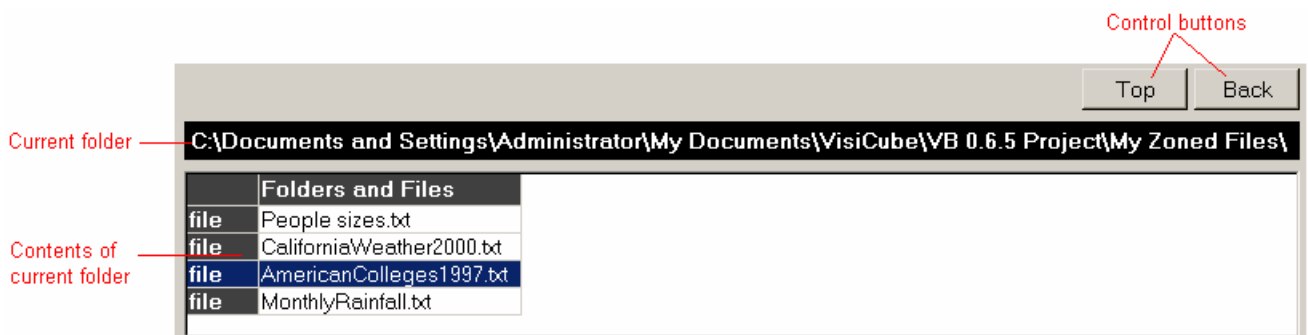
Multi-select lists

A multi-select list is a selection list in which multiple entries may be selected. When an entry in a multi-select list is left-clicked, it is selected and all previously selected entries (other than itself) are unselected. Multiple selections can be made in any of the following manners:

- Shift-left-click an entry to cause it and all entries between it and the prior selection (inclusive) to become selected.
- Ctrl-left-click an entry to add it to the current list of selections.
- Left-click the first entry in a series of desired entries and, while holding this button down, drag the mouse pointer up or down the list to the last entry in the series. Every value in the series will be selected.

File Locators

File locators are complex elements which provide you a mechanism to locate and select a file on your computer system.



The basic functionality of a file locator is such that you steer your way through the hierarchical file storage system for your computer until you locate the folder containing the desired file. The contents of the current folder are displayed in the list. Both folders and files are listed, along with an indication of their type. Once you have located the desired folder, you select the desired file from the list. The selected file is displayed in white text on a blue background.

The name of the current folder is displayed in the information box.

Hit the **Top** button to move back to the top of the hierarchy and refresh the list.

Hit the **Back** button to move back to the folder which contains the current folder and refresh the list.

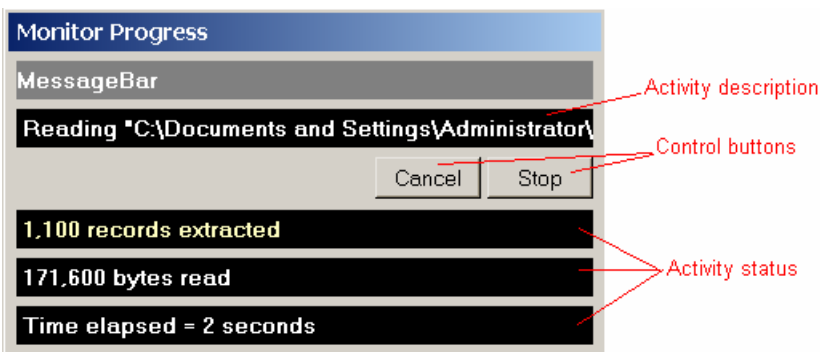
Click on a folder in the list to make it the current folder and refresh the list.

Click on a file in the list to select it (for whatever purpose this component is being used).

Progress Dialog

Any time *VisiCube* processes a request of yours, it disables all components of the interface until that action is completed. For example, when you click on a *SynchroView* node in the *Navigator*, it terminates the old *SynchroView* and initiates the new one. During this initialization, for example, it may be necessary to read data from disk, make computations, or generate a plot. As long as this processing is being done, you cannot interact with *VisiCube*. Once that process completes, you may continue. Of course, this processing time is usually extremely short and you are not likely to notice any delay. However, some processes are much more intensive and the wait time can become noticeable.

In such cases, *VisiCube* will present a modal dialog in which you can monitor the progress of that process. In addition, you are also given the opportunity to terminate that process prematurely. A **Cancel** button is provided which can be used to terminate the process and return to the state from which the process was initiated. For example, use of the **Cancel** button when trying to switch to a new *SynchroView* causes you to return to the prior *SynchroView*. If the process can be terminated in a meaningful state, you are also presented a **Stop** button which can be used to stop the process and continue with the partial results generated by that process. For example, this might occur when reading sample data from your system.



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.
- An active toggle uses white text on a black background.
- *SynchroView* nodes in the *Navigator*, when not highlighted, use blue text. Cluster 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).

Visual Colors

The basic colors used in all visuals are black and shades of gray on white backgrounds. Any other color always encodes something of relevance to the particular visual. The colors used have been chosen for contrast and compatibility. They cannot be changed.

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.

Data

The term “data” is used to describe things that are known. It is these known things which you analyze to try to understand other things which are not yet known. Data is, therefore, at the heart of your analysis.

Data Facts

But, the term “data” is plural. It is comprised of individual *facts* which cannot, generally, be decomposed further in any meaningful manner. For example, rainfall data in which it was recorded that 3.5 inches of rain fell in Eureka on Monday is comprised of three facts: 3.5 inches; Eureka; and Monday. It might be argued that “3.5 inches” can be divided into “3.5” and “inches”, but it should be recognized that neither of these individual components has meaning without the other. A fact, therefore, is the smallest datum with which *VisiCube* deals.

Data Types

Each fact is of a given type. Defining the type allows *VisiCube* to validate and translate the facts and make use of those translated values during analysis. *VisiCube* supports the following types of data.

Number

Numbers are those facts which are numeric in a form that can be recognized and processed by *VisiCube*. Such numbers must consist of the following characters:

- An optional lead sign (+ or -) is accepted. Positive is assumed if the sign is not present.
- An optional decimal point (a period) is accepted, but only one may be used. It is assumed to occur after the last digit if it is not present explicitly.
- Thousands separators (commas) may be present, but are ignored regardless of their position.
- Optional exponential notation (in the form of E12) is accepted at the end of the number.
- All other non-blank characters must be numeric digits.
- Blanks may occur before or after the non-blank characters.

Note that numbers are interval in nature (with regard to scale).

Temporal

Temporals are those facts which are whole numbers, in a form that can be recognized and processed by *VisiCube*, which represent a single value in a time sequence. Examples of temporal facts include:

- Centuries
- Years
- Months of a year
- Days of a month
- Hours of a day
- Minutes of an hour
- Seconds of a minute

Such temporals must consist of the following characters:

- All non-blank characters must be consecutive numeric digits.

Note that temporals are serial in nature (with regard to scale).

Date

Dates are those facts which are calendar dates in a form that can be recognized and processed by *VisiCube*. Such dates can be of almost any standard form which unambiguously identifies a single calendar date (where any character representation, if any, of the month is in English). Examples of supported date forms include:

- 2002 Dec 25
- 12/25/2002
- 25-12-2002
- 2002-12-25
- December 25, 2002

In addition to such standard formats, *VisiCube* also supports 8-digit dates in the form of “YYYYMMDD” (such as “20021225”) where:

- “YYYY” is the year
- “MM” is the month of the year
- “DD” is the day of the month

Note that dates are serial in nature (with regard to scale).

String

Strings are those facts which are neither dates, temporals, nor numbers. They are not interpreted or validated in any manner and may consist of any characters.

Note that strings are nominal in nature (with regard to scale).

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.

Cardinality

The *cardinality* of a set is the number of members in the set. For example, the cardinality of `{1, 2}` is 2 and of `[a, b, c]` is 3.

Ordered Sets

In general, the members of a set have no particular order. Therefore, the order in which the members are listed makes no difference. For example, `{Joe, Mary}` and `{Mary, Joe}` identify the same set since they contain the same members.

A set in which the order of the members is used in the identification of the set is called an *ordered set*. Ordered sets are represented by displaying their members inside a pair of brackets (instead of braces) as in:

`[Joe, Mary]`

Since order matters, `[Joe, Mary]` and `[Mary, Joe]` are not the same sets.

Further, ordered sets may utilize missing members. By definition, an ordered set is one in which the position of the member in the set is meaningful. Therefore, it is to be expected that an ordered set have a specific number of members (or cardinality). For example, an ordered set that contains the first, middle, and last names of a person might be `[Adam, Scott, Brown]`. However, if the middle name is missing, the set might be `[Adam, , Brown]`. Note that the position of the members is unchanged. The difference is that the second member is missing. It is important to understand that the cardinality of both is 3. The missing member is said to exist. It's just that the value is missing.

Missing Members

An ordered set may have missing members.

Sets of Sets

As noted above, the member of a set can be anything at all. Therefore, a member can itself be a set. An example of a set containing another set is:

```
{ Erin, {Joe, Mary}, Ernie }
```

where the second member of the set is a set.

If all of the members of a set are themselves sets, then the set is said to be a set of sets as in the following:

```
{ {Joe, Mary}, {Erin, Ernie} }.
```

Set Operations

Set operations are those operations which, when applied to one or more sets, produce a new set. *VisiCube* makes great use of two binary operations known as *union* and *intersection*.

The union of two sets produces a set consisting of those members that are contained in either of the original sets. Unions are indicated by the connective “OR”. For example:

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

The intersection of two sets produces a set consisting of those members that are contained in both of the original sets. Intersections are indicated by the connective “AND”. For example:

```
{Joe, Mary} AND {Mary, Helen} → {Mary}
```

Subsets

A set is a subset of another set if every member of the first set is also a member of second set. For example:

```
{Joe, Mary} is a subset of {Joe, Mary, Helen}  
{Joe, Mary} is not a subset of {Joe, Fred, Helen}  
{Joe, Mary} is a subset of {Joe, Mary }
```

It should be understood from the above rule that, by definition, any set is a subset of itself.

Rules

Another important way of representing a set is to describe it 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.

Subset Definition

Most of the sets utilized in *VisiCube* are sets of sets. And it is often the case that a subset of such a set of sets must be constructed. However, in the cases where the subset is quite large, instead of explicitly naming its members, it is desirable to define the desired subset through the application of rules and binary set operations.

It is a critically important concept of set theory that any desired subset can be constructed through the application of these rules (each of which produces a subset) and the binary set operations of union and intersection. *VisiCube* uses such definitions to construct subsets.

For example, consider a set of people whose members are each sets that contain two members, Name and Age, as follows:

```
{ {Joe, 33}, {Mary, 31}, {Sue, 25}, {Joe, 28} }
```

The subset consisting of:

- Those people who are named Joe and are over the age of 30; as well as
- Those people who are named Sue and are under the age of 30

would be:

```
{ {Joe, 33}, {Sue, 25} }
```

To construct the subset, we could use the following definition:

```
{ { {Name=Joe} AND {Age>30} } OR { {Name=Sue} AND {Age<30} } }
```

It can be seen that the definition noted above is flexible in that it is independent of the content of the original people set. It is also apparent that, when the size of the people subset gets quite large, the construction provides a much more concise method of specifying the members of the subset.

Statistics

Though not primarily a statistical application, *VisiCube* does apply many common statistics and statistical methods to facts. Understanding these statistics helps you understand the visuals and analyze your data. In the following discussion, consider the following set of facts:

```
{ 1, 2, 4, 5, 7, 7, 7, 9 }
```

Mean

The *mean* of a set of facts is their average. This is computed by adding the facts and dividing that sum by the number of facts in the set. The mean of the sample set is 5.25 as can be seen from the following computation:

$$(1 + 2 + 4 + 5 + 7 + 7 + 7 + 9) / 8 = 42 / 8 = 5.25$$

Median

The *median* of a set of facts is the fact that falls in the middle, or center, of the facts when they are listed in order. I.e., there are exactly as many facts in the set which are on one side of the median as

there are on the other. If the set includes an even number of facts, then the median is computed by finding the mean of the two facts closest to the middle. In our example, which has eight facts, 5 and 7 are the facts closest to the middle of the list. Their mean is 6. Therefore, the median of the set is 6.

Quantile

Given a percentage, n (between 0 and 100), the n *quantile* of a set of facts is the number at which approximately n percent of the facts in the set are less than, or equal, to the number. Note that approximation must be used since there may not be a fact that exactly meets this criterion. Conversely, there may be multiple facts that meet this criterion.

A *quartile* is a quantile based on quarters of the set. The lower quartile is the 25% quantile, the middle quartile is the 50% quantile, and the upper quartile is the 75% quantile. Stated another way, the lower quartile is the number below which $\frac{1}{4}$ of the facts occur, the middle quartile is the number below which $\frac{1}{2}$ of the facts occur, and the upper quartile is the number below which $\frac{3}{4}$ of the facts occur. Further, it can be seen that the median is the middle quartile because $\frac{1}{2}$ of the facts fall below it.

Our sample data has the following quartiles:

- Lower quartile = 3.00
- Middle quartile = 6.00
- Upper quartile = 7.00

By using consistent quantiles (namely those noted above) for various distributions, you can effectively compare those distributions directly in a statistical manner. *VisiCube* uses these quantiles to facilitate that analysis, specifically by including them in various visualizations. The method employed by *VisiCube* to compute these quantiles is complex and will not be explained in this manual. It is only important to you that the same computation is used consistently.

Interquartile Range

The *interquartile range* of a set of facts is the difference between the upper quartile and the lower quartile values for the set. This is a quantification of the spread of the data in the set. A small interquartile range indicates tightly packed data and vice versa.

In our sample data, the interquartile range is 4 as follows:

$$\text{UpperQuartile} - \text{LowerQuartile} = 7.00 - 3.00 = 4.00$$

Adjacent Value

Adjacent values are those facts in a set which are determined to be close to the median. Specifically, the upper adjacent value is the largest fact in the set which is less than, or equal to, the upper quartile plus 1.5 times the interquartile range. Similarly, the lower adjacent value is the smallest fact in the set which is greater than, or equal to, the lower quartile minus 1.5 times the interquartile range.

In our sample data, the upper adjacent value would be the largest fact which does not exceed:

$$7.00 + (1.5 \times 4.00) = 13.00$$

Therefore, the upper adjacent value would be 9.

Similarly, the lower adjacent value would be the smallest fact which is not less than:

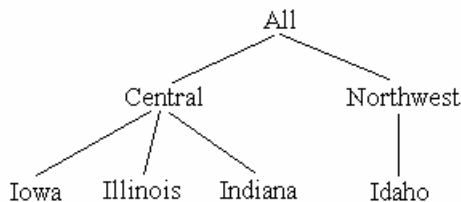
$$3.00 - (1.5 \times 4.00) = -3.00$$

Therefore, the lower adjacent value would be 1.

In this case, all of the facts of the sample set are adjacent values. However, this is not often the case. Typically, there are facts in the set which are not adjacent values. These are the extreme facts in the set. If 999 had been included in the set, it would exceed the upper adjacent value (even after re-computing the upper quartile and interquartile range based on its inclusion).

Taxonomy

Taxonomy is the science of classification which is, in turn, a system in which entities are systematically arranged into hierarchical groups based on factors that are common to each. For example, four states may be classified based on their locations within the U.S. as follows:



The individual entities appear at the bottom of the hierarchy. The set of all of the entities is represented by the top of the hierarchy (labeled here as “All”). Additional levels of the hierarchy are utilized to further classify the entities. In this example, a “region” level has been introduced and each state has been assigned to the appropriate region. It can be seen from this example that Iowa, Illinois, and Indiana all share the common trait of being located in the central part of the U.S.

Each distinct taxonomic unit at any level of the hierarchy is known as a *taxon* (plural *taxa*). In this example, the taxa of the “region” level of the hierarchy are “Central” and “Northwest”.

Experimental Data

Before using *VisiCube* effectively, you need to understand how it sees and manages your data. In this section, we describe the paradigm utilized by *VisiCube* so that the product, as well as this manual, will make sense to you. Specifically, our model is that of experimental research which has been historically accepted as the scientific method for discovering fact. The following is an explanation of the vocabulary used in this model.

Study

A *study* is a process in which you apply your mind in order to acquire knowledge through the careful and critical investigation and examination of a particular subject. The output of a study is often a thesis or monograph in which the results are compiled and published. *VisiCube* is intended to assist you in the examination of that subject as well as the publication of those results.

Phenomena

At the heart of scientific study is the practice of noting and recording pertinent information about specific *phenomena*. A phenomenon is an event, circumstance, specimen, sample, or experience that is apparent to the senses and that can be scientifically described or appraised. The key is that a phenomenon is observable and, therefore, recordable. I.e., information about the phenomenon can be obtained scientifically and stored for later analysis. It is the very recording of this information which makes it available to *VisiCube* for use in the process of analysis.

A given study, by definition, involves at least one type of phenomenon. For example, a meteorological study may be interested in two phenomena: monthly rainfall in the center of the continent and jet stream patterns over the same continent.

Observations

Of course, the observability of a phenomenon does not dictate that the phenomenon was actually observed. For example, the rainfall in Camp Point, Illinois, on May 23, 1973, was certainly observable. But it may be that nobody took note of that rainfall. The act of observing a phenomenon (and, therefore, recording information about that phenomenon) is an *observation*.

Surveys

Phenomena may also be observed in multiple ways. For example, rainfall could be observed for depth, breadth, intensity, or duration. It can also be observed by different observers or with different methodologies. The observers can even use different instruments to record their observations. To keep track of these differences, we use the concept of a *survey*. A survey is a set of related observations which are obtained and recorded in the same manner with regard to a single type of phenomenon. A study, therefore, may well be comprised of many different surveys.

It is also important to realize that, within a given survey, a single phenomenon may be recorded more than once. For example, if rain gauges are used to measure the amount of precipitation, it may be the case that multiple rain gauges were used to observe the same rainfall event (such as the rain that fell during May in Camp Point). It may even be the case that a single gauge was utilized, but that it was read multiple times. Each gauge, and each reading, would be the basis for a unique and distinct observation. Such redundant observations are often made in scientific studies to validate the information that is gathered and recorded.

Indicators

Each survey, being comprised of similar observations, uses a set of *indicators*. Each indicator is a single attribute, descriptor, or characteristic of the phenomenon whose value is determined during the observation. The indicators, taken together, define the phenomenon being observed...at least as far as the particular survey is concerned.

Consider a survey of the monthly rainfall in various American cities during 1973. It may be that the indicators of interest (those that will be recorded) are:

- Name of the city
- Name of the state

- Population of the city
- Name of the month
- Amount of precipitation

The following describes important differentiators between varieties of indicators. An understanding of these differences will greatly assist you in your use of *VisiCube* and your analysis of your data.

Dependence

One manner in which indicators differ from each other is that of dependence on each other. In the experimental model, each indicator can be thought of being either dependent or independent. *Independent indicators* are those which define the experimental condition and are, therefore, manipulated by the researcher. *Dependent indicators* are those which define the experimental results and are, therefore, measured by the researcher.

In the above example, assuming the researcher controls the location and month in which observations are to be made, the names of the city, state, and month would be independent indicators. The population and precipitation would then be dependent indicators.

Note that in the algebraic equivalent to this scientific model, these indicators are referred to as variables. And it is this very algebra which is utilized to evaluate the accuracy of the observations which were made and to predict the values of dependent indicators for phenomenon which were not observed.

Applicability

Another way in which indicators differ from each other is based on their applicability to the observation being recorded. A *direct indicator* is one which contains facts that directly describe the particular observation. An *indirect indicator* is one which contains facts that describe an entity referenced in the observation, as opposed to the observation itself.

In our rainfall example, the “precipitation” indicator directly describes the applicable observations and should, therefore, be considered a direct indicator. However, the “population” indicator describes the cities which, in turn, describe the observations. The “population” indicator should be considered indirect.

Chapter 3: VisiCube Concepts

Understanding the difference between direct and indirect indicators helps explain why there is no single correct way to record information. Our rainfall data could, just as correctly, be recorded in two distinct surveys. The first would be city population information:

Camp Point	Illinois	1,203
Hannibal	Missouri	18,825
Keokuk	Iowa	33,510
Quincy	Illinois	42,487

And the second would be the rainfall amount information:

Camp Point	Illinois	March	4.1"
Camp Point	Illinois	April	4.0"
Hannibal	Missouri	March	3.2"
Hannibal	Missouri	April	3.8"
Hannibal	Missouri	May	2.9"
Hannibal	Missouri	May	2.9"
Keokuk	Iowa	March	3.5"
Keokuk	Iowa	April	4.5"
Keokuk	Iowa	May	3.3"
Quincy	Illinois	March	4.1"
Quincy	Illinois	April	4.9"
Quincy	Illinois	May	3.7"

Another perfectly valid way in which this information might be recorded is one in which cities are identified with a code. In such a case, the name of the city, as well as the state in which it resides, become indirect indicators and the surveys could be rearranged in yet another manner. In this case, the city records could be:

CPT	Camp Point	Illinois	1,203
HNB	Hannibal	Missouri	18,825
KKK	Keokuk	Iowa	33,510
QCY	Quincy	Illinois	42,487

While the precipitation records would be:

CPT	March	4.1"
CPT	April	4.0"
HNB	March	3.2"
HNB	April	3.8"
HNB	May	2.9"
HNB	May	2.9"
KKK	March	3.5"
KKK	April	4.5"
KKK	May	3.3"
QCY	March	4.1"
QCY	April	4.9"
QCY	May	3.7"

Scale

Finally, the type of scale used for an indicator can be used to differentiate one from another. A scale is the measuring instrument utilized to obtain a fact. In the experimental model, the scale for any indicator must be exhaustive in that it can represent any potential fact. The scale must also be exclusive; the result of a measurement with that scale cannot be represented by more than one fact. E.g., if a scale is used to distinguish between “large” and “small”, use of that scale cannot generate a single fact which is both “large” and “small”.

An indicator necessarily uses one of five types of scales which further determine the values which can be obtained from the scale. These scales are summarized here and then described below:

	Nominal	Ordinal	Serial	Interval	Ratio
Values have inherent order		X	X	X	X
Values are regular			X	X	X
Values are continuous				X	X
Values have relative magnitude					X

A *nominal scale* is the simplest scale and implies nothing more than the basic requirements of a scale (those of exhaustiveness and exclusivity). I.e., it is one in which the facts are categorical in nature (as are all facts), but unordered. The state name indicator in our example uses a nominal scale. Note that the facts which are recorded in a nominal scale are discreet by nature. There is no implication about the existence of a potential fact between the others. For example, a nominal scale which uses “N” and “Y” as its potential facts in no way implies that any other letter is a valid fact.

An *ordinal scale* is like a nominal scale except that the facts also imply order. The month name indicator in our example is an example of an ordinal scale. To make the scale ordinal, a numeric representation of the fact is often utilized. For example, “4” might be used to represent “April” and “5” used to represent “May”. Like nominal scales, ordinal scales are also discreet.

A *serial scale* is like an ordinal scale except that the facts are regular in occurrence. By this we mean that the difference between potential facts has meaning. In the above example, the difference between “1” (poor) and “2” (rich) has no meaning. However, differences between facts of a serial scale have definite and consistent meaning. E.g., the difference between the years 1997 and 2001 is the same as the difference between 1861 and 1865 and that difference is 4 years. In fact, the difference implies the existence of the other possible values in the applicable range. For example, a serial scale which includes the years 1861 and 1865 necessarily includes the years 1862, 1863, and 1864 regardless of whether there are any facts recorded with those values. Like ordinal scales, serial scales are also discreet in that all potential values are known.

An *interval scale* is like a serial scale except that the potential values are no longer discreet. They are continuous and infinite. Between any two potential values, there are more (intermediate) potential values. Although accuracy of the scale may be limited, the facts being recorded by the instrument are not limited to a finite list. A clock is an interval scale in that it can identify a moment in time. 11:45 AM is such a moment, as is 11:46 AM. But so is a moment halfway between those...11:45:30 AM. Similarly, an extremely accurate clock might be able to record a value of 11:45:30.87456546 AM.

A *ratio scale* is the strictest scale. It has all the attributes of an interval scale except that there is also an absolute zero in the scale. The presence of the absolute zero allows for the determination of

magnitude. A good example of such is a Kelvin temperature scale (where absolute 0 is -273.15° Celsius). No value can be less than 0 and a value of 200 represents a temperature that is twice as hot as 100. In our example, both population and precipitation are ratio scales which, like interval scales, are continuous.

Records

As noted above, each observation is recordable. I.e., the information obtained during the observation can be recorded for later reference and analysis. This *record* is the permanent repository for that information. Further, since each observation in a survey uses the same indicators, each record in the survey has the same structure.

Consider our study of monthly rainfall in the tri-state area around Quincy, Illinois. If twelve observations were made during the spring, the resultant records might look like the following (assuming the above-mentioned indicators are utilized to describe each observation):

Camp Point	Illinois	1,203	March	4.1"
Camp Point	Illinois	1,203	April	4.0"
Hannibal	Missouri	18,825	March	3.2"
Hannibal	Missouri	18,825	April	3.8"
Hannibal	Missouri	18,825	May	2.9"
Hannibal	Missouri	18,825	May	2.9"
Keokuk	Iowa	33,510	March	3.5"
Keokuk	Iowa	33,510	April	4.5"
Keokuk	Iowa	33,510	May	3.3"
Quincy	Illinois	42,487	March	4.1"
Quincy	Illinois	42,487	April	4.9"
Quincy	Illinois	42,487	May	3.7"

Note that the expected May record for Camp Point is not present. This would be the case if the expected observation was not made. (It might be that the gauge malfunctioned, for example.) Note, also, that there are two records for Hannibal in May. This would be the case if a second gauge (and, therefore, observation) were utilized for that town and month. (Perhaps it was suspected that the first gauge was not accurate or that its location had been chosen poorly.)

The value which is recorded for a given indicator in a given record is a fact. In the above example, the fact recorded for the “city” indicator of the last record is “Quincy” and the “population” fact for the first record is “1,203”.

A record is, then, a set of facts. More importantly, it is an ordered set of facts. For example, in a record containing first and last names, [Franklin, Thomas] and [Thomas, Franklin] are different ordered sets and, therefore, different records.

A fact need not be present in a record. It may not be applicable to the particular record, may have been missed during the observation, or may have been missed during the recording of the observation. In any case, a record (which is an ordered set) always has the same number of members. However, any of those members can be missing. Looking at this in the opposite direction, a missing value is a valid fact and is considered a member of the record. For example, the records [Erin, 31, Novato] and [Chip, , Ukiah] have the same cardinality (of 3) and are compatible. The difference is that the value of the second indicator is recorded in the first record and missing in the second.

Data Files

Data files are physical files which reside on permanent storage devices (such as your computer's hard disk) and contain the data (in its original form) that you wish to analyze. Such a file is created and altered by software products other than *VisiCube*. Examples include text files, spreadsheets, and databases. They vary quite a lot in structure but are similar in that they contain your experimental data. It is from these files that *VisiCube* will obtain your data and make it available for analysis.

Data Sources

To facilitate the management of data files, and to make them compatible with each other, *VisiCube* uses the concept of a *data source*. A data source is a logical construct which provides a standardized and uniform view of a single survey. I.e., it provides access to the records and indicators of a given survey in a manner which is independent of the nature of the physical file containing the survey data. Once defined, a data source can be used to capture the survey data and is completely reusable.

The definition of a data source includes everything necessary to enable *VisiCube* to obtain the survey data from the physical file and present it to you as a two-dimensional table in which:

- Each row represents a single record.
- Each column represents a single indicator.

A data file can be used as the basis for any number of data sources. For example, a text file containing the following six lines:

```
Here is some 2002 country info...
  United States  3,678,896 square miles
  Canada        3,831,033 square miles
Here is some 1975 metro info...
  New York City 16,678,818 people
  Mexico City   11,943,150 people
```

Could be used as the basis for a "Country" data source with two records:

```
United States  3,678,896
Canada         3,831,033
```

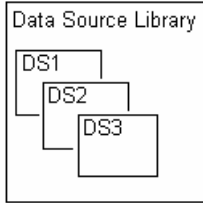
And another "Metro" data source with two records:

```
New York City  16,678,818
Mexico City    11,943,150
```

Various analyses could then be done on this source data without regard to the fact that it was originally embedded in a text file containing various types of information and extraneous lines of non-data text.

Data Source Library

VisiCube tracks all data sources in a library. You may define as many data sources as you wish. The definitions are stored in the library and this library is managed from the Manage Data Sources *SynchroView*.



Cubes

At the heart of *VisiCube* is the cube which is the foundation upon which all visuals are based. The name of the product, *VisiCube*, is meant to indicate this visualization of cube data. Cubes are, in fact, the logical structure into which your data is going to be organized before it is visualized.

Example

It will be useful to keep a simple example in mind as we discuss cubes. Despite the terminology, cubes are actually quite intuitive, especially if you are familiar with spreadsheets (which are, more or less, two-dimensional cubes). Our example is one in which the number of deaths in a city are recorded by year based on sex and height. In tabular form, this sample data would be:

Year	Sex	Height	Number
2000	Male	Short	43
2000	Male	Tall	31
2000	Female	Short	26
2000	Female	Tall	12
2001	Male	Short	37
2001	Male	Tall	26
2001	Female	Short	25
2001	Female	Tall	14

A simple cube representation of this same data would be:

	Year		
	2000	2001	
Height	Short	43	37
	Tall	31	26
Sex	Female	43	37
	Male	26	26

Hypercubes

A hypercube, or simply a *cube*, is a logical model in which data values are organized into a multidimensional Cartesian space. René Descartes, for whom the Cartesian space is named, pioneered the field of analytical geometry with his publication of *The Discourse On Method* in 1637. This branch of mathematics is based on the application of algebra to geometry. In it, Descartes developed the concept that the location of a point in space is completely determined by its coordinates along the dimensions of that space.

The term cube comes from the observation that points in a three-dimensional space look like a cube. Similarly, a one-dimensional space can be represented as a line segment and a two-dimensional space as a square. “Hyper” is used to indicate that this model is not limited to three dimensions. It just becomes difficult to come up with a physical analogy in our three-dimensional world for a hypercube of four or more dimensions. Our example, above, is a three-dimensional cube.

Dimensions

A *dimension*, then, is a single axis through the space of a cube. In our example, the dimensions of the cube are Year, Sex, and Height.

The *domain* of a single dimension of a cube is the set of distinct coordinates of that dimension. Unlike the dimensions of a Cartesian space whose domains are continuous and infinite (being the set of all real numbers), the dimensions of a cube have domains which are discrete and finite. I.e., they contain a finite list of discrete values. In our example, the domain of the Sex dimension is {Male, Female}.

A *coordinate* is a single member of a given domain. In our example, “Male” is a coordinate of the Sex dimension because it is one of the members of the domain for that dimension. Similarly, “Tall” is a coordinate of the Height dimension.

In *VisiCube*, you are allowed to supplement the definition of each dimension beyond its domain. These extended definitions support various capabilities including the selection of records and the labeling of dimension coordinates. Such definitions are described in [Dimension Definitions](#).

Cells

The points in cube space are called *cells* since they can actually contain something unlike their Cartesian equivalents which have no size at all. Specifically, these cells contain information of interest to the user of the cube. In our example, the cells contain counts. By necessity, a cube must have at least one dimension else there would be no cells in which to store the desired information.

Just as a Cartesian space is comprised of the set of all points in that space, a cube is comprised of the set of all cells in that cube. And, as in Cartesian spaces, each cell is located along a given dimension by a single coordinate. Therefore, going back to Descartes, each cell is located in the cube space with one, and only one, ordered set of coordinates. Note that the set is ordered. I.e., the individual coordinates must be specified in a pre-defined order to be meaningful just as points in a Cartesian plane are identified by an ordered pair of coordinates. This becomes obvious in a cube in which two dimensions share a common scale. The *coordinates* of a cell, then, are the ordered set of individual coordinates for each dimension of the cube. In our example, the Height coordinate of the cell containing the value of 31 is “Tall”. The coordinates of that same cell are [Male, Tall, 2000].

Each cell of a cube is the one container for all factual information identified by the matching coordinates. In our example, the Male-Tall-2000 cell is the container of all facts for which Sex is “male”, the Height is “Tall”, and the Year is “2000”. It happens to be the case that we are only storing a single fact in each cell of the cube: the number of deaths in the city. But there is nothing to prevent us from also storing another fact in those cells such as the number of births.

The number of cells in a cube is the product of the cardinality of each of the domains for each of the cube's dimensions. In our example, which has three dimensions each with a cardinality of 2, the number of cells is 8. ($2 \times 2 \times 2 = 8$.) The number of cells in a cube whose two dimensions have cardinalities of 5 and 40 is 200. ($5 \times 40 = 200$.) The number of cells in a 10-dimensional cube whose dimensions each have a cardinality of 10 is 10,000,000,000. And this is not an especially large cube!

But, don't be concerned. Although the number of cells in a cube can be astronomically large, it is always finite. Since the number of dimensions in a cube is finite and the cardinality of each domain is finite, then the number of cells in the cube is also finite. Unlike Cartesian space, in which the number of points is infinite, a cube contains a fixed number of cells. There are always first and last cells along any given dimension. Further, remember that a cube is a logical object. Since it does not actually exist in a physical form, you won't exceed the physical limits of your computer simply by using such a cube. Besides, *VisiCube* manages these cells in a manner such that you don't need to worry about the quantity of these cells. Bear in mind, also, that most cells of a cube are not used! Consider a cube in which the dimensions are month (January through December) and day (1 through 31). Even if you collect data for every day of the year, there are still some cells, such as the ones for February 30 and April 31, that will not be used. As noted earlier, the cube includes a cell for every set of coordinates. That does not mean that the cell is valid or utilized. Generally, very few cells of a complex cube are actually utilized.

Finally, it is important to understand that, contrary to our over-simplified example, each cell is actually a set of sets. In the terms of relational algebra, a cell is a bag of tuples. Each tuple is a set and every tuple in the bag has the same number of elements (or cardinality). For our purposes, we think of the cell as simply being a set of sets. Namely, each cell contains one set for each desired indicator and each of those sets contains every fact for that indicator which is identified by the applicable coordinates. The following example should make this clearer.

A Better Example

Our example above is a very simple cube in that each cell contains a single fact. In *VisiCube*, if only one indicator (the count) were to be tracked in the cube, each cell would hold a single set. And, if there is only one record which maps to each cell, the sets would all contain a single value. In other words, in our previous example, *VisiCube* would utilize a cube in which each cell contained a set whose only member would be the value indicated. Instead of the value “31” in the Male-Tall-2000 cell, there would be a set of {31}. If the data included a second record for a given cell, then a second member would be included in the set in that cell. For example, if a second Male-Tall-2000 record occurred in the data with a count of 32 (perhaps due to a recount), the cell would contain {31, 32}. Finally, if the data included a second indicator which was to be captured in the cube cells...average age, for example...each cell would contain a second set.

What follows is a better example of the true nature of cubes as they are implemented in *VisiCube*. Consider the following death records:

Date	Sex	Age	Height	Weight
2000-03-26	Male	48	64	143
2000-09-12	Male	25	73	231
2000-09-12	Female	26	58	106
2000-12-24	Male	71	67	128
2001-02-20	Male	90	71	157
2001-05-14	Male	66	77	196
2001-10-10	Female	84	61	125
2001-10-28	Female	72	65	141

A common cube representation of this 2-dimensional data would be:

Male	Age: {48, 25, 71}	Age: {90, 66}
	Ht.: {64, 73, 67}	Ht.: {71, 77}
Female	Age: {26}	Age: {84, 72}
	Ht.: {58}	Ht.: {61, 65}
	Wt.: {143, 231, 128}	Wt.: {157, 196}
		Wt.: {125, 141}
	2000	2001

Note that the cardinality of each set in a given cell is the same. This is due to the fact that a member is added to every set in a cell for each record that maps to the cell. The cardinality of any one of the sets in a cell can be used to determine the number of records that mapped to that cell and, therefore, had the matching coordinates. I.e., the number of Female-2000 records is 1 as is apparent from the cardinality of any one of the sets in the Female-2000 cell.

Measures and Measurements

A *measure* is any indicator whose facts are used to populate the cells of a cube. A *measurement* is a single fact recorded for a measure. In the above example, Age, Height, and Weight are used as measures. Every measure in a cube causes the creation of a set of measurements in each cell of the cube. Note that each cell in our example contains three sets of measurements, one for each of the measures.

In *VisiCube*, a unit of measure can be recorded for each measure. This is displayed in most of the visuals for clarity.

Plots

A *plot* is a primitive graph. By primitive, we mean that it cannot be decomposed into smaller components which themselves have meaning independent of the plot. *VisiCube* utilizes such plots, often in combination, to create more powerful visualizations of data. The types of plots utilized by *VisiCube* are described below.

Plot Capacity

Each plot has a specific capacity with respect to measures. Namely, the capacity of a plot indicates the number of measures that can be graphed in that plot.

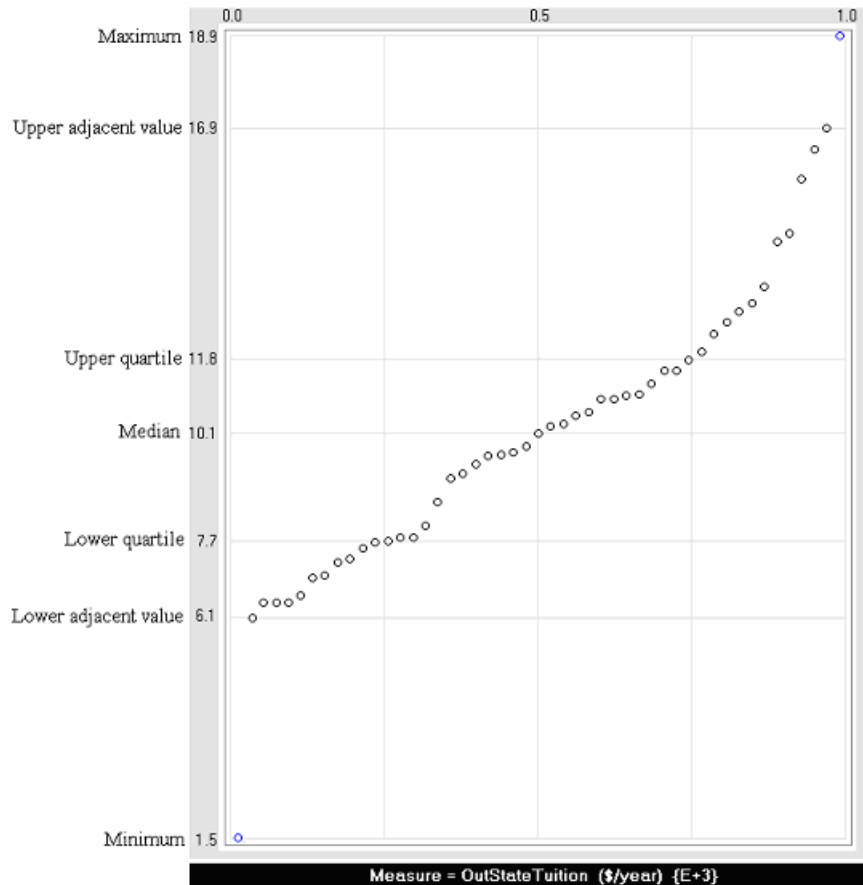
- Univariate plots are those in which a single measure is graphed.
- Bivariate plots are those in which two measures are graphed.
- Trivariate plots are those in which three measures are graphed.
- Multivariate plots are those in which more than a single measure is graphed (which, of course, includes all bivariate and trivariate plots).

Quantile Plots

A quantile plot is a univariate plot in which the distribution of a measure is presented with pertinent statistical values used as demarcations along the measure's scale. The values are presented in order from minimum to maximum along the other axis. This axis is divided into quantiles, typically quarters, and labeled with the fraction (between 0 and 1) which represents the applicable quantile. For example, the 0.5 quantile (which is the median) is the quantile for which half of the values on the minimum side of the quantile line. The result is a pattern that represents the distribution of those values.

The following example displays out-of-state tuition costs in 1997 for colleges in Illinois (using a scale marked in thousands of dollars). Among other things, it is apparent from the plot that:

- The median cost is \$10,100.
- The college whose cost is reported to be \$1,500 appears to be a major variation from the expected values. It could represent an error in the data.



VisiCube often generates a box plot for presentation along side a quantile plot so that the statistical values are made more visual and, therefore, easier to interpret and compare (against other measures).

Box Plots

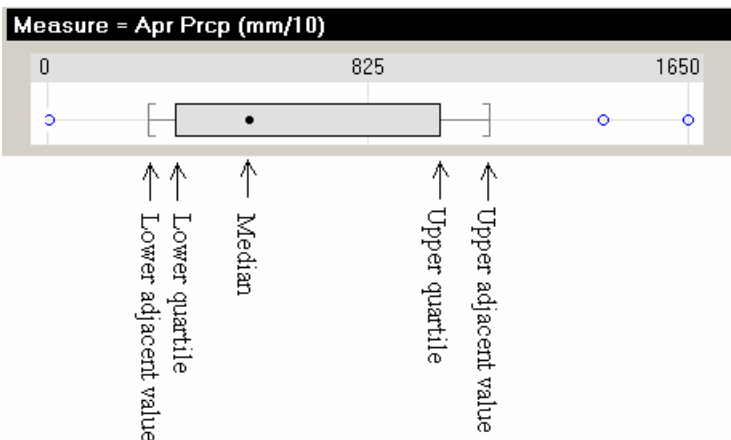
A box plot is a univariate plot in which a statistical summary of the distribution of a measure is presented. This type of visualization was introduced by John Tukey in his pioneering book, *Exploratory Data Analysis* (1977). In such a plot, the measure is plotted along the lone axis and the statistics are displayed in a graphical form along the axis. Specifically, in *VisiCube*, a box plot is displayed as follows:

- The median is shown as a filled (solid) dot.
- The interquartile range is shown as a shaded box that encompasses the median.
- The upper and lower quartiles are, therefore, shown by the ends of the interquartile range box.
- The upper and lower adjacent values are shown as brackets (opening toward the median).
- Each non-adjacent value is displayed individually as an unfilled (open) dot.

Note that the adjacent values, which usually include nearly all of the values for the measure, are not displayed individually. To see all of the adjacent values, you should utilize a quantile plot instead of this statistical summary of those values.

The following example displays precipitation data for the month of April in 1871 at all N.O.A.A. stations. Among other things, it is apparent from the plot that:

- Three values, those which are not adjacent, appear to be outside of the expected range. (They are marked in blue and may need to be investigated more specifically.)
- Many more than half of the values fall below the midpoint of the scale (825). (This is obvious from the fact that the median is well to the left of that midpoint.)



Box plots are especially useful when compared to each other. Such a comparison makes it easy for you to determine, at a glance, how the distributions of the two measures compare with each other.

Time-Series Plots

A time-series plot is a univariate plot in which the variation of a measure across time is presented. The measure is plotted along one axis and the time dimension is plotted across the other. The minimum and maximum instances of the time dimension are used to determine the extent of the time dimension. Then every potential instance of that time dimension is included (whether or not the instance actually occurs in your data).

The data points for each adjacent time instance are connected to form a line...often very jagged in nature...to enable you to see the trend of the data across the progression of time.

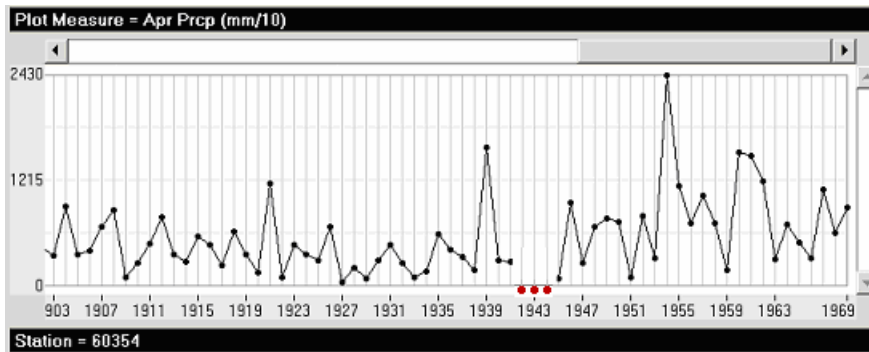
If a time instance has no data point, the line is broken at this point...making it clear that measurements were not made at the noted time. If there are no data records at all for the time instance, a red dot is displayed along the time axis at that instance.

If there is a single data point for a given time instance, the data point is marked with a black dot. If, however, there are multiple data points for the time instance, each of the points is marked with a blue dot. Note that in the case where you see only one blue dot, the situation is one in which multiple data points had the exact same value.

The minimum and maximum measures, along with their midpoint, are displayed along the measure axis.

The following example displays precipitation data for the month of April in each year at N.O.A.A. station #60354. It is apparent from the plot that:

- Measurements were not recorded in the years 1942, 1943, and 1944 (possibly due to WWII).
- The years between 1927 and 1938 were among the driest Aprils ever recorded (possibly a cause of the Great Depression).
- The wettest April recorded was in 1954.



Scatter Plots

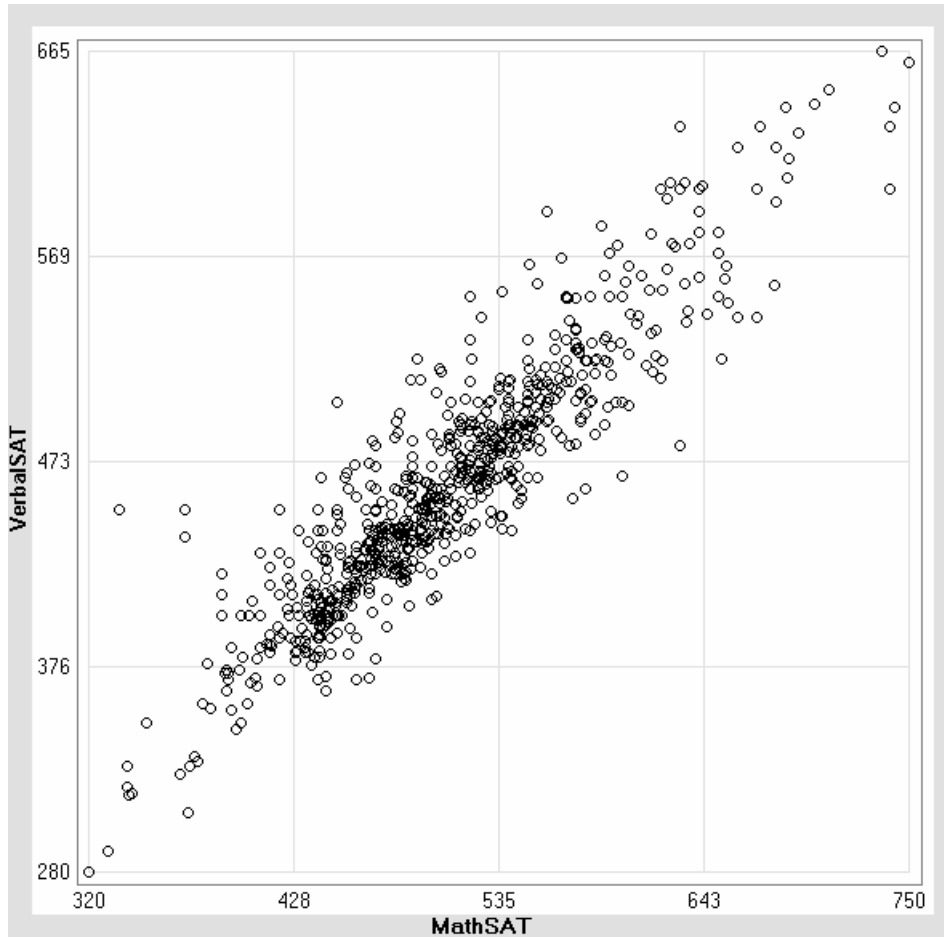
A scatter plot is a multivariate plot in which a comparison of measures is presented, one measure along each axis. A two-dimensional scatter plot, therefore, is bivariate in nature and a three-dimensional scatter plot is trivariate in nature. [Note that a scatter plot of more than three dimensions cannot be represented directly in the physical world since that world is three-dimensional.]

A scatter plot provides a useful exploratory method for understanding the correlation between the two measures (which may lead to conclusions about the dependence of one measure on the other). A strong correlation is represented by a plot in which the values for one measure steadily increase (or decrease) as the values of the other measure change.

Each measure is plotted using a scale whose extremes are the minimum and maximum values for the measure and is divided into equal sections (usually quarters).

The following example compares verbal and math SAT scores for the incoming freshman classes at all colleges in the U.S. in 1997. It is apparent from the plot that:

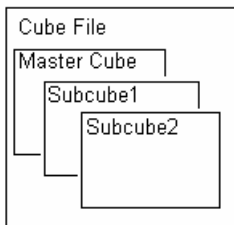
- There is a strong and direct correlation between the two measures. (As one score increases, so does the other.)
- Math scores are generally higher than verbal scores.



Cube Files

Cube files, like data files, are physical files which reside on permanent storage devices (such as your computer's hard disk). Unlike data files, cube files can be created and used only by *VisiCube* due to their proprietary format. A cube file is completely self-defining and, therefore, fully portable between installations of *VisiCube*.

Fundamentally, a cube file contains a set of related cubes defined against a single set of data.



Master Cube

A cube file is the permanent container into which you place one set of related data that you wish to analyze. This data is stored in the form of a cube, which we refer to as the *master cube*. A master cube is, therefore, present in every cube file.

Dimensions

The definition of a dimension of the master cube can be extended to include more than its domain. Doing so provides you the means to create cubes and generate visuals more precisely.

To understand such a definition, it is best to think of it as a table in which each instance is defined by a row. The columns of the table represent each *property* of the dimension. Each property is an attribute in which a value is recorded that describes the particular instance of the dimension.

Consider a “state” dimension with 3 properties (Key, Capitol, and Region) and 4 instances:

<u>Key</u>	<u>Region</u>	<u>Capitol</u>
Iowa	Central	Des Moines
Idaho	Northwest	Boise
Illinois	Central	Springfield
Indiana	Central	Indianapolis

Before looking at properties in more detail, you need to understand two other concepts.

First, the domain for a dimension is that set of unique values which identify the instances of that dimension. These values are the coordinates of that dimension. We often refer to these as the key values, or simply the keys, for that dimension because each value uniquely identifies a single instance of the dimension’s domain. In our example, the name of the state has been used as the key to the dimension. Every dimension has a key. In this example, the state names are utilized as the keys. I.e., “Iowa” identifies a single instance of the dimension and none other.

Second, the instances of a domain are classified hierarchically. In the simplest of dimensions, this classification hierarchy is comprised of only two levels. At the bottom are the individual instances. At the top is the dimension, itself, which we generally refer to as being “All” of the instances.

It should be apparent that each property of a dimension serves a purpose in the definition of that dimension. Specifically, a property serves one of the following purposes:

	Description	Classification	Identification
Property contains descriptive values	X	X	X
Property is part of classification hierarchy		X	X
Property contains key values			X

A *description property* is one that only contains descriptive values with regard to the instances of the dimension. It is the most basic property. The other types of properties are also description properties, but they have additional meaning and use. In our example, “Capitol” is a description property. It describes the instances of the dimension, but is not involved in the classification or identification of those instances.

A *classification property* is one that, in addition to being a description property, is used as part of the classification hierarchy for the dimension. In our example, “Region” is a classification property. It is used in classification, but not identification, of the instances of the dimension.

An *identification property* is one that, in addition to being a classification property, is used to uniquely identify the instances of the dimension. In other words, it is the property that contains the coordinates of the dimension and which forms the base of the classification. In our example, “Key” is the identification property.

In *VisiCube*, note that there is always one, and only one, identification property for a dimension and it is always named “Key”. The simplest, and most common dimension definitions are comprised of a single property, named “Key”.

Subcubes

You can define any number of additional cubes which contain subsets of the data in the master cube. Each such additional cube is referred to as a *subcube*. Each subcube dimension shares the same definition as that in the master cube. I.e., the properties of a particular dimension in a cube file are shared by every cube in that cube file (which includes that dimension).

Subcubes are useful for a few reasons. First, each cube provides a different perspective on the data in the cube file. It is often the case that you will want to use multiple perspectives. Second, each cube can utilize different data records. Finally, it may be the case that you wish to format a single type of visual in multiple manners. This can be accomplished by creating two subcubes and defining that type of visual differently for each.

Visuals

A *visual* is a graphical presentation of data which is often complex in nature. We use the term to emphasize the manner in which it is consumed (visually). While cubes are the foundation upon which analyses are developed in *VisiCube*, visuals are the building blocks placed upon that foundation. It is through one, or more, visuals that you obtain the perspectives necessary to analyze your data.

We think of *VisiCube* as a data microscope because, with it, you can examine any part of your data in extreme detail. The visuals can be thought of as the slides upon which data is placed for use with that microscope. You study the slides and, through them, analyze your data.

In addition to the cubes, themselves, the cube file also contains visualization information for each cube (including both the master cube and every subcube). It is the inclusion of this visualization information which makes the portability of the cube file important. The cube file acts as a set of reports on the data in the file and it is these reports which you are likely to want to share with others. The inclusion of the actual data makes the reports stable. The data does not change over time. The data in the cube is a snapshot of the source data at the time the cube is created. Therefore, the visuals defined for each cube also remain stable.

Extraction

Extraction is a powerful *VisiCube* facility in which data in a proprietary cube file can be extracted and saved to a non-proprietary, plain-text data file. This can be used on the cube files that you obtain from other people (such as the sample cube files supplied with *VisiCube* which are described in [Appendix E – Sample Cube Files](#)) to expose the data for your personal manipulation and use in new cube files.

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 \(*VisiCube SynchroViews*\)](#) for details about the specific capabilities of *VisiCube*. Please see [Chapter 6 \(*Examples*\)](#) for detailed examples of this process.

General Approach

VisiCube usage consists of the following basic steps. First, you locate and define the sources of the data that you wish to analyze. Then, you capture that data into a proprietary cube file. That data is stored in the form a cube (which we refer to as the master cube) in the cube file. Next, you prepare the data in your cube file to facilitate analysis...though this is usually not necessary. If you wish to utilize more than one perspective of your data (i.e., the one supplied by the master cube), you can create additional perspectives of your data in subcubes. Finally, you analyze your data by manipulating visual representations of your data.

Define Your Data Sources

VisiCube must be able to access and interpret your data before you can proceed with analysis of that data. In defining your data sources, 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 can be used in the creation of cube files.

Capture Your Source 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 is the data in the form of a cube.

To create a cube file, you select the desired data source (or sources) and obtain the desired data from those sources. The captured data is stored in the cube file in the form of a cube (which is referred to as the master cube). You provide the name of the cube file and save it in permanent storage at a location of your choice (which is accessible to your computer).

To formulate the master cube, you specify the desired indicators and records from your source data. Each selected continuous indicator (one that uses an interval or ratio scale) is used as a measure in the

master cube. All of the selected indicators, including the continuous ones, are used as dimensions of the master cube.

Prepare Your Data For Analysis

Once the master cube has been created, the definition of its components (the dimensions and measures) can be further refined and expanded in a couple of ways...though doing so is completely optional.

With regard to measures, you can update their default definitions (such as the unit of measure utilized) and limit the values for each. In *VisiCube*, we allow you to identify outliers for each measure in the master cube...those outside of the specified limits. Then, regardless of the cube being visualized, the outliers are handled in a consistent manner.

The definitions of the dimensions of the master cube can also be updated to facilitate analysis. Specifically, additional properties can be added to the dimension's definition (in addition to the identification property named "Key" which is always present in the definition). First, description properties can be added from other data sources. For example, consider a "state" dimension whose identification property utilizes 2-character postal codes such as "AL" and "MN". The dimension might be further described with additional properties such as name, population and geographic size. These description properties are obtained from a data source which has as its key the same 2-character postal codes. Second, classification properties can be added which, taken together with the key, form a complete classification of the dimension. For example, the instances of the same "state" dimension could be classified into regions such as "North", "West", "East", and "South". Such classification properties can be added through an explicit classification process (either manual or automatic). All of these additional properties can then be utilized during analysis in the same manner as the key, itself.

Create Perspectives Of Your Data

The master cube provides you with a perspective of the data in the cube file. Analysis usually proceeds through that master cube. However, it can be desirable to create additional perspectives of that same data. This is accomplished through the creation of subcubes in the cube file. Each subcube inherits all of the attributes of the master cube (with regard to dimensions and measures), but provides you a slightly different perspective of the data in that master cube. It is useful to think of each subcube as a separate view of your data or some subset of that data.

A common usage of subcubes is to "collapse" the data in the master cube into a subcube of fewer dimensions. Consider a master cube whose measures are "Time" and "Place". It may be desirable to use a subcube which contains all of the same measures, but consists of only a single "Time" dimension. I.e., one in which the facts for a single time are stored together regardless of place. Another common usage is to restrict the data that is analyzed. For example, a subcube could be comprised of only those facts which pertain to China (place) and 2001 (time).

Analyze Your Data Visually

All cubes in a cube file, both subcubes and the master cube, can be analyzed independently and in exactly the same manner. You simply select the cube that you wish to analyze and proceed with that

analysis. That analysis is done through the use of visuals which you define and save (should you wish to).

VisiCube provides a variety of visualization facilities, each of which provides a different technique for examining your data and is completely dynamic. Each change to a visual is immediately reflected in the applicable *SynchroView*. At any time, a visual can be saved for later reuse or reference. Therefore, each cube has an attendant list of visuals which have been saved for it.

The Current Object

The design of *VisiCube* is one in which you manipulate an object with any of a number of related *SynchroViews* which you are free to move between. The *Navigator* is divided into two parts, one for each type of object that can be manipulated by a *SynchroView*: a data source or a cube file. The identity of the current object being manipulated is displayed in the top line of the control bar. When accessing a data source *SynchroView*, the name of the data source is displayed. When accessing a cube file *SynchroView*, the name of the cube file is displayed.

The current object is the one which is manipulated by any *SynchroView*. In general, you select the current object from a management *SynchroView* and then manipulate it in any related *SynchroView*. To change the current object, you return to the management *SynchroView* and select a new object.

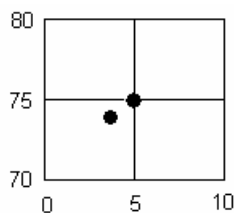
When closing *VisiCube*, the current object is recorded. When you next open *VisiCube*, that same object is automatically opened. In fact, you are also returned to the last *SynchroView* you were using.

Jittering

A problem that can occur in plotting, especially in multivariate plots, is that data values which have identical coordinates overlap each other in various plots. Consider a scatter plot of the following data which uses a black dot to represent data points:

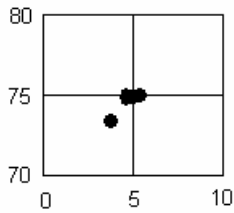
<u>Station</u>	<u>Temp.</u>	<u>Prec.</u>
47385	74	4
47386	75	5
47387	75	5
47388	75	5

A scatter plot of those four records would look like the following:



It appears (incorrectly) that there are only two records because of the duplication of values in the records. While this would be an appropriate presentation if you are interested in the distinctness of the values, it would not be appropriate if the frequency of particular values is important.

Jittering is a technique in which a small amount of uniform, but random, noise is added to the data before plotting. The effect is that the data points are plotted in slightly different positions to expose those which would normally be hidden:



VisiCube allows you to define the noise level for each measure used in each cube file and, through the various visuals, dictate when such jittering is to be utilized.

Time-Series Visuals

To create time-series visualizations, you must designate a *time dimension*. This is the dimension whose instances will be plotted along the time axis in a serial fashion. Those instances must be ordered and regular, but discreet (unlike numeric data which is continuous). In such data, unlike any other types of data, the unused values can be determined and displayed in the plots. For example, temporal data whose domain is:

{1997, 1999, 2002}

can be plotted in a series whose domain is:

{1997, 1998, 1999, 2000, 2001, 2002}

because the missing values can be determined automatically. By doing so, the plot makes clear that the 1998, 2000, and 2001 values had no facts recorded for them.

Only Temporal and Date dimensions can be utilized as the time dimension in a time-series plot since these are the only data types which represent time data. Therefore, only cubes which include a Temporal or Date dimension can be visualized in a time-series plot.

Time-series plots in *VisiCube* always utilize a dimension which defines the groups for which the measure is to be plotted. This *group dimension* is most commonly the location at which the time data was measured (the “space” part of the space-time continuum of our universe), but it need not be so. A separate time-series plot is generated for each instance of the group dimension. For example, for weather data collected on different dates at various weather stations, you would designate the date dimension as the “time” and the weather station dimension as the “group”. The result would be separate time-series plots for each weather station.

Since time-series plots use a relatively fixed time scale, such plots will be very long in cases where the time dimension spans a great range of values. To handle such cases, *VisiCube* provides you an ability to divide the plot into multiple sections. You specify the number of time instances that are to be included in each section and whether the sections are to overlap each other. (If overlap is desired, 10% of the instances at the end of a section will be repeated at the start of the next section.) Then *VisiCube* automatically divides the plot into the appropriate number of section.

Chapter 5 VisiCube SynchroViews

In this chapter, we describe each of *VisiCube's SynchroViews* and how they fit into the big picture. Note that details about usage of the windows in the individual *SynchroViews* are not included here. (These details are available in the online help system.)

The organization of this chapter generally matches that of the *Navigator* to facilitate cross-referencing between the product and this manual.

Organization (in the Navigator)

The *SynchroViews* are organized by, and accessed through, the *Navigator*. The structure of the *Navigator* is hierarchical and represents the structure of the basic objects upon which *VisiCube* operates. That hierarchy is divided into two primary areas which represent the types of objects with which you can interact: Data sources and Cube files.

Each *SynchroView* is accessed by an end node of the hierarchy and the name (and purpose) of that *SynchroView* can be determined by reading up the hierarchy from the node, itself, to the primary clustering node. For example, the “Brush” node...whose hierarchical path is Brush|Records|Cubes|Cube Files...represents the *SynchroView* which gives you the ability to brush the records of a cube in a cube file. In fact, it accesses the “Brush Records” *SynchroView*.

As can be seen, the “Data Sources” part of the hierarchy is quite small. This is because there is little you need to do with data sources. Fundamentally, you need only the ability to manage your data sources. Once the desired data is captured in a cube file, the real work of analysis begins. And it is here, in the “Cube Files” part of the hierarchy, where most of the *SynchroView* exist.

The “Cube Files” part of the hierarchy has three levels which represent the structure of a cube file. Remember that a cube file is composed of one, or more, cubes and that each cube, has its own set of visuals. The “Cube Files” part of the *Navigator* hierarchy has same three levels: Cube files, cubes, and visuals. Each of these clusters is further organized in a manner such that the name and use of the matching *SynchroView* is apparent (as noted above).

Manage Data Sources SynchroView

This *SynchroView* gives you the ability to manage all of the data sources cube files (regardless of their type) and specify the current data source for *VisiCube*.

The primary window of this *SynchroView*, the *Manage Data Sources* window, gives you the ability to accomplish the basic management tasks including select, add, copy, and remove. (You can change the selected data source by accessing the [Change Zoned Source SynchroView](#).) The other windows provide you views of the structure of the data source and the facts stored in it. Although you can initiate an “add” or “change” operation in this *SynchroView*, the actual type-specific definition of the data source is updated in the *SynchroView* which deals with data sources of the applicable type.

For example, if a zoned source is currently selected in this *SynchroView* and the **Change** button is clicked, you are transferred to the [Change Zoned Source SynchroView](#).

Although the definition of a data source is dependent on the type of physical data it utilizes, the resultant definition is one in which the data source can be represented as a two-dimensional table (consisting of rows and columns). Each row of the table represents a single record and the table can have any number of records (including none at all). Each column of the table represents a single indicator. There is at least one indicator in every data source (else there would be no data defined by the data source). Each indicator has a unique name within that data source and contains facts of one, and only one, type: String, Number, Temporal, or Date. Additionally, you can further define each indicator by specifying the unit of measure (for numeric indicators) and the value which is used to represent a null (or missing) fact.

Change Zoned Source SynchroView

This *SynchroView* gives you the ability to change the definition of a zoned data source (if it is the current data source as determined in the [Manage Data Sources SynchroView](#)).

A zoned file is a text file in which carriage returns mark the end of each record. Further, every indicator in each record is in the same absolute position and has the same length. I.e., the indicators in such files can be defined by vertical stripes through the records. All indicators in a zoned file are necessarily of fixed length.

To fully define a zoned source, you specify the zoned file that contains the applicable data and the zones in which the applicable indicators occur. The type of data found in each zone, in addition to the size of the zone, is defined so that VisiCube can process the data appropriately. The set of zones that comprise this part of the definition is known as the template of the zoned source. This template can be copied between zoned sources in cases where you have multiple zoned files with the same structure.

It is important to note that the physical file designated for the source may not be specified, accessible, or valid. (E.g., it might have just been created with an ADD and is incomplete or it might point to a file which is no longer existent or in the expected state. Or the drive, or disk in that drive, might not be available.) Such cases are handled with the appropriate messages in the windows and suppression of applicable functionality.

Manage Cube Files SynchroView

This *SynchroView* gives you the ability to manage cube files which are currently accessible and specify the current cube file for *VisiCube*.

Note that each cube file is a stand-alone physical file. I.e., it is completely self-defining. *VisiCube* keeps track of nothing about that cube file. All knowledge about the file is obtained from the file, itself. That's what makes them completely portable. *VisiCube* can only manage those files which it can actually locate and read from your computer system. Therefore, the list of manageable files includes only those that can be located on the system and which can be read from the system without error. This *SynchroView* displays only those cube files which are found in a single folder. (But, you have the ability to select the folder.)

The primary window of this *SynchroView*, the *Manage Cube Files* window, gives you the ability to accomplish the basic management tasks including copy, rename, and delete as well as selecting the current cube file. Note that the creation of new cube files is accomplished with the [Add Cube File SynchroView](#).

Since cube files are independent, stand-alone files, VisiCube provides no special facilities to move them about your computer system. As with any file, you may use your standard file management software to accomplish those tasks.

An additional capability of this *SynchroView*, is the ability to extract data from the selected cube file and use it to create a new zoned file and a zoned data source for that zoned file.

The *View Cube File* window provides you a view of the content of the current cube file. The fundamental component of a cube file is a single master cube which contains all of the desired data records and a structural definition which allows you to view that data in a meaningful manner. It is this master cube which is displayed in this window for your review.

Add Cube File SynchroView

This *SynchroView* gives you the ability to create a cube file from one or more data sources.

You are provided the ability to capture source data, build a master cube from that data, and save it in a new cube file. And you are given the ability to place that new file on your system at a location of your choice.

More than one data source can be utilized. At the time of creation, the data from each is concatenated together and processed as if it came from a single data source. Multiple data sources can only be utilized if they share the same template.

To determine whether individual data records are included in the master cube, they are filtered through the predicate list. Only those records which pass the predicates test are included in the new master cube.

At the time of creation, you are asked to specify the structure of the master cube by selecting dimensions and measures from the list of all indicators from the source data. Only numeric indicators (those that have a data type of “Number”) can be utilized as measures of the master cube. A minimum of one dimension and one measure must be selected. The names of the indicators from the first data source are used as the names of the dimensions and measures.

Note that the master cube is built from the data source but, after creation, is no longer connected to that data. I.e., changes to such a data source after creation of a cube file have no effect on the content of such a cube file. The master cube is, effectively, a snapshot of source data at the time the cube file is created.

During creation of the cube file, any data value which is determined to be invalid or null is recorded in the cube as a missing value. These include the following:

- A value that matches the “missing value indicator” for the indicator.
- A value that is not of the applicable data type for the indicator.

Define Measure SynchroView

This *SynchroView* gives you the ability to define any measure of the cube file.

The measures are created with default definitions when the cube file is created. If any of those definitions are not adequate for your purposes, you use this *SynchroView* to update them.

From the list of all measures in the cube file, you can select the measure whose definition is to be altered. This *SynchroView* is also your primary tool for reviewing the measures that are in the cube file.

Define Dimension SynchroView

This *SynchroView* gives you the ability to define the basic attributes of any dimension of the cube file. (Other *SynchroViews* are available to define the extended attributes of the dimension.) It can also be used to see the instances of the dimension which can be helpful when you are classifying or describing the dimension.

The dimensions are created with default definitions when the cube file is created. If any of those definitions are not adequate for your purposes, you use this *SynchroView* to update them.

From the list of all dimension in the cube file, you can select the dimension whose definition is to be altered. This *SynchroView* is also your primary tool for reviewing the dimensions that are in the cube file.

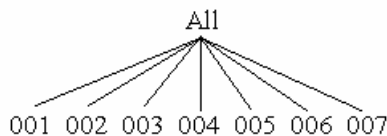
Classify Dimension SynchroView

This *SynchroView* gives you the ability to update the classification of any dimension of the master cube in the current cube file.

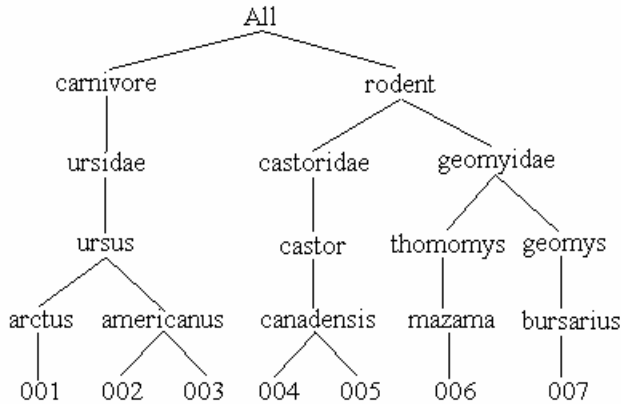
Every dimension is classified. At a bare minimum, that classification is a two-level hierarchy in which the lowest level is comprised of the domain coordinates (key values) and the highest level is the dimension, itself. Consider a “Specimen” dimension whose domain is

{001, 002, 003, 004, 005, 006, 007}

The default classification for the dimension would be the following:



However, any number of additional, intermediate levels of the classification hierarchy may be defined. Consider the following classification of the same specimens:



Note that the same coordinates (at the bottom of the hierarchy) are classified as in the previous example and that they all still remain members of the “All” taxon (at the top of the hierarchy). However, four new levels have been added to the hierarchy to represent the order, family, genus, and species of the specimens.

Note that each node in the hierarchy represents a taxon of the hierarchy and that each taxon has as its members those taxa from the next lower level of the hierarchy that map to the particular taxon. For example, the taxon “rodentia” has two members: “castoridae” and “geomyidae”.

To clarify this example, the definition of this “Specimen” dimension would be one that includes 5 properties as follows:

<u>Key</u>	<u>Species</u>	<u>Genus</u>	<u>Family</u>	<u>Order</u>
001	arctos	ursus	ursidae	carnivora
002	americanus	ursus	ursidae	carnivora
003	americanus	ursus	ursidae	carnivora
004	canadensis	castor	castoridae	rodentia
005	canadensis	castor	castoridae	rodentia
006	mazama	thomomys	geomyidae	rodentia
007	bursarius	geomys	geomyidae	rodentia

Each level of the hierarchy is a classification property of the dimension and, therefore, can be utilized in the visuals just like any other property of the dimension. For example, each such property can be used as the label for the dimension in visuals which utilize labels. Also, a subcube could be built which contains only carnivores without having to explicitly select each applicable ID when building that subcube.

Classification involves the explicit assignment of taxa from the lower level classification property to taxa in the new classification property. In our example, the “002” and “003” keys were assigned to the “americus” species while “004” and “005” were assigned to “canadensis”.

Describe Dimension SynchroView

This *SynchroView* gives you the ability to describe a dimension of the master cube in a cube file.

The description of a dimension includes the descriptive properties of the dimension (those beyond the key property and the classification properties of the dimension).

Consider a cube file, named “Big Trees”, whose master cube consists of three dimensions (Date, State, and Type) and one measure (Size) as follows:

<u>Date</u>	<u>State</u>	<u>Type</u>	<u>Size (ft)</u>
1969	HI	Acacia	140
1971	MI	American Basswood	115
1979	MI	Yellow Birch	107
1972	GA	Painted Buckeye	144

Unless described (or classified) further, the State dimension is comprised of a single property which contains a list of the key values for the instances of the domain:

Key
GA
HI
MI

Further, this single property must be used as the label for the dimension (since there are no other properties of the dimension).

It might be the case that you wish to use state names, instead of postal codes, as labels for the dimension. To do so, you would need a data source that contains information such as:

<u>Code</u>	<u>Name</u>
AL	Alabama
GA	Georgia
HI	Hawaii
IL	Illinois
MI	Michigan

Assuming this data source is keyed on the “Code” indicator, it can be used to describe the State dimension in our example. The additional indicators from the data source would be added to the dimension’s description as additional description properties. In this case, the “Name” indicator would be used to form a descriptive “Name” property of the dimension. Each record in the data source whose key indicator has the same value as the key property for one of the instances of the dimension would be used to populate the additional properties. The dimension would then be comprised of two properties as follows:

Key Name
GA Georgia
HI Hawaii
MI Michigan

Manage Cubes SynchroView

This *SynchroView* gives you the ability to manage the cubes of the current cube file and to decide which is to be considered the current cube for further definition and all visualization purposes.

The primary window of this *SynchroView*, the *Manage Cubes* window, gives you the ability to accomplish the basic management tasks including delete. You can also update the notes for the selected cube. (To add a new cube to the list, use the [Add Cube SynchroView](#).) The other windows provide you a view of the current cube so that you can be sure you have selected the desired one.

Add Cube SynchroView

This *SynchroView* gives you the ability to create a new cube in the current cube file.

It must be understood that the master cube is a valid cube and can be analyzed itself. There is no need to create additional cubes in a cube file unless you find it helpful in your analysis to do so. Each additional cube is a subset of the master cube and is often referred to as a subcube. In other words, it can contain no measures, dimensions, nor instances of those dimensions which are not also present in the master cube. However, it can be the case that the master cube contains instances which you do not wish to include in a particular analysis. In such a case, you would need to create a subcube that contains only the instances of interest. It can also be helpful to utilize simpler subcubes in cases where the master cube has a great number of dimensions or measures.

In any case, the new cube will be a subset of the master cube in the cube file and will consist of the instances, dimensions, and measures which you specify. Specifying the dimensions and measures is a straightforward process (through simple selection lists), but specifying the instances requires a fair amount of thought and effort.

VisiCube supports the definition of the desired subset of instances through a recursive mechanism. A fundamental principle of set theory is that you can specify any complex subset by combining simpler subsets with union and intersection operations. Recursion implies that these simpler subsets can, themselves, be specified through a combination of even simpler subsets. And so on. In the end, the complex subset can be specified through a combination of many very simple subsets.

For example, from a master cube containing rainfall data for North America in the 20th century, you may wish to construct a new cube consisting of the following instances:

```
(
  (
    (State IN ['Florida'])
    AND
    (County IN ['Dixie', 'Liberty', 'Pasco'])
  )
  AND
  (
    (Rainfall BETWEEN 4.0 AND 9.9)
    OR
    (Year IN [1960])
  )
)
```

Note that each set of parentheses in the above example represents a subset. The innermost subsets, those defined by predicates, are the most granular part of the specification. We refer to these as atomic subsets because they are built from a single set...the master cube, itself. All of the other subsets are built with a standard set operation of union (OR) or intersection (AND). We refer to these as binary subsets because they are built from two other sets. The outermost subset is the one that is to be used in creating the subcube.

A predicate utilized by an atomic subset is defined against a single property of a single dimension. Note that both must be specified. In the above example, note that the State property of the Location dimension was used in the first predicate. (The one that identifies the Florida instances of the data.)

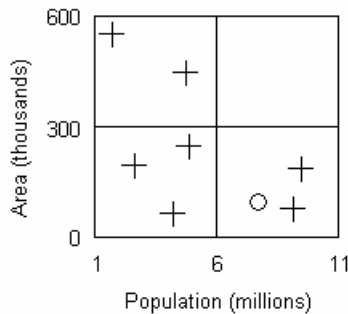
While it is often the case that a dimension has only a single property, it is still the property that actually contains the values which can be utilized in comparison tests (such as these predicates). Therefore, you must specify the predicate as well as the dimension.

Brush Records SynchroView

This *SynchroView* gives you the ability to define how the records of the current cube are to be brushed in the various visuals for that cube.

It is often desirable to be able to visually distinguish specific data records from the others presented in a plot. *Brushing* is a technique in which these records are defined for use in a plot. It enables you to identify specific records in a manner that is independent of a single dimension and its instances.

Consider the following scatter plot in which population is plotted against land area for eight countries. It utilizes brushing of the Austrian record so that it is displayed as an “o”, instead of a “+”, and can easily be located in the plot.



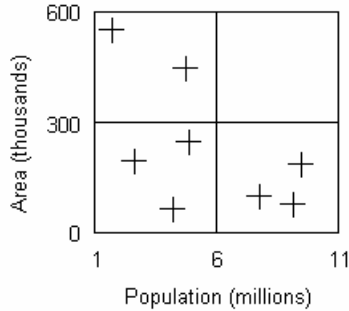
VisiCube allows you to brush the most important records (individually or as a group) of the current cube in this *SynchroView* and, through the various visuals, dictate when such brushing is to be utilized. The approach utilized is one in which you select the desired records from a list or click on the matching data points in a scatter plot for those records.

Separate Dimension SynchroView

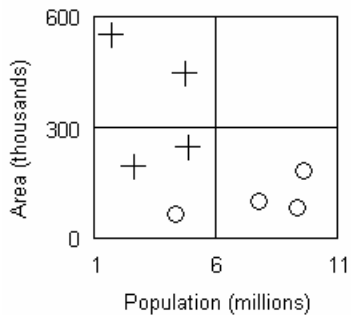
This *SynchroView* gives you the ability to define how the instances of any dimension of the current cube are to be separated in the various visuals for that cube.

Often, the set of data points presented in a plot can be further divided into logical subsets and it is often the desirable to be able to visually separate these subsets. *Separation* is a technique in which these subsets are assigned different encoding mechanisms (such as color, shape, or pattern) for use in a plot. Note that separation utilizes logical subsets which are based on dimension values whereas the related technique of brushing (described below) utilizes physical subsets which are based on records.

Consider a simple scatter plot of eight African and European countries, which shows the records and their distribution plainly:



However, separating the data by continent (using “+” Africa and “o” for Europe) results in the following, more informative, plot which clearly indicates that the geographically larger countries are in Africa while the more populous countries are in Europe:



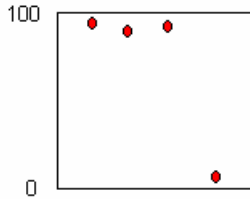
VisiCube allows you to define the separation to be used for each dimension of the current cube in this *SynchroView* and, through the various visuals, dictate when such separation is to be utilized. The approach utilized is one in which you select the desired instances from a list and then select a set of encoding symbols to be utilized for those instances.

Limit Measure SynchroView

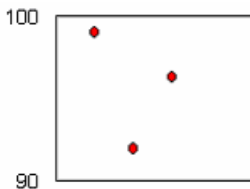
This *SynchroView* gives you the ability to define how the facts of any measure of the current cube are to be limited in the various visuals for that cube.

A measure may include extreme values which can distort visualizations and statistics. These values are often referred to as *outliers* because they tend to lie outside of the range of expected values. Sometimes, such values are the result of bad measurements or recordings of those measurements. On the other hand, they may be valid but are simply so extreme that they adversely affect the analysis of the remainder of the data.

Consider four values which are expected to fall between 90 and 100. If the last value were mistakenly recorded as 4 instead of 94, the resulting plot (below) would greatly obscure the differences between the normal values and hinder your ability to analyze the measure.



If the displayed values are limited to those in the expected range, the last value would be eliminated and the resulting plot (below) would facilitate, instead of hinder, the analysis of the other values.



In this *SynchroView*, *VisiCube* allows you to define outliers for each measure of the current cube and, through the various visuals, control when those outliers are presented. This is accomplished by defining upper and/or lower limits of acceptable values for the measure. All values outside of those limits are considered outliers. A quantile plot is utilized to show you the effect of the desired limits before, and after, they are actually set.

View Univariate Records SynchroView

This *SynchroView* gives you the ability to view records of the current cube with the aid of a univariate plot.

It is often desirable to see all of the facts regarding a particular record utilized in a univariate plot. Those facts are presented in this *SynchroView* through a list in which each fact is identified with the name of the corresponding measure or dimension.

The *SynchroView* provides you the ability to select the desired record in either of the following manners:

- You can select the record from a list of those displayed in the plot; or
- You can select the data point in the plot.

Define Box Array SynchroView

This *SynchroView* gives you the ability to define the box array visual for the current cube. This visual is an array of box plots, one for each desired instance of a single dimension, for one measure of the cube. It is available for all cubes.

The plots are generated in a side-by-side manner with a single scale to facilitate immediate and direct comparison of the distributions for each of those instances. You specify the measure, the dimension, and the instances of that dimension that are to be plotted. *VisiCube* then generates an array of box plots...one for each instance. This visual provides you a mechanism through which you can compare the statistical summaries of measurements for each instance.

An additional, single quantile plot is also displayed. The box plot set is dynamic in that you can select any of the individual plots and its values will be displayed in this quantile plot so that you can analyze the distribution of those values in more detail.

Define Quantile/Box Plot SynchroView

This *SynchroView* gives you the ability to define the quantile/box visual for the current cube. This visual is a pair of complementary plots for one instance of a single dimension and one measure of the cube. The first is a quantile plot which shows the distribution of the measurements for the instance. The second is a box plot which shows the statistical summary of the measurements for the instance. This visual is available for all cubes.

You specify the measure, the dimension, and the instance of that dimension that are to be plotted. *VisiCube* then generates the plots using a common scale so that the statistical values can be compared directly to each data point in the distribution.

Define Quantile Array SynchroView

This *SynchroView* gives you the ability to define the quantile array visual for the current cube. This visual is an array of quantile plots, one for each desired instance of a single dimension, for one measure of the cube. It is available for all cubes.

The plots are generated in a grid with a single scale to facilitate immediate and direct comparison of the distributions for each of those instances. You specify the measure, the dimension, and the instances of that dimension that are to be plotted. *VisiCube* then generates an array of quantile plots...one for each instance. This visual provides you a mechanism through which you can compare the distributions of measurements for each instance.

View Multivariate Records SynchroView

This *SynchroView* gives you the ability to view records of the current cube with the aid of a multivariate plot.

It is often desirable to see all of the facts regarding a particular record utilized in a multivariate plot. Those facts are presented in this *SynchroView* through a list in which each fact is identified with the name of the corresponding measure or dimension.

The *SynchroView* provides you the ability to select the desired record in either of the following manners:

- You can select the record from a list of those displayed in the plot; or
- You can select the data point in the plot.

Define 2D Scatter Plot SynchroView

This *SynchroView* gives you the ability to define the 2D scatter plot for the current cube. This visual is a single scatter plot of two measures of the cube. It is available only for cubes in a cube file which has at least two measures.

Since the scatter plot represents each of the two measures with a separate, perpendicular axis, the scatter plot is, by definition, two-dimensional. You specify the two measures that are to be plotted and *VisiCube* generates the scatter plot. This visual provides you a mechanism through which you can analyze the amount of correlation between the two measures. The higher the correlation, the more the set of plotted points will approximate a line.

Define 3D Scatter Plot SynchroView

This *SynchroView* gives you the ability to define the 3D scatter plot for the current cube. This visual is a single scatter plot of three measures of the cube. It is available only for cubes in a cube file which has at least three measures.

Since the scatter plot represents each of the three measures with a separate, perpendicular axis, the scatter plot is, by definition, three-dimensional. And, since a three-dimensional scatter plot cannot be represented directly in the two-dimensional interface of *VisiCube*, a layering technique is utilized which allows you to view the records in each layer separately and see how they compare to the other records in the plot.

You specify the two measures which are to be utilized as the axes of the base of the plot and a third measure which is to be utilized as the vertical axis of that plot. You also specify the number of layers into which that vertical measure is to be divided. *VisiCube* then calculates the size and range of each layer, determines the layer to which each record belongs, generates the scatter plot, and provides you two views of that plot. The first indicates the current layer of interest (using a simple, stylized 3D cube representation). The second is a vertical view (perpendicular to the base) which shows all data points as they are plotted against the base measures. Those records which occur in the current layer of interest are highlighted while all other records are not.

You may change the current layer in either of the following manners:

- You may select the desired layer from a list of all the layers; or
- You may turn on the animation feature of the visual (which causes the layer to be changed regularly at a speed of your choosing).

This visual provides you a mechanism through which you can analyze the amount of correlation between the vertical measure and the two base measures. The higher the correlation, the more the highlighted records will tend to flow across the vertical view as the plot is animated.

Define Scatter Matrix SynchroView

This *SynchroView* gives you the ability to define the scatter matrix for the current cube. This visual is a matrix of 2D scatter plots for measures of the cube. It is available only for cubes in a cube file which has at least two measures.

You select all of the measures which are to be analyzed. *VisiCube* generates the matrix such that each combination of two of those selected measures provides the basis for a pair of the scatter plots, one being the reflection of the other.

The matrix gives you an ability to compare the various scatter plots in a single visual. Each column and row corresponds to one of the selected measures. A measure can be studied in relation to the other measures in the matrix by scanning along the row (or column) of the matrix. In this manner, the relationship of a single measure may be understood throughout a series of distributions.

It may be useful to note that the series of squares rising diagonally from the bottom left to the top right of the matrix identify the row and column they join. This identification includes the name of the measure, the unit of measure (if any is specified), and indicates whether jittering is being utilized for the measure.

View Time-Series Records SynchronoView

This *SynchronoView* gives you the ability to view records of the current cube with the aid of a time-series plot.

It is often desirable to see all of the facts regarding a particular record, or set of records, utilized in a time-series plot at a given point in time. Those facts are presented in this *SynchronoView* through a list in which each record is presented as a row in the list and the dimensions and measures of the records are presented as columns.

The *SynchronoView* provides you the ability to select the desired time point directly from the time-series plot.

Define Multi-Measure Time-Series Plot SynchronoView

This *SynchronoView* gives you the ability to define the multi-measure visual for the current cube. This visual is an array of time-series plots, one for each desired measure, for one instance of a single dimension of the cube. It is available only for cubes which include two dimensions including one that can be used as a time dimension.

You specify the dimension which is to be utilized as the time dimension. You also specify the dimension to be utilized as the group dimension and the instance of that dimension for which the time-series plots are to be generated. Finally, you specify the measures for which individual plots are to be generated. *VisiCube* then generates the array of plots.

You also have the ability to manage the sectioning that is used for the time dimension (since it is often the case that the time span will be too large to fit in a single, reasonably sized plot).

Define Multi-Group Time-Series Plot SynchroView

This *SynchroView* gives you the ability to define the multi-group visual for the current cube. This visual is an array of time-series plots, one for each desired instance of a single dimension, for one measure of the cube. It is available only for cubes which include two dimensions including one that can be used as a time dimension.

You specify the dimension which is to be utilized as the time dimension. You also specify the dimension to be utilized as the group dimension and the instances of that dimension for which the individual plots are to be generated. Finally, you specify the measure that is to be plotted in each of the time-series plots. *VisiCube* then generates the array of plots.

You also have the ability to manage the sectioning that is used for the time dimension (since it is often the case that the time span will be too large to fit in a single, reasonably sized plot).

Chapter 6 Examples

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 NT (4.0), Windows 2000, Windows XP, or any later version of these Windows operating systems.
- 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, or better, resolution.

Appendix B: VisiCube Limitations

Limitations of VisiCube

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

- The maximum number of different subsets that can be visualized on any one plot is 6.
- The maximum number of groups that can be utilized in a time-series plot is 12.
- The minimum number of time instances in a single section of a time-series plot is 50.
- The maximum number of time instances in a single section of a time-series plot is 1000.
- The minimum number of layers that can be utilized in a 3D scatter plot is 2.
- The maximum number of layers that can be utilized in a 3D scatter plot is 100.
- The minimum supported date is January 1, 100.
- The maximum supported date is December 31, 9999

Limitations of Version 0

The following is a list of the specific limitations of Version 0 of *VisiCube* which will not exist in the fee version of the product:

- Version 0 only supports zoned text files as data sources. However, you can get around this limitation by exporting data from other sources (such as spreadsheets, databases, and statistical package files) into zoned text files. Most software packages that manage data support export capabilities of this sort.

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 *SynchroViews*) 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 Interval 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* has been added.
- The *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 *SynchroViews* (including *Single Quantile / Box Plot SynchroView*, *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 *SynchroViews* 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 *SynchroViews*.
- 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.

Version 0.6: Released January 4, 2003

Highlights:

- Implemented abstract data sources to provide for reusability of physical files and compatibility with different types of data sources.
- Provide ability to determine structure of cube file when it is generated by limiting the dimensions and measures that will be included in it.
- Eliminated cube file library so that cube files are now usable without prior definition to *VisiCube*.
- Support Date and Temporal data types.
- Categorization replaced with more general classification techniques.
- User interface improved with more intuitive elements.
- Navigator reorganized and simplified.
- Manual expanded.

Other:

- Added ability to update definition of a measure after a cube file is created.
- Changed cube file deletion logic so that the deleted file is now captured in the system recycling bin so that they can be recovered in case of a mistake.
- Replaced Outlier Facility SV with Limit Measure SV for consistency with new NAV architecture.
- Terminology synchronized between manual and product.

Appendix E: Sample Cube Files

There are seven sample cube files distributed with the *VisiCube*. They are described in more detail here.

Brain sizes and intelligence

Willerman, L., Schultz, R., Rutledge, J. N., and Bigler, E. (1991), In Vivo Brain Size and Intelligence, *Intelligence*, 15, 223-228.

Willerman et al. (1991) collected a sample of 40 right-handed Anglo introductory psychology students at a large southwestern university. Subjects took four subtests (Vocabulary, Similarities, Block Design, and Picture Completion) of the Wechsler (1981) Adult Intelligence Scale-Revised. The researchers used Magnetic Resonance Imaging (MRI) to determine the brain size of the subjects. Information about gender and body size (height and weight) is also included. The researchers withheld the weights of two subjects and the height of one subject for reasons of confidentiality.

Three species of iris

Anderson, E, The Irises of the Gaspé Peninsula, *Bulletin of the American Iris Society*, 59, 2-5, 1935.

Measurements of 3 species of Iris. This data was used in a landmark paper on mathematical classification by R. A. Fisher (The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7, 179-188, 1936).

Increased incidence of malignant melanoma

A. Houghton, E. W. Munster, and M. V. Viola. Increased Incidence of Malignant Melanoma After Peaks of Sunspot Activity. *The Lancet*, 8, 757-760, 1978.

Egyptian Skulls

Thomson, A. and Randall-Maciver, R. (1905) Ancient Races of the Thebaid, Oxford: Oxford University Press.

Also found in Hand, D.J., et al. (1994) A Handbook of Small Data Sets, New York: Chapman & Hall, pp. 299-301.

Also found in Manly, B.F.J. (1986) Multivariate Statistical Methods, New York: Chapman & Hall.

Four measurements of male Egyptian skulls from five different time periods. Thirty skulls are measured from each time period.

American Colleges

U. S. News and World Report's Guide to America's Best Colleges (1995).

This data set was used for the 1995 Data Analysis Exposition sponsored by the Statistical Graphics Section of the American Statistical Association.

California Weather Data

Extracted from the Global Surface Summary of day Data, Version 6, produced by the National Climatic Data Center (NCDC) in Asheville, NC.

This data is provided online by the *National Oceanic and Atmospheric Administration* (NOAA) from their website at www.ncdc.noaa.gov.

Nigeria Temperature Data

Extracted from the Global Historical Climatology Network (GHCN) Temperature Database, version 2.

This data is provided online by the *National Oceanic and Atmospheric Administration* (NOAA) from their website at www.ncdc.noaa.gov.

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.

brushing

The visualization technique utilized to highlight the most important records in a plot.

box plot

A plot in which the statistical summary of a measure is graphed.

cardinality

The number of members in a set (such as a domain).

cell

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

taxon

A unique value used in a classification property (for a dimension). For example, a “Region” property utilized in a classification may include taxa (values) of “West”, “East”, “North”, and “South”.

classification

A hierarchy utilized to classify the instances of a dimension.

classification property

A description property which is part of the classification hierarchy for a dimension.

comparand

One or more values to which an expression is compared in a predicate. The composition of the comparand is dependent on the comparator.

comparator

An operator used in a predicate to compare an expression and a comparand. The comparison results in an assertion of truth, either true or false.

coordinate

A value of the identification property of a dimension. Each coordinate uniquely identifies an instance of the domain. It is used to locate a cell along one dimension of a cube.

cube

A multidimensional structure, comprised of cells, into which data is organized. This is a logical construct which is sometimes referred to as a hypercube in other literature.

cube file

A file with a proprietary format in which *VisiCube* stores data, in the form of a master cube, and supplementary cubes which are built from subsets of the master cube. Such a file is created and altered only by *VisiCube* and is fully portable between installations of *VisiCube*.

data

A general term describing information which can be analyzed with *VisiCube*.

data file

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

data record

See “record”.

data source

A source of data for *VisiCube*. This is a logical construct which identifies and defines the physical source of your. This data is usually obtained from a single survey. The physical source is typically a data file. The definition of the source includes instructions on how the data can be obtained 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 record and each column is a distinct indicator.

data type

An attribute that describes the type of facts contained in an indicator, dimension, or measure.

datamology

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

datamologist

You.

date

A fact which represents a calendar date.

delimited file

A text file which contains a set of data records. Further, the indicators of each data record are delimited by a specific character. Typically, the delimiting character is a comma or a tab. (See also “zoned file”.)

description property

A single attribute of a dimension which contains values that describe the particular instances of the dimension.

dimension

An axis through the space of a cube. The domain of a dimension is the set of distinct facts for the indicator from which the dimension is generated. (See also “measure”.)

domain

The set of unique key values which identify every possible coordinate of a dimension.

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).

expression

A symbol representing a data value to which a comparand is compared in a predicate. Typically the name of an indicator, dimension, or measure.

fact

A discreet value which is recorded for a given indicator in a given record and, generally, cannot be decomposed into smaller meaningful parts.

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).

group dimension

The dimension designated to contain group data in a time-series plot.

hypercube

See “cube”.

identification property

The classification property at the base of the classification hierarchy for a dimension. The values of this property are the coordinates of the dimension. I.e., those values which uniquely identify the instances of the dimension.

indicator

A single attribute of a phenomenon whose value is determined during an observation. The value is a fact which is stored in a record.

instance

An entity that is identified by a coordinate of a dimension and described by a set of values from the dimension's definition. For example, the United States of America may be one of the instances of a "Country" dimension. And the definition might include a "Key" property of "U.S." and a "Hemisphere" property of "Western".

intersection

A set operation (utilizing the "AND" connective) which produces a set whose members are in each of the original sets.

interval (scale)

A type of scale used to determine values for facts which are continuous.

jittering

A visualization technique utilized to make visible those facts which would otherwise be hidden in a plot. The particular technique is one in which the data points in the plot are adjusted by a random, but slight, amount so that they no longer obscure each other.

key (indicator)

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

key (of a dimension)

The property which contains the distinct facts from the indicator which was used to define the instances of a dimension when it was created.

keyed source

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

label

A property (of a dimension) whose values are used to identify data in various visuals.

limiting

The visualization technique utilized to exclude extreme measurements (known as outliers) from a plot so that the remaining, non-extreme values can be better analyzed.

master cube

The primary cube in a cube file. It contains all of the data that is captured in the cube file. Its dimensions and measures dictate the structure of any subcubes included in the cube file. Every cube file has one, and only one, master cube.

measure

An indicator whose facts are used to populate the cells of a cube. (See also "dimension".)

measurement

The value (fact) of a particular measure in a given record.

member

A single entry in a set.

Navigator

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

nominal

A type of scale used to determine values for facts which are used for identification purposes only.

non-keyed source

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

number

A fact which represents a numeric value.

observation

The recorded information obtained from the observing a particular phenomenon in a particular survey.

ordered set

A set in which the order of the members is an identifying property of the set. It is represented by a list enclosed in brackets such as: [male, female].

ordinal

A type of scale used to determine values for facts which have an inherent order.

outlier

A fact which falls outside of the specified limits of measure. Such facts may be the result of incorrect measurements or recording errors.

phenomenon

A circumstance, specimen, sample, experience, or the like that is apparent to the senses and that can be scientifically described or appraised.

plot

A primitive graph used as the basis for more complex visuals in *VisiCube*.

predicate

A specification which is affirmed or denied with regard to a data record when determining whether to utilize the record when building a new set of data. Predicates are always of the form:

expression comparand comparator

Examples include:

Year = 1998

Precipitation BETWEEN (1.00 AND 2.99)

State IN (Vermont, New Hampshire, Maine)

See also “expression”, “comparator”, and “comparand”.

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*.

property

Any description property (which, therefore, includes classification and identification properties) of a dimension. (This is analogous to an indicator with regard to a phenomenon.)

ratio (scale)

A type of scale used to determine values for facts which have relative magnitude.

record

An ordered set of facts pertaining to a single observation where each fact belongs to one, and only one, indicator. (See also “indicator”.)

quantile plot

A plot in which the distribution of a measure is graphed.

scale

An instrument used to record a value to represent a fact.

scatter plot

A plot in which two measures are graphed against each other.

separation

The visualization technique utilized to distinguish data for different sets of instances in a single plot.

serial

A type of scale used to determine values for facts which are regular. I.e., their differences are meaningful.

set

A collection of any number of members. It is represented by a list enclosed in braces such as: {male, female}.

string

A fact which represents a sequence of characters (which is neither a date, a number, nor a temporal).

study

A process in which you apply your mind in order to acquire knowledge through the careful and critical investigation and examination of a particular subject.

subcube

A cube in a cube file which is constructed from the master cube in the same cube file. The contents of the subcube are a subset of the contents of the master cube.

subset

A relationship between two sets where every member of the subset is a member of the other set.

survey

A set of related observations which are obtained as part of a study with regard to a single type of phenomenon.

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*.

taxon

A unique value used in a classification property (for a dimension). For example, a “Region” property utilized in a classification may include taxa (values) of “West”, “East”, “North”, and “South”.

template

A logical structure which defines the contents of a data source. It is comprised of a set of indicators in a specific order. Each indicator has a given data type. Two templates are compatible if the number of indicators match and each relative indicator has the same data type.

temporal

A fact which represents a temporal value. I.e., one that represents a single value in a time sequence (such as the months of a year).

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.

time dimension

The dimension designated to contain time data in a time-series plot.

time-series plot

A plot in which a measure is plotted against time in a serial manner.

union

A set operation (utilizing the “OR” connective) which produces a set whose members are in any of the original sets.

visual

A complex graphical presentation of data in a cube which is typically comprised of one, or more, plots.

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. Further, the indicators 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.