

## The User's Reference Guide for



**Forrest W. Young, Pedro Valero-Mora & J. Gabriel Molina**



## The User's Reference Guide for



Forrest W. Young, Pedro Valero-Mora & J. Gabriel Molina

**Reference:**

Young, Forrest W., Valero-Mora, Pedro, & Molina, J.Gabriel. The User's Reference Guide for ViSta 7 – The Visual Statistics System". The Visual Statistics Project ([www.visualstats.org](http://www.visualstats.org)). March, 2003. , Pittsburgh, PA, USA and Valencia, Spain.

**Documentation Copyright:**

Copyright © by Forrest W. Young, Pedro Valero-Mora & J. Gabriel Molina, 2003. All rights reserved.

**Contact Information:**

Forrest Young: [forrest@visualstats.org](mailto:forrest@visualstats.org)  
Pedro Valero-Mora: [pedro@visualstats.org](mailto:pedro@visualstats.org)  
Gabriel Molina: [gabi@visualstats.org](mailto:gabi@visualstats.org)  
Newsgroup: [vista@unc.edu](mailto:vista@unc.edu)  
ViSualStats: <http://www.visualstats.org/>

**Software Copyrights:**

XLISP-PLUS version 3.04.

Portions Copyright © 1988, by David Betz. Modified by Thomas Almy and others.

XLISP-STAT Release 3.52.13 (beta)

Copyright © 1989-1999, by Luke Tierney.

WIN-XLISPSTAT Release 3.52.13.1 (beta)

Modifications by Fabian Camacho and Forrest W. Young.

Parcil Parse C into Lisp.

Copyright © 1992 by Erann Gat. All rights reserved. Used under terms of GNU General Public License, Free Software Foundation.

ViSta – The Visual Statistics System.

Copyright © 1992-2003 by Forrest W. Young. All rights reserved.

ViVa - ViSta's Interactive Vector Algebra System.

Copyright © 1998-2003 by Forrest W. Young. All rights reserved.

**Version and Date:**

Version: ViSta 7.2.3 (EWD.2003.02.25.8030)

Printed: February 26, 2003



**seeing data**  
**asking truth**  
**gaining insight**

first, you see your data for what they seem to be  
then you ask them for the truth -  
are you what you seem to me?

you see with broad expanse, you ask with narrow power  
you see and ask and see  
and ask and see ... and ask

with brush you paint the possibilities  
with pen you scribe the probabilities  
for in pictures we find insight  
while in numbers we find strength

- forrest young



# User's Reference Guide for



Forrest W. Young, Pedro Valero-Mora & J. Gabriel Molina

## Part I: Welcome

<b>Topic 1 Welcome User!</b>	<b>7</b>
1.1 Welcome!	
1.2 Vista Preview	
<b>Topic 2 Welcome Developer!</b>	<b>10</b>
2.1 Welcome Developer!	
2.2 Developer Features	
2.3 Application Development	
2.4 System Development	

## PART II: GETTING STARTED

<b>Topic 3 Getting Started</b>	<b>10</b>	<b>12</b>
3.1 Getting Started		
3.2 Installing Vista		
<b>Topic 4 Using The Desktoo</b>	<b>14</b>	
4.1 The Desktop		
4.2 The Menu Bar		
4.3 Pop-Up Menus		
4.4 The Toolbar		
4.5 The Workmap		
4.6 The Datasheet		
4.7 The Selector		
<b>Topic 5 Using DataSheets</b> (partially written)		
<b>Topic 6 Using Plots and Views</b> (not written)		
<b>Topic 7 Using Excel</b>	<b>21</b>	

- 5.1 Excel And Vista
- 5.2 From Excel To ViSta
- 5.3 5.3 From Vista To Excel
- 5.4 Notes About Using Excel
- 5.5 Changing Excel'S Vista Menu

**Topic 8 Entering Data 30**

- 7.1 Editing Data
- 7.2 Vista'S Datasheet
- 7.3 Excel Spreadsheets
- 7.4 Sas Datasets
- 7.5 Importing Data
- 7.6 Simulating Data
- 7.7 Writing Datacode

**PART III: UNDERSTANDING DATA**

- Topic 8 Managing Data 43**
- 8.1 Managing Data
  - 8.2 Variable And Data Types
  - 8.3 Manipulating Variables
  - 8.4 Using Viva
  - 8.5 Making Variables
  - 8.6 Making Data
- Topic 9 Graphing Data 59**
- 9.1 Graphing Your Data
  - 9.2 Plot Menu Overview
  - 9.3 Box, Diamond And Dot Plots
  - 9.4 Histogram, Distribution Plots
  - 9.5 Cumulative Plot
  - 9.6 Comparison Plot
  - 9.7 Mosaic Plot And Bar Chart
  - 9.8 Scatter Plot
  - 9.9 Spinning Points Plot
  - 9.10 Scatter Matrix
  - 9.11 Orbiting Points Plot
- Topic 10 Analyzing Data 68**
- 10.1 The Analyze Menu
  - 10.2 Analysis Of Variance ...
  - 10.3 Regression Analysis ...
  - 10.4 Univariate Analysis ...
  - 10.5 Frequency Analysis ...
  - 10.6 Log Linear Analysis ...
  - 10.7 Multivariate Regression ...
  - 10.8 Principal Components ...
  - 10.9 Item Analysis ...
  - 10.10 Correspondence Analysis ...
  - 10.11 Homogeneity Analysis ...
  - 10.12 Metric Averaged Mds ...
  - 10.13 Multidimensional Scaling ...
  - 10.14 Cluster Analysis ...
  - 10.15 Impute Missing Data ...
- Topic 11 Modeling Data 76**
- 11.1 The Model Menu
  - 11.2 About The Analysis
  - 11.3 Visualize Model
  - 11.4 Report Model ...
  - 11.5 Interpret Model
  - 11.6 Save Model As ...
  - 11.7 Delete Model
  - 11.8 Create Data ...
  - 11.9 Print Data Analysis ...

## PART IV: ENHANCING VISTA

<b>Topic 12</b>	<b>Enhancing Vista</b>	<b>78</b>
12.1	Enhancing Vista	
12.2	Application Development	
12.3	System Development	
12.4	Writing Applets	
12.5	Writing Plots	
12.6	Writing Spreadplots	
12.7	Writing Plugins	
12.8	Writing Updates	
12.9	About Datatypes	
12.10	About Filetypes	

## About The Menus

### The File Menu 100

- 1 The File Menu
- 2 New Data ...
- 3 Open Data ...
- 4 Simulate Data ...
- 5 Import Data ...
- 6 Export Data ...
- 7 Save Data As ...
- 8 Save Model As ...
- 9 New Edit...
- 10 Open Edit ...
- 11 Load Edit ...
- 12 Print File ...
- 13 Print Active Pane...
- 14 Print Entire Window
- 15 Exit

### The Edit Menu 109

- 1 The Edit Menu
- 2 Cut
- 3 Copy
- 4 Paste
- 5 Clear
- 6 Copy-Paste

### The Data Menu 111

- 1 The Data Menu
- 2 About These Data
- 3 Data Header
- 4 Visualize Data ...
- 5 Summarize Data ...
- 6 List Data
- 7 Create Stats Object
- 8 Create Data Object
- 9 Delete Data Object ...
- 10 Impute Missing Data ...
- 11 Edit Data
- 12 Edit Source
- 13 Edit Variables
- 14 Print Data
- 15 Print Source
- 16 Merge Variables
- 17 Merge Observations
- 18 Merge Matrices

### The Transform Menu 124

- 1 The Transform Menu
- 2 Sort-Permute ...
- 3 Ranks
- 4 Normal Scores
- 5 Absolute Value
- 6 Rounding ...
- 7 Reciprocal
- 8 Exponential ...
- 9 Logarithm ...
- 10 Trigonometric ...
- 11 Correlations

### The Transform Menu (continued) 124

- 12 Covariances
- 13 Distances
- 14 Transpose Data

- 15 Standardize Data ...
- 16 Orthogonalize Data
- 17 Folded Power
- 18 Box-Cox Power
- 19 Dummy Coding
- 20 Split By Categories
- 21 Join Category Variables
- 22 Program Editor

**The Analyze Menu 131**

- 1 The Analyze Menu
- 2 Analysis Of Variance ...
- 3 Regression Analysis ...
- 4 Univariate Analysis ...
- 5 Frequency Analysis ...
- 6 Log Linear Analysis ...
- 7 Multivariate Regression ...
- 8 Principal Components ...
- 9 Item Analysis ...
- 10 Correspondence Analysis ...
- 11 Homogeneity Analysis ...
- 12 Metric Averaged Mds ...
- 13 Multidimensional Scaling ...
- 14 Cluster Analysis ...
- 15 Impute Missing Data ...

**The Model Menu 138**

- 1 The Model Menu
- 2 About The Analysis
- 3 Visualize Model
- 4 Report Model ...
- 5 Interpret Model
- 6 Save Model As ...
- 7 Delete Model
- 8 Create Data ...
- 9 Print Data Analysis ...

**The Options Menu` 141**

- 1 The Options Menu
- 2 Preferences ...
- 3 Run Excel ...
- 4 Record Listener
- 5 Repair Font
- 6 Renew Filetypes
- 7 Refresh System
- 8 Developers Mode

**The Help, Desktop and Window Menus 146**

- 1 The Help Menu
- 2 The Desktop Menu
- 3 The Window Menu

## WELCOME

WELCOME to ViSta - The Visual Statistics System. ViSta is ready to help you "Have fun seeing what your data seem to say". ViSta's fun and playful approach to data analysis helps you see what your data seem to say, and to test the truthfulness of what you think you've seen.

ViSta is more than a statistics system, it is your statistical advisor, consultant and teacher --- ViSta is your own personal statistician! ViSta is designed to help you learn about your data --- and to help you learn about how to learn about your data. ViSta will help teach you about:

- SEEING YOUR DATA to understand what it seems to say
- TESTING THE TRUTH of what you think you've seen
- GAINING INSIGHT through cycles of seeing and testing

ViSta is your stats video game. ViSta has tools for playing with your data ... tools which are graphical, dynamic, and interactive ... tools which encourage you to have fun with your data ... and ...

- When you are having fun, your playful and relaxed state augments your ability to understand your data ... increases your insight about your data.

So --- Have fun and learn lots!

ViSta has evolved in a special way over the years. We haven't simply added new features for the sake of adding new features. Rather, we have evolved ViSta with one goal always foremost: Maximize the quality of the user's experience. With this guiding principle, we are pleased that ViSta 7 is faster, smoother, and easier to use than ever before. It is also more flexible, expandable and capable.

- **FAST and SMOOTH:** ViSta 7 starts quickly, loads data quickly, and processes it quickly. Since it is faster, it can analyze larger data. ViSta is also smoother. The visualizations and desktop work more smoothly, bringing you closer to your data, presenting a data analysis environment that is a more satisfying and relaxing, and bringing you a richer visual experience for a deeper understanding of your data.
- **EASY TO USE:** ViSta 7 has a simplified user interface that has only four types of windows: The DeskTop, SpreadPlot, DataSheet and Report windows. The DeskTop and SpreadPlot windows feature panes that correspond to what previously were many separate windows. When you manipulate the DeskTop or a SpreadPlot, you manipulate all of the panes, meaning there is only one window to manage rather than four or more. DataSheets and Reports continue to be straight-forward to use.
- **FLEXIBLE:** ViSta 7 has PLUGINS and ADDONS that let you mold the data analysis methods and environment to fit your needs. PLUGINS add analysis methods to ViSta's core of basic methods, letting you tailor the selection of data analysis methods to those that are useful for you. ADDONS enhance the data analysis environment, letting you add new features as they become available. This permits ViSta's core system, and its unique selection of data visualization, exploration and description features, to remain stable and mature.
- **EXPANDABLE:** ViSta 7 continues to be an open system. Developers can write new plugins that introduce new data analysis capabilities, and new addons that add new features to the data analysis environment. System programmers can extend the overall capabilities of the system.

**Are you interested in developing new features for ViSta?** Just type (devel-mode) in the listener and welcome to the ViSta Development Team! Thanks for your interest in contributing to ViSta's development. Developers like you keep ViSta at the cutting edge of statistical software. I really appreciate the time and effort you and the other developers are taking to make ViSta the best freely available statistical visualization and analysis system.

When you type (devel-mode) in the listener, your developer's tools have been loaded, and ViSta's user environment has been changed into ViSta's developer's environment. The changes include an additional menu on the menubar: The Develop menu contains the tools for developing new applications for ViSta. Also, a "Toggle Devel Mode" item has been added at the end of the Options menu which toggles you in and out of developer's mode.

ViSta consists of a core engine plus plugins and addons. This design lets ViSta be both stable and expandable: The core is stable, while the plugins and addons provide the path to growth. This architecture also provides for an obvious organization of ViSta developers into Application developers and System developers. Applications

developers develop new plugins and addons, whereas system developers can also enhance ViSta's core engine.

**Application Development.** All of the code and tools that you need to be a ViSta Application Developer are already downloaded onto your machine. You can proceed to develop your plugin or addon application without coordinating your efforts with those of other application developers.

However, please check out [www.visualstats.org/developer](http://www.visualstats.org/developer) for information on what other folks are doing, and to leave information about what you are doing. This way, duplication of effort can be avoided, and the user and developer community can be kept up-to-date. If you need professional software development support, please contact [forrest@visualstats.org](mailto:forrest@visualstats.org) about our professional development and programming services.

If you wish, you can submit your application to us for distribution by [visualstats.org](http://visualstats.org). You are, of course, free to distribute your app independently from [visualstats.org](http://visualstats.org). Submitting an application to us is much like submitting a paper for publication. When your app is ready, just email the installation module to [forrest@visualstats.org](mailto:forrest@visualstats.org)

The installation module should include code, data, examples and documentation. The ViSta Editorial Board will review and make a recommendation concerning distribution. If it is approved for distribution it will be included on the [visualstats.org](http://visualstats.org) website, and links will be made so that it can be downloaded.

**System Development.** ViSta System Developers have access to portions of the source code, and can make changes to the system. But this isn't a free-for-all. Rather, because of the critical nature of systems development, and the importance of the core engine to the entire system, the system development effort is a coordinated effort. The effort is coordinated through the use of CVS, a version control system. CVS permits individuals who are part of a widely distributed development effort to work independently and simultaneously on a common set of code. The code is on your machine, where your CVS client coordinates your development with that of other developers, all the while permitting you to work independently from other developers. When you have completed your changes, the central CVS server will review all code changes to detect changes which conflict with those made by other system developers. You will then need to resolve these conflicts before the changes are accepted.

Contact Forrest Young if you wish to download ViSta from the ViSta CVS server at the University of North Carolina at Chapel Hill.

## Who has developed ViSta?

**Forrest W. Young:** ViSta has been developed by Forrest W. Young, Professor Emeritus of Quantitative Psychology at the University of North Carolina at Chapel Hill, located in Chapel Hill, NC, USA. He has worked on this project for more than a decade, and has received considerable help for his students and colleagues. Forrest is the creator and designer of ViSta, both in terms of its look and feel, and in terms of its internal software architecture. He has also implemented the design and has written much of the documentation.

Prof. Young received his PhD in Psychometrics from the University of Southern California in 1967. He has been on the faculty of UNC-CH ever since. His teaching interests focus on "Seeing what your data seem to say". This visually intuitive approach to statistics helps to clarify the meaning of data. His courses, ranging from his introductory undergraduate course on Psychological Statistics, to his advanced graduate courses on Data Analysis, Visualization and Exploration, reflect this focus.

To make the process of understanding data visually intuitive, the burden is moved from the person to the computer. You don't need to make an intensive effort to understand your data: Rather, your computer makes intensive calculations so that the data can be shown in a visually comprehensible way.

This approach to the role of computers is based on the intelligence augmentation (IA) philosophy of Computer Science: Your computer is a device which should augment your intelligence. It is also based on a Cognitive Science theory for the construction of an environment for data analysis.

Prof Young and his students, over the course of a 10-year research and development project, have created ViSta, a visual statistics system instantiating Prof. Young's theories concerning visual environments for statistical analysis.

ViSta is a freely available system that is being used for teaching introductory and multivariate statistics, for data analysis by statistically inexperienced researchers as well as by those who are more advanced, and for advanced

research and development in graphical and computational statistics.

ViSta is based not only on Prof. Young's theory-based approach to data analysis, but also on his 30-year career in computational and graphical statistics. Prof. Young's early research interests focused on Multidimensional and Nonlinear Multivariate Data Analysis (for which he was elected the President of the Psychometric Society, and received the American Market Research Association's O'Dell award, both in 1981). Via these research interests, Prof. Young became involved in software development early in his career.

Prof. Young has served as a professional consultant on statistical system interface design with SAS Institute, Statistical Sciences (the S-Plus system), and BMDP Inc. He has written or designed data analysis modules for the SAS, SPSS and IMSL systems. He is a member of the American Statistical Association's sections on Computational and Graphical Statistics.

**Pedro M. Valero-Mora**, Profesor Titular at the Universitat de Valencia (Spain), received his PhD in Psychology from this same university in 1996. He has worked on the department of Methodology of the Behavioural Sciences of the Universitat of València from 1990, teaching Introductory and Multivariate Statistics, Data Processing and Computers.

Pedro Valero's research interests have focused on two trends that have converged along the last years. On the one hand, he has worked on Statistics, mainly on the topic of graphics and statistical visualization. On the other hand, he has a big interest in the discipline of Human Computer Interaction, which is concerned with the design of computer interfaces that make them more useful, friendly and rewarding for users. The mixing of these interests with his teaching has drawn his attention to the development of innovative computer interfaces for statistical analysis and visualization.

To attain this objective, his programming efforts have been guided by the principles of direct manipulation, instant output and graphical visualization. The assumption is that the users should be able to manipulate directly visual representations of statistical techniques that respond immediately to their actions so they can understand more easily the consequences of the different choices under their control. As a result, the users can feel more confidence experimenting with the system so they can obtain a deeper understanding of the statistical concepts underlying the software.

The development of this type of software has been possible by the Lisp-Stat language and the ViSta system. These tools have made possible programming methods for Missing Data Imputation, visually controlled transformations and log-linear analysis, as well as, jointly with Rubén Ledesma from the University of Mar del Plata, an adaptation of a homogeneity analysis module. Other contributions to ViSta have consisted mainly on bug corrections, testing, documentation writing and translating, etc. These contributions have found its way into papers in academic journals focused on graphics and computers in statistics as well as in Human-Computer Interaction debate forums.

**J. Gabriel Molina** is Associate Professor in the Department of Research Methods at the University of Valencia (Spain). He received his PhD in Psychology from the University of Valencia in 1997 with a dissertation on computerized item banking for the development of adaptive tests. He is also member of the Traffic Research Institute at the University of Valencia.

Prof. Molina research interests focus on Data Processing and Analysis, Computerized Tests, and Traffic Psychology. In all these fields, he has worked in the development of computer methods oriented to enhance data management and analysis from a psychological basis, that is, computer tools that are useful for psychologists and social researchers and, at the same time, that take into account the developments in Psychology related to how to build more efficient programs for the final users.

## Acknowledgements

Over the last 10 years, ViSta has evolved from a testbed for the design of a highly interactive, very dynamic statistical visualization system, to just such a system. As the creator, designer and primary implementer of ViSta, I am deeply indebted to many, many people who have contributed in one way or another over the years. Without them, ViSta would not exist.

**Patricia Young:** My deepest thanks and appreciation go to my wife Patricia Young. During the last decade she has stood watch while I wandered in the developer's cave, evolving ViSta from what it was on the screen to what

it was in my mind's eye. Patty's patience, understanding, forbearance, support, and steady love, have been crucial to the realization of my dreams. Her artistic talent, abilities and spirit have shaped those dreams, and have had a deep impact on ViSta's design.

**Norman Cliff:** I deeply appreciate Norman Cliff's influences on me. As my mentor throughout graduate school, and as the chairman of my dissertation committee, Norm had a clear and major influence on my career's beginnings. But of even more importance, Norm's appreciation of the simple approach, his gentle strength, quiet certitude, civility and his strong sense of the ethical, have had a continuing impact throughout my entire career. Thanks Norm. You have been a great role model.

**Pedro Valero-Mora:** Mucho Gracias to Pedro Valero-Mora for his time, energy, enthusiasm and, last but not least, his code. Pedro wrote the code for missing data features, including visual transformation and imputation methods, as well as the visually-based Box-Cox and Folded Power transformations. His newest contribution, the log-linear plugin, appears in this release for the first time. Thanks for everything you've done, Pedro. I expect exciting new developments to come.

**Dominic Moore:** I great appreciate Dominic' s friendship, and support. His encouragement, criticisms, cheers, enthusiasm, pep-talks, humor, luniness, food, music and incredible speakers are food for my soul.

**Luke Tierney** Thanks and much appreciation to Luke Tierney for creating and providing the Xlisp-Plus statistical development system, which is the foundation on which ViSta is built. And thanks to the many folks who have built the Xlisp-Plus system on which XLisp-Stat was itself built: Open Software at its best.

**Sandy Weisberg:** Sandy's timely and insightful suggestions on getting ViSta out the door have helped, though perhaps not as much as everyone hoped! No-one else understands XlispStat the way we do, Sandy.

**David Lubinsky** and **John B. Smith** made very fundamental contributions to WorkMap architecture. Without their seminal input there'd be no ViSta.

**Richard Faldowski** and **Carla Bann** contributed fundamentally to SpreadPlot architecture, and spent many intense months implementing their ideas. They also invented several specific spreadplots and developed several analysis modules.

**Louis Le Guelte:** "Merci Beaucoup" to Louis Le Guelte for the French translation.

**Maria R. Rodrigo:** "Mucho Gracias" to Maria R. Rodrigo, Gabriel Molina, and Pedro Valero for the Spanish translation.

**Bibb Latane** of Social Science Conferences, funded development of the Excel-ViSta connection, of multipane windows, and the developer's conference.

**Angell Beza**, Associate Director of UNC's Odum Institute for Research in the Social Sciences, was instrumental in obtaining the SSCI funds.

**Jose Sandoval**, also at UNC's Odum Institute, developed the first public courses for teaching about using ViSta.

**Doug Kent**, funded by UNC's Office of InformationTechnology, implemented Version 5's MS-Windows features. Jenny Williams of UNC's Office of Information Technology, was my NT-Guru, especially on the network installation.

**Ernest Kwan, Rich Faldowski, Carla Bann, David Flora, Lee Bee Leng, Mary McFarland and Chris Weisen:** Mosaic Plots and Bar Charts are based on an algorithm by Ernest Kwan. Frequency Polygons and Histograms are based on algorithms by Jan de Leeuw and Jason Bond. Many of the Model Objects were written by Carla Bann, Rich Faldowski, David Flora, Ernest Kwan, Lee Bee Leng, Mary McFarland and Chris Weisen.

**Erann Gat:** The C-to-Lisp Parser used in ViVa was written and copyrighted by Erann Gat, who reserves all rights. Used under the terms of the GNU General Public License, the developer's menu derives from **Kjetil Halvorsen**, the Symbol Editor Dialog was implemented by **Frederic Udina**, and the bitmap editor by **Fabian Camacho**. Code for earlier versions was modified for Unix by **Anthony J. Rossini, Charles Kurak, Albrecht Gebhardt** and **Andrew V. Klein**

**My undergraduate students:** Quality assurance was provided, unwittingly, by my undergraduate students, who

suffered through "creeping-featuritis" and old documentation.

Thanks to all of you for helping me make ViSta what it is today. Without you, it would not exist.

Forrest

## PART I: GETTING STARTED

If you have not yet installed ViSta, then follow the simple installation steps given in the next section. Then, once you have installed ViSta, you can get started using it. The best way to get started using ViSta depends on whether you are a new or returning ViSta user, and whether you want to use ViSta with Excel.

- **New User?** If so, read the “Using the DeskTop” and “Using the Data Menu” sections of this guide (or the corresponding topics of ViSta’s help. They will familiarize you with ViSta and its capabilities.
- **Experienced User?** Go ahead! Jump right in!
- **Excel User?** Then the material about “Using Excel” is for you. You should also read the “Using the DeskTop” and “Using the Data Menu” if you are not familiar with ViSta.

### Installing ViSta

#### Single-user (stand-alone, no network) Installation:

A] DOWNLOAD ViSta

B] DOUBLE -CLICK THE DOWNLOADED MODULE.

C] INSTALLSHIELD RUNS to start the installation. Follow InstallShield's instructions. Usually you will take default choices everywhere.

D] VISTA RUNS to complete the installation. You are immediately presented with the NETWORK INSTALL choice. The first dialog box says to choose between

SINGLE USER, NO NETWORK

ALL OTHER SITUATIONS

If you choose SINGLE USER, NO NETWORK, ViSta installs and starts. The single-user installation is complete.

#### Multiple networked users:

If you are installing ViSta to be used by multiple users across the network, then choose ALL OTHER SITUATIONS and follow these steps:

1. Choose ALL OTHER SITUATIONS for network install (as explained in the help).
  2. Read the explanation and click NEXT.
  3. Choose THE NETWORK for network install.
  4. Make the appropriate choice (note that choices 3 and 4 have a long explanatory note about how to actually make these choices occur)
  5. Enter the User's Default Directory (default recommended).
  6. Click REVISE if you wish to make changes. Click INSTALL when everything is the way you want it.
  7. Click ACCEPT on the summary dialog. Rarely do you want to change values here.
- ViSta then completes the installation in the next few seconds.

For use on a network, you may need to set ViSta’s Initial window size and location. There are two alternate approaches to setting window location/size. The first approach is simplest.

1. For NT when ViSta is being setup to serve clients without special privileges, the steps to create default window location/size are:
  - Logon to NT as administrator.
  - Start Vista with command line option -useHKLM.
  - Set Vista main window the way users will see it on startup.
- 2) For NT when ViSta is being setup to serve clients without privileges, Here are the steps to take to create a default window location/size, font type/size, directory structure and initial ViSta startup preferences (It really doesn't matter if you do steps 6-8, but if you dont then at exit-time, unprivileged users Vista will quietly attempt and fail to write the windowPos to HKLM).
  - 1) Login to NT as administrator
  - 2) Start Vista
  - 3) Set useHKLM ini entry to 'yes' to enable writing window info
  - 4) Set Vista main window the way users will see it on startup
  - 5) Close Vista. Vista writes windowPos value to HKLM where users will pick it up when they run Vista.

- 6) Restart Vista.
- 7) Clear or delete the useHKLM ini entry.
- 8) Close ViSta

For your information, here is what ViSta does with HKLM and HCU on startup and shutdown:

#### Reading on startup:

Entry in HKLM	Entry in HCU	Behavior
yes	No	use entries in HKLM
yes	yes	use entries in HKLM
no	yes	use entries in HCU
no	no	provide default window size

#### Writing on shutdown:

useHLKM	Behavior
yes	write entries to HKLM
no	write entries to HCU

## Running ViSta

You can run ViSta by double-clicking the ViSta icon on the desktop or in the Start menu, by using one of the other icons in the Start menu, by double-clicking a ViSta datafile or applet file, or by downloading a ViSta applet or datafile from the net.

When you run ViSta, you will (by default) see a splash screen containing ViSta's logo. This is quickly replaced by ViSta's main window, which is called the ViSta DeskTop window, or DeskTop for short. It is titled "ViSta – The Visual Statistics System" (initially, the title includes a copyright). Note that if you wish you can hide one or both of these windows at startup.

In addition to the splash screen and DeskTop windows, ViSta has DataSheet windows, View (also called SpreadPlot) and Plot windows (a view window is a window containing several Plots), Report windows, Help windows, and the XlispStat window. You don't normally see these windows at startup, though you can if you wish to.

In this section we focus on using the DeskTop window. In later sections of the chapter we focus on the other windows.

## Using the Desktop

ViSta's DeskTop window has three window-panes and a menubar. By default, you see the menubar and all three window-panes at the same time. You can use the various MAXIMIZE items of the DESKTOP menu to focus exclusively on one specific pane. When you use a MAXIMIZE item it is replaced with a RESTORE DESKTOP item. The RESTORE DESKTOP item restores your view of all window-panes. The DEFAULT DESKTOP item restores your view to the original default view. The window-panes and menubar are described below.

### Using The Menu Bar

The menus on ViSta's menubar fall into three groups: The System menus, the Statistical menus and the Support menus. ViSta DeskTop also has popup menus which are accessed by right-clicks.

#### 1. The System Menus: FILE and EDIT

You use the FILE menu to get data into ViSta, to print results, and to output information from ViSta. For inputting data into ViSta, the File menu has items to create NEW DATA; to OPEN DATA for analysis by ViSta; to SIMULATE DATA according to statistical models; and to IMPORT DATA from text files. For printing results, the File menu has items to print files, the contents of a window, and the contents of a pane of a graphics window. For outputting information, the File menu has items for exporting data and for saving data and models.

The EDIT menu is not directly focused on data analysis, but provides the usual copying, pasting, etc.

## 2. The Statistical Menus: DATA, PLOTS, VIEWS, TRANSFORM, ANALYZE, and MODEL

The statistical menus provide you access to ViSta's statistical visualization and analysis capabilities. These six menus are the six major steps that you take to gain understanding of your data, and their arrangement order reflects the order in which these steps occurs when looking at your data. A typical data analysis is a cycle repeating these six steps, with the cycle spiraling in on a deeper understanding of the data.

**DATA:** Use the DATA menu "to see what the data seem to say", to manipulate the data, and to get quick summary information, both visual and numeric, about them.

**PLOTS:** Use the PLOTS menu to obtain simple visual information about the data.

**VIEWS:** Use the VIEWS menu to get more advanced and complete visual information about the data.

**TRANSFORM:** Use the TRANSFORM menu to transform the data (whether this is necessary depends on "what you see" when looking at your data).

**ANALYZE:** Use the ANALYZE menu to create a model of what your "data seem to say".

**MODEL:** Use the MODEL menu to look at the model resulting from the analysis (again, "to see what the data seem to say", but this time we look at the view the model provides).

## 3. The Support Menus: OPTIONS, HELP, DESKTOP and WINDOW

The OPTIONS, HELP, DESKTOP and WINDOW provide support facilities that effect how ViSta interacts with you. Briefly, the OPTIONS menu allows you to alter ViSta's data analysis environment; the HELP menu provides help about using ViSta; the DESKTOP menu allows you to alter the way the desktop window works with you; and the WINDOW menu provide access to ViSta's other major windows. Each of these menus has help information which you can access through the HELP menu.

## Using the WorkMap

The DeskTop's upper-left window-pane is called the WorkMap. The WorkMap maps the steps you take during a data analysis session. The map is constructed and displayed as your analysis progresses. At first, before you start analyzing your data, there is no map. As you step through your data analysis, icons representing data analysis steps are added to the WorkMap. The icons are usually connected to previous icons by lines that show the flow of your data analysis steps.

The WorkMap is the heart of ViSta's data analysis and visualization environment. It provides an up-to-date picture of your data analysis session, and, with the help of the Toolbar, is the control center where you interact with ViSta to understand your data. The WorkMap is where you create analyses and visualizations of your data.

The WorkMap records the steps you take during a data analysis session. The map is more than just an aid for remembering what you've done: You can interact with the map to re-visit previous steps in your analysis, and, when you wish, to analyze your data in new ways.

The WorkMap grows as your analysis progresses. At first, before you start analyzing your data, there is no map. As you step through your data analysis, icons representing the statistical objects created by your data analysis steps are added to the WorkMap. The icons are usually connected to previous icons by lines that show the flow of your data analysis steps.

**LEFT CLICK** an icon to make it the focus of the analysis. Some icons represent data, some other aspects of your data analysis. A left click on a data icon shows the data in the DeskTop's DataSheet, and the observations and variables in the Selector.

**RIGHT CLICK** an icon to display a menu of actions you can take with the icon.

**DOUBLE LEFT CLICK** an icon to reveal its contents. For data icons, you will see a large datasheet (click the RESTORE button to restore the desktop). For other icons, you will a view of the contents that is appropriate to the

icon.

DRAG an icon to a new location by placing your cursor on the icon, pressing your mouse button down, and dragging to a new location while holding the button down.

CTRL-DRAG an icon -tree (an icon and all the icons attached below it) by holding down the CTRL key before dragging.

## Using The Toolbar

The ToolBar contains buttons which provide instant access to ViSta's analyses, visualizations and reports. The left three buttons provide access to Help and to a statistical (data or model) object's report and visualization. The right-hand group of buttons provide quick access to various ANALYSIS possibilities. These buttons correspond to items of the ANALYSIS menus.

ViSta can decide to a limited extent what analyses are appropriate for the kind of data you are using. This decision is reflected in the changing appearance of the buttons (and of the corresponding menu items in the ANALYSIS menu). An active (colorful) button is judged potentially appropriate. An inactive (gray) button is judged definitely inappropriate. The ANALYSIS menu items change accordingly. Clicking an activated (not gray) tool button carries out the action of the button. For the left three buttons, these actions are Help, Reports and Visualizations. For the rest of the buttons, these actions are specific analyses. Right-clicking the toolbar gives you a menu which lets you redefine the number of buttons and the button actions.

## Using The Selector

The upper-right pane of the DeskTop window is called the SELECTOR.. The Selector display lists of the observations and variables in the currently active data object (when there is one). You can select items in these lists to form a subset of ACTIVE observations and variables.

ACTIVE variables and observations are those which are highlighted, or if none are highlighted, those which are listed.

Only the ACTIVE variables are used in ViSta's analyses and visualizations.

You can create a new data object containing only the ACTIVE observations and variables with the DATA menu's CREATE DATA menu item.

You make a SELECTOR item ACTIVE by clicking it. You can select several items by dragging your cursor over them. You can add items to a selection by CTRL-clicking, or CTRL-dragging.

You form a selection with the buttons at the top of the SELECTOR. These buttons let you put data items IN the selection or take them OUT of the selection. You can also DROP a selection or RESET all data items to the selection.

## Using The Listener

The lower pane of the DeskTop window is called the Listener. The listener listens to the state of ViSta, displays messages about the system, and lets you type commands in ViSta's underlying Lisp and ViVa languages..

There are two Listener Windows: The DeskTop Listener, which is in the main desktop window, and the XLispStat Listener, which is in the XLisp-Stat window. You need to use the WINDOW menu's XLISPSTAT WINDOW item to see the XLispStat window. Only one of these is active at a time. Both work identically.

Several items of the WINDOW menu control aspects of the DeskTop Listener Window. The MAXIMIZE LISTENER menu item lets you maximize its size, while HIDE LISTENER hides it from view. The RESTORE LAYOUT item changes the desktop back to the basic layout. The DESKTOP LAYOUT item lets you change the

number of lines shown in the DeskTop Listener.

To save the information that you see in the Listener, use the OPTIONS Menu's RECORD LISTENER item. This item is a toggle which turns recording on and off. All information that appears in the window while the item is checkmarked (e.g., AFTER you use the item the first time and before you use it again) will be recorded in the file you specify in the file dialog.

For more information about using the Listener for programming, see Luke Tierney's book on Lisp-Stat.

## Using A DataSheet

Datasheets display your data and let you edit your data. Datasheets can be used to create a brand new data object, to change the data in an already existing data object, or to add new data to an already existing data object.

Datasheets support the standard editing functions. Click on a cell to edit it. Type the desired information. Use the delete key to delete what you have typed. Use the cursor keys or the return, home, end or tab keys to move to another cell.

Datasheets are not spreadsheets: They do not let you enter mathematical formulas in the cells. To do this, use a spreadsheet such as Excel. When you have the desired spreadsheet you can transfer it to ViSta as a datasheet, taking advantage of ViSta's advanced and extensive visualization and analysis features. When you make the transfer from Excel to ViSta, Excel's numerical and textual information is transferred as is, as are the values resulting from the Excel's formulas. The formulas themselves are not transferred.

### CONTENTS OF THE DATASHEET:

The datasheet contains OBSERVATION LABELS, VARIABLE NAMES, VARIABLE TYPES and DATA, as follows:

**OBSERVATION LABELS:** The left column of cells contains observation labels. These may contain any information that you wish. It is recommended that each label be unique.

**VARIABLE NAMES:** The top row of cells contains variable names. These may contain any information that you wish. It is recommended that each variable name be unique.

**VARIABLE TYPES:** The second row of cells contains variable types. ViSta recognizes two variable types: Numeric and Category. Type N for numeric, C for Category. Any other information typed in these cells is ignored.

**DATA:** The remaining cells of the datasheet contain the data. What you can type depends on the variable type specified for the variable (in the second header row).

**CATEGORY** variables form strings from the characters typed in a cell of the datasheet. Most characters are acceptable, although some are ignored and some are interpreted as movement characters.

**NUMERIC** and **ORDINAL** variables assume that their cells contain numbers. For this reason you can only enter the ten digits and the characters + - . and , (the . and , can only be typed at "appropriate" places, where, I regret to say, "appropriate" is according to American rules of numeric notation).

**SCIENTIFIC NOTATION** may be used, with the scientific-notation precision-type character (d, l, f, or s) being typed at the "appropriate" place. While you can type d (double-float) l (long-float) f (single-float) or s (short-float), ViSta only uses double-float and long-float, with the conversion details depending on the machine you are using. You can determine the precision by typing the global variable MACHINE-EPSILON. The value of this variable is the smallest floating point number for which  $(1 + \text{MACHINE-EPSILON})$  is not equal to 1.

**MISSING VALUES.** The datasheet can contain missing values. Missing values are entered as --- (three minus signs typed without spaces between them) and are displayed in the datasheet as --- or as NIL. For numeric variables --- has the numeric value equal to MACHINE-EPSILON. For character variables is is equal to NIL.

### REFORMATING THE DATASHEET:

You expand the datasheet (add rows, columns or matrices) by clicking on special cells in the datasheet. To add a new observation (row) to multivariate data, click on the "New Obs" cell located below the left-hand column. To add a new variable (column) to the data, click on the "New Var" cell located to the right of the top row. You expand matrix data in a similar fashion. You can also use the EXPAND button (or menu item) to add several rows or columns simultaneously. You can control the width of columns and the number of decimal places shown by using the FORMAT button on the button bar, or appropriate items of the DESKTOP menu.

### SAVING THE DATA:

You may use the button bar's SAVE button, or the FILE menu's SAVE DATA menu item to save the data. When

you save the data, the current data object will be updated, unless it is attached to another icon. If so, the old data object will be left unchanged, and a new one will be created from the information in the datasheet.

## Topic 5: Using Excel

### GETTING STARTED - Using Excel - Excel and ViSta (II .5.1)

WE REGRET THAT THIS FEATURE ONLY WORKS FOR VERSIONS OF MS-WINDOWS USING THE ENGLISH LANGUAGE.

ViSta works with Excel: Each program can transfer data to the other program, translating it into the other program's native language. The two programs are very complimentary, each one enhancing the capabilities of the other.

**EXCEL USERS:** You can use ViSta to visualize and analyze your Excel data. Simply use Excel's ViSta menu to transfer your data to ViSta or to save your Excel Data as a ViSta datafile. Then you can use ViSta's highly interactive graphical environment to visually explore your data. Or, if you wish, you can use other items of Excel's ViSta menu to have ViSta automatically visualize and analyze your data.

**VISTA USERS:** You can use Excel to prepare data for analysis and visualization by Vista. You can also transfer your ViSta data to Excel. In both cases you can use Excel to enter and manipulate data. Thus, you can enter your data into an excel spreadsheet and use Excel's capabilities to prepare your data for use in ViSta. Then, when the data are ready, simply transfer them to ViSta, or save them as a ViSta datafile using items of Excel's ViSta menu.

**ALL USERS:** You can switch back and forth between ViSta and Excel at any time, and send data back and forth as often as you need to. You can use both programs interactively, giving you access to Excel's data preparation and manipulation capabilities, and to ViSta's data visualization and analysis capabilities.

Here is a summary of what you have to do. More detail is available in other help items.

#### USE VISTA TO RUN VISTA EXCEL

Excel and ViSta communicate with each other by a communication's link created by an Excel Macro. You must run ViSta first, and then use ViSta to run Excel to create the link. This can be done by ViSta's "Run Excel" item of the "Options" menu, or by using the "ViSta-Excel" item in the Start Menu's ViSta group.

#### FORMATTING EXCEL'S DATA FOR VISTA

You can analyze data contained in an Excel spreadsheet with ViSta if the data are formatted correctly. You can use whatever method Excel provides for getting data into an Excel datasheet. For example, you can enter your data into a new spreadsheet or you can retrieve a previously defined spreadsheet as you normally would. You will then probably need to edit the data to get them ready for ViSta. Here's what to do:

- a) The data must be in a rectangular array of contiguous cells.
- b) The top row of cells must contain variable NAMES. The NAMES can be anything you wish, but cannot contain single or double quote marks. NAMES are used by ViSta to identify columns of data.
- c) The second row must contain variable TYPES. The variable TYPE can only be NUMERIC or CATEGORY (upper or lower case, no quote marks).
- d) The left column must contain observation labels. Labels can be any information you desire. They are used by ViSta to identify the rows of data.
- e) The first cell in the NAMES row and the first cell in the LABELS row are ignored.

#### GETTING YOUR DATA INTO VISTA

There are two steps involved in getting your data into ViSta:

- a) **SELECT DATA:** You must select the portion of your spreadsheet that contains the data you wish to use with ViSta.
- b) **EXCEL'S VISTA MENU:** The second step is to use Excel's ViSta menu to transfer your data to ViSta and, optionally, have ViSta visualize or analyze them. Each item of the menu transfers your data to ViSta. Some items also perform "canned" analyses.

#### RE-PROGRAM THE CONNECTION

The ViSta-Excel connection doesn't do what you want? Then see the documentation on chaing Excel's ViSta menu.

**GETTING STARTED - Using Excel - From Excel to ViSta (II .5.2)**  
SENDING DATA FROM EXCEL TO VISTA

WE REGRET THAT THIS FEATURE ONLY WORKS FOR VERSIONS OF MS-WINDOWS USING THE ENGLISH LANGUAGE.

ViSta can visualize and analyse data in an Excel spreadsheet, but to do this you must run Excel and ViSta by using:

- 1) the VISTA-EXCEL item of Window's START menu
- 2) ViSta's RUN EXCEL item of its' OPTIONS MENU. If you run Excel directly it may not be able to communicate with ViSta.

To make this process work as well as possible, follow these steps:

- I: PREPARE EXCEL DATA FOR VISTA
- II: SAVE YOUR DATA
- III: EXIT EXCEL --- yes, it means what it says!
- IV: RUN VISTA.
- V: USE VISTA'S "RUN EXCEL" MENU ITEM TO RUN EXCEL
- VI: OPEN YOUR DATA
- VII: SELECT YOUR DATA.
- VIII: USE ITEMS of EXCEL'S VISTA MENU

**I: PREPARE EXCEL DATA FOR VISTA**

1: Prepare the datasheet as you normally would, but subject to the following restrictions:

- a) The data must be in a rectangular array of contiguous cells. There can be no empty cells. Missing values must be indicated by NIL or nil and nothing else (quotes and double quotes DO NOT WORK).
- b) The top row of cells must contain variable NAMES. The NAMES can be anything you wish, but cannot contain spaces, tabs, or any other kind of "white space", nor can they contain single or double quote marks. NAMES are used by ViSta to identify columns of data.
- c) The second row must contain variable TYPES. The variable TYPE can only be NUMERIC or CATEGORY (upper or lower case, no quote marks). The variable type may contain any kind of "white space", nor can they contain single or double quote marks.
- d) The left column must contain observation labels. Labels can be any information you desire. They are used by ViSta to identify the rows of data. The NAMES can be anything you wish, but cannot contain spaces, tabs, or any other kind of "white space", nor can they contain single or double quote marks. NAMES are used by ViSta to identify columns of data.
- c) The second row must contain variable TYPES. The variable TYPE can only be NUMERIC or CATEGORY (upper or lower case, no quote marks). The variable types cannot contain any kind of "white space", nor can they contain single or double quote marks.
- e) The first cell in the NAMES row and the first cell in the LABELS row are ignored but must contain a variable or value.

**II: YOU MUST SAVE YOUR DATA AND**

III: YOU MUST EXIT EXCEL before you can visualize the data. This counter-intuitive step is required because you cannot visualize your data unless the ViSta menu is installed on Excel's menubar, and only ViSta can install it.

**IV: RUN VISTA**

V: THEN USE VISTA TO RUN EXCEL.

You can do this in two ways:

a: Run ViSta. Then use ViSta's RUN EXCEL menu item to run Excel.

b: Use the ViSta-Excel Icon in the Program Files folder. This runs a small program which automatically runs ViSta and then uses ViSta to run Excel. In Excel you may see an empty datasheet. It contains the macro for communicating between Excel and ViSta. You may minimize it, but don't dismiss it.

VI: OPEN YOUR DATA. You now must re-open your data to get it back into Excel

VII: SELECT YOUR DATA. You must select the portion of your spreadsheet that contains the data you wish to use with ViSta.

VIII: USE THE VISTA MENU. Every item of the ViSta menu sends your data to ViSta, and then does the indicted actions, using the new data. So, if your data are large and the transfer is slow, it is best to use the menu once and then ask ViSta to save the data as a ViSta datafile. There may be a menu item to do this, right in Excel's ViSta menu. If so, we recommend using it and then using ViSta to do the analysis.

### **GETTING STARTED - Using Excel - From ViSta to Excel (II .5.3)**

ViSta can send data to Excel. However, the method for doing this has not yet been automated, so you must take the following steps:

EXPORT DATA.

Use the EXPORT DATA menu item to export your data. Include both types of optional information so that the variable names and types as well as the observation labels are exported.

RUN EXCEL

Use the RUN EXCEL menu item to run excel.

OPEN DATA

In Excel, use the OPEN item of the FILE menu to open the data that you just exported. Follow the wizard's instructions to import the data into Excel. The data are SPACE DELIMITED.

### **GETTING STARTED - Using Excel - Notes about Using Excel (II .5.4)**

ABOUT EXCEL SERVICE RELEASE 2: Apparently, the connection does not work unless you have Excel Service Release 2. Otherwise you get an error message that the LEFT function is missing from Excel. It should be there, and if you have this problem, please update Excel. If this doesn't work, let me know at [bugs@visualstats.org](mailto:bugs@visualstats.org)

ABOUT MACROS:

During the process of using ViSta with Excel, you may be warned that a spreadsheet contains a macro, or you may be told that you are being prevented from loading a spreadsheet because it contains a macro. You should not be worried, as the instructions for Excel to work with ViSta are contained in the macro. You may need (or wish) to change your security settings so that the macro can be loaded more easily. When all is right, the macro installs the ViSta menu on Excel's menubar.

ABOUT RUNNING VISTA FIRST:

We recommend running ViSta before running Excel because running them in the other order may unsynchronize the startup relationships of the two programs. You can be sure of running ViSta first in two ways:

- 1) Run ViSta, then use ViSta's OPTIONS -> RUN EXCEL menu item to run Excel.
- 2) Use the VISEXCEL icon on your desktop or in the start menu. It runs ViSta before Excel.

If you run Excel first, you may not be able to connect with ViSta. The connection will usually work, but occasionally it will not and one of these error messages may occur:

PLEASE TRY LATER:

ViSta is already running but is hidden.

Excel connects but ViSta remains invisible.

**DATA DID NOT GET THROUGH:**

Excel finds the wrong instance of ViSta, generates the error message, and runs the right instance of ViSta.

**TERMINATING VISTA**

When you quit Excel, ViSta may still be running, and it may be hidden so you don't see it. Then, if you try to run Excel again, you may not be able to connect with ViSta. So you should make sure you terminate the ViSta session as well. Look at the task bar. If you see a ViSta icon, click it. Then you should see ViSta and can close it.

**GETTING STARTED - Using Excel - Changing Excel's ViSta Menu (II .5.5)****CHANGING EXCEL'S VISTA MENU**

You can change the items in Excel's ViSta menu. You can remove the ones that are there, add new ones, etc. The actions of the menu items can be any ViSta Lisp script. Thus, you can have menu items that do any type of visualization/analysis that ViSta Lisp is capable of.

You take the following steps to customize the menu items in Excel's ViSta menu:

1. Prepare the Scripts: Write the scripts to be used by the Excel menu items. The scripts in the startup\excel directory are very simple, single action scripts. More complex scripts can be written using Lisp.
2. Save the Scripts: Each script must be saved as a file in ViSta's startup\excel directory. The filename must match the name in the excelmenus.txt file (see next step).
3. Change the Menus: Look at the existing excelmenus.txt file to see how the information in it connects the menus you see in Excel with the scripts in the startup/excel folder. Then edit the file, entering the new menu name followed by the lisp script associated with it.

## Topic 6: Using the Menus

### Using the Menus - The Menu Bar (II .6.1)

#### ABOUT VISTA'S MENUBAR MENUS:

The menus on ViSta's menubar fall into three groups:

- File menus: FILE and EDIT
- Analysis menus: DATA, PLOTS, VIEWS, TRANSFORM, ANALYZE and MODEL
- Support menus: OPTIONS, HELP, DESKTOP and WINDOW

The File Menus: FILE and EDIT menus.

- You use the FILE menu to get data into ViSta, to print results, and to output information from ViSta. For inputting data into ViSta, the File menu has items to create NEW DATA; to OPEN DATA for analysis by ViSta; to SIMULATE DATA according to statistical models; and to IMPORT DATA from text files. For printing results, the File menu has items to print files, the contents of a window, and the contents of a pane of a graphics window. For outputting information, the File menu has items for exporting data and for saving data and models.
- The EDIT menu is not directly focused on data analysis, but provides the usual copying, pasting, etc.

The Data Analysis and Visualization Menus

- The analysis menus provide you access to ViSta's statistical visualization and analysis capabilities. These six menus are the six major steps that you take to gain understanding of your data, and their arrangement order reflects the order in which these steps occur when looking at your data. A typical data analysis is a cycle repeating these six steps, with the cycle spiraling in on a deeper understanding of the data.
- DATA: Use the DATA menu "to see what the data seem to say", to manipulate the data, and to get quick summary information, both visual and numeric, about them.
- PLOTS: Use the PLOTS menu to obtain simple visual information about the data.
- VIEWS: Use the VIEWS menu to get more advanced and complete visual information about the data.
- TRANSFORM: Use the TRANSFORM menu to transform the data (whether this is necessary depends on "what you see" when looking at your data).
- ANALYZE: Use the ANALYZE menu to create a model of what your "data seem to say".
- MODEL: Use the MODEL menu to look at the model resulting from the analysis (again, "to see what the data seem to say", but this time we look at the view the model provides).

The Support Menus

- The OPTIONS, HELP, DESKTOP and WINDOW provide support facilities that effect how ViSta interacts with you. Briefly:
- OPTIONS menu allows you to alter ViSta's data analysis environment;
- HELP menu provides help about using ViSta;
- DESKTOP menu allows you to alter the way the desktop window works with you;
- WINDOW menu provide access to ViSta's other major windows.

Each of these menus has help information which you can access through the HELP menu.

### GETTING STARTED - Using the Menus - Pop-Up Menus (II .6.2)

You can pop-up a menu by right-clicking the desktop or one of the desktop's icons. These menus give you access to nearly all of the capabilities that can be accessed via the menubar. The menu you get depends on exactly what you have clicked on, as is summarized below.

#### RIGHT-CLICKING A DATA ICON

Icon Body: Data Menu  
 Icon Cap: About Menu  
 Icon Buttons:  
 Up Left: Report Menu

Up Right: Visualization Menu  
Lo Left: Transformations Menu  
Lo Right: Analysis Menu

#### RIGHT-CLICKING A DATASHEET ICON

Icon Cap: About Menu  
Icon Body DataSheet Menu

#### RIGHT-CLICKING A MODEL ICON

Main Body: Model Menu  
Cap: About Menu  
Icon Buttons:  
Left: Report Menu  
Right: Visualization Menu\

#### RIGHT-CLICKING THE DESKTOP

Anywhere Desktop menu

#### RIGHT-CLICKING THE TOOLBAR

Anywhere Change the buttons.

## Topic 7: Entering Data

### GETTING STARTED - Entering Data - Editing Data (II .7.1)

#### ABOUT EDITING YOUR DATA

The Edit Data menu item enables you to change your data. At the same time, the menu item safeguards both your original data and your changes. It does not, however, permanently save your changes. You must use the SAVE DATA AS menu item to do that.

While you edit your data you are never actually changing the original data. You also cannot accidentally destroy the original data by unknowingly replacing it with your changes. Nor can you accidentally analyze your incompletely changed data.

These features are all based on the following important aspect of ViSta: In ViSta, data can exist in two forms called a dataobject and a datasheet. These two forms are represented on the workmap in two distinct ways: By a dataobject icon and by a datasheet icon.

Data in dataobjects can be analyzed but cannot be edited, while data in datasheet objects can be edited but cannot be analyzed. This prevents data from being accidentally changed, and from being prematurely analyzed. To analyze the changes you have made, you must first use the SAVE DATA AS menu item (see below).

When you use the EDIT DATA menu item it seems as though ViSta opens your dataobject and displays it to you in a window as a datasheet ready for editing. However, to safeguard your data, when you use the Edit Data menu item, ViSta unobtrusively creates a datasheet object from your dataobject and opens the datasheet object, not the dataobject.

If you look at the workmap, you will see that a datasheet icon has appeared. This icon represents the unobtrusively made datasheet object. You are editing the data in the datasheet object, not in the dataobject. In this way, your original data are safeguarded in its dataobject, and you can always return to your unedited data by clicking on the data icon from which the datasheet was unobtrusively made.

## SAVING YOUR CHANGES

The SAVE DATA AS menu item both saves your data to your computer's file system as a DATAFILE and produces a new DATAOBJECT on the workmap. The new data object contains your edited data and is ready for analysis.

You can save your editing as often as you wish. Each time, you get a new datafile and a new dataobject, representing and saving your edited data at different stages of editing.

You can return to any previously saved edited version of your data by clicking on the appropriate dataobject or datasheet.

You can close a datasheet window and reopen it. The editing you have done is not lost when you close the window, even if the changes have not been saved.

## LOSING YOUR CHANGES

YOUR EDITING CHANGES ARE LOST if you do not use the SAVE DATA AS item before you exit ViSta.

## GETTING STARTED - Entering Data - ViSta's DataSheet (II .7.2)

### ABOUT VISTA'S DATASHEETS

Datasheets display your data and let you edit your data. Datasheets can be used to create a brand new data object, to change the data in an already existing data object, or to add new data to an already existing data object.

Datasheets support the standard editing functions. Click on a cell to edit it. Type the desired information. Use the delete key to delete what you have typed. Use the cursor keys or the return, home, end or tab keys to move to another cell.

Datasheets are not spreadsheets: They do not let you enter mathematical formulas in the cells. To do this, use a spreadsheet such as Excel. When you have the desired spreadsheet you can transfer it to ViSta as a datasheet, taking advantage of ViSta's advanced and extensive visualization and analysis features. When you make the transfer from Excel to ViSta, Excel's numerical and textual information is transferred as is, as are the values resulting from the Excel's formulas. The formulas themselves are not transferred.

### CONTENTS OF THE DATASHEET:

The datasheet contains OBSERVATION LABELS, VARIABLE NAMES, VARIABLE TYPES and DATA, as follows:

**OBSERVATION LABELS:** The left column of cells contains observation labels. These may contain any information that you wish. It is recommended that each label be unique.

**VARIABLE NAMES:** The top row of cells contains variable names. These may contain any information that you wish. It is recommended that each variable name be unique.

**VARIABLE TYPES:** The second row of cells contains variable types. ViSta recognizes two variable types: Numeric and Category. Type N for numeric, C for Category. Any other information typed in these cells is ignored.

**DATA:** The remaining cells of the datasheet contain the data. What you can type depends on the variable type specified for the variable (in the second header row).

**CATEGORY** variables form strings from the characters typed in a cell of the datasheet. Most characters are acceptable, although some are ignored and some are interpreted as movement characters.

**NUMERIC** and **ORDINAL** variables assume that their cells contain numbers. For this reason you can only enter the ten digits and the characters + - . and , (the . and , can only be typed at "appropriate" places, where, I regret to say, "appropriate" is according to American rules of numeric notation).

**SCIENTIFIC NOTATION** may be used, with the scientific-notation precision-type character (d, l, f, or s) being typed at the "appropriate" place. While you can type d (double-float) l (long-float) f (single-float) or s (short-float), ViSta only uses double-float and long-float, with the conversion details depending on the machine you are using.

You can determine the precision by typing the global variable MACHINE-EPSILON. The value of this variable is the smallest floating point number for which  $(1 + \text{MACHINE-EPSILON})$  is not equal to 1.

MISSING VALUES. The datasheet can contain missing values. Missing values are entered as --- (three minus signs typed without spaces between them) and are displayed in the datasheet as --- or as NIL. For numeric variables --- has the numeric value equal to MACHINE-EPSILON. For character variables is is equal to NIL.

#### REFORMATING THE DATASHEET:

You expand the datasheet (add rows, columns or matrices) by clicking on special cells in the datasheet. To add a new observation (row) to multivariate data, click on the "New Obs" cell located below the left-hand column. To add a new variable (column) to the data, click on the "New Var" cell located to the right of the top row. You expand matrix data in a similar fashion. You can also use the EXPAND button (or menu item) to add several rows or columns simultaneously. You can control the width of columns and the number of decimal places shown by using the FORMAT button on the button bar, or appropriate items of the DESKTOP menu.

#### SAVING THE DATA:

You may use the button bar's SAVE button, or the FILE menu's SAVE DATA menu item to save the data. When you save the data, the current data object will be updated, unless it is attached to another icon. If so, the old data object will be left unchanged, and a new one will be created from the information in the datasheet.

### **GETTING STARTED - Entering Data - Excel SpreadSheets (II .7.3)**

#### SENDING DATA FROM EXCEL TO VISTA

WE REGRET THAT THIS FEATURE ONLY WORKS FOR VERSIONS OF MS-WINDOWS USING THE ENGLISH LANGUAGE.

ViSta can visualize and analyse data in an Excel spreadsheet, but to do this you must run Excel and ViSta by using:

- 1) the VISTA-EXCEL item of Window's START menu
- 2) ViSta's RUN EXCEL item of its' OPTIONS MENU If you run Excel directly it may not be able to communicate with ViSta.

To make this process work as well as possible, follow these steps:

- I: PREPARE EXCEL DATA FOR VISTA
- II: SAVE YOUR DATA
- III: EXIT EXCEL --- yes, it means what it says!
- IV: RUN VISTA.
- V: USE VISTA'S "RUN EXCEL" MENU ITEM TO RUN EXCEL
- VI: OPEN YOUR DATA
- VII: SELECT YOUR DATA.
- VIII: USE ITEMS of EXCEL'S VISTA MENU

#### I: PREPARE EXCEL DATA FOR VISTA

1: Prepare the datasheet as you normally would, but subject to the following restrictions:

- a) The data must be in a rectangular array of contiguous cells. There can be no empty cells. Missing values must

be indicated by NIL or nil and nothing else (quotes and double quotes DO NOT WORK).

b) The top row of cells must contain variable NAMES. The NAMES can be anything you wish, but cannot contain spaces, tabs, or any other kind of "white space", nor can they contain single or double quote marks. NAMES are used by ViSta to identify columns of data.

c) The second row must contain variable TYPES. The variable TYPE can only be NUMERIC or CATEGORY (upper or lower case, no quote marks). The variable type any kind of "white space", nor can they contain single or double quote marks.

d) The left column must contain observation labels. Labels can be any information you desire. They are used by ViSta to identify the rows of data. The NAMES can be anything you wish, but cannot contain spaces, tabs, or any other kind of "white space", nor can they contain single or double quote marks. NAMES are used by ViSta to identify columns of data.

c) The second row must contain variable TYPES. The variable TYPE can only be NUMERIC or CATEGORY (upper or lower case, no quote marks). The variable types cannot contain any kind of "white space", nor can they contain single or double quote marks.

e) The first cell in the NAMES row and the first cell in the LABELS row are ignored but must contain a variable or value.

II: YOU MUST SAVE YOUR DATA AND

III: YOU MUST EXIT EXCEL before you can visualize the data. This counter-intuitive step is required because you cannot visualize your data unless the ViSta menu is installed on Excel's menubar, and only ViSta can install it.

IV: RUN VISTA

V: THEN USE VISTA TO RUN EXCEL.

You can do this in two ways:

a: Run ViSta. Then use ViSta's RUN EXCEL menu item to run Excel.

b: Use the ViSta-Excel Icon in the Program Files folder. This runs a small program which automatically runs ViSta and then uses ViSta to run Excel. In Excel you may see an empty datasheet. It contains the macro for communicating between Excel and ViSta. You may minimize it, but don't dismiss it.

VI: OPEN YOUR DATA. You now must re-open your data to get it back into Excel

VII: SELECT YOUR DATA. You must select the portion of your spreadsheet that contains the data you wish to use with ViSta.

VIII: USE THE VISTA MENU. Every item of the ViSta menu sends your data to ViSta, and then does the indicted actions, using the new data. So, if your data are large and the transfer is slow, it is best to use the menu once and then ask ViSta to save the data as a ViSta datafile. There may be a menu item to do this, right in Excel's ViSta menu. If so, we recommend using it and then using ViSta to do the analysis.

### **GETTING STARTED - Entering Data - SAS Datasets (II .7.4)**

The SAS2VSTA macro. Written by Michael Friendly

The SAS2VSTA macro generates a ViSta input file from a SAS dataset. It handles multivariate data and frequency classification data. Doesn't handle frequency table data.

#### **USAGE**

The SAS2VSTA macro is called with keyword parameters. The VAR= parameter is required. The arguments may be listed within parentheses in any order, separated by commas. For example:

```
%sas2vsta(data=new, var=Name Group X1 X2 X3, id=name,
  about=%str(A sample data set, converted to ViSta));
```

Observations may be selected by the WHERE= parameter. They appear in the output in their order in the input data set. Sort them first if you want some alternative order.

By default, output is written to a file of the same name as the SAS data set, with the extension '.lsp', in the current SAS directory.

## PARAMETERS

### DATA=

The name of the input data set [Default: DATA=\_LAST\_]

### DNAME=

Data name, the name of the data set for ViSta. [Default: DNAME=&DATA]

### VAR=

List of variable names to be output, a blank separated list. Variables appear in the output in the order listed, and in the :VARIABLES list in the case given.

### FREQ=

Name of a frequency variable, if any. Added as the last variable, and the :FREQ t flag is set. All other variables are treated as Category variables in this case.

### ID=

The name(s) of observation ID variable(s), which are used as row :LABELS. If two or more variables are given, their values are joined using the SEP= character for each observation.

### SEP=

Separator character just to join adjacent ID variables. If you want to use a character which the SAS macro processor treats as special, use, e.g., SEP=%str(,).

### ABBREV=

Maximum length of any ID variable when there are 2 or more listed in ID=. If ABBREV= is specified, each ID= variable is abbreviated to this length to construct a single observation label.

### WHERE=

WHERE clause to subset the observations in the output file. Use WHERE=%str(var=value) if value contains any funny characters.

### TITLE=

Generate a ViSta input file from a SAS dataset string for the data for ViSta [Default: TITLE=&DATA]

### ABOUT=

:ABOUT description; use %str() if it contains ','. If not specified, we use the data set label, if there is one. NB: you must use quotes, as in  
data foobar (label="My foobar data");  
for the label to be accessible to this macro.

### OUT=

Output destination, one of OUT=FILE, PRINT or STDOUT [Default: OUT=FILE]

### OUTFILE=

Name of output LSP file. [Default: &DATA.lsp]

### OUTDIR=

Name of output directory. [Default: Current SAS directory]

### LS=

Output linesize [Default: 80]

### Example

This example starts with data for a 2 x 2 x 6 contingency table on frequency for Admit x Gender x Dept for admissions to graduate school at Berkeley.

```

title 'Berkeley Admissions data';
proc format;
  value admit 1="Admit" 0="Reject"      ;
  value yn    1="+"    0="-"           ;
  value dept  1="A" 2="B" 3="C" 4="D" 5="E" 6="F";
data berkeley (label="Berkeley admissions data");
  do dept = 1 to 6;
    do gender = 'M', 'F';
      do admit = 1, 0;
        input freq @@;
        output;
      end; end; end;
      format dept dept. admit admit.;
/* Admit Rej Admit Rej */
cards;
  512 313  89  19
  353 207  17  8
  120 205 202 391
  138 279 131 244
  53  138  94 299
  22  351  24 317
;

```

To create a file, berkeley.lsp for ViSta, with the table variables ordered Gender, Dept, Admit, and the ID= variables separated by '-', use the following macro call:

```

%sas2vsta(data=berkeley, var=Gender Dept Admit, freq=Freq,
  id=Gender Dept Admit, sep=-,
  about=%str(Berkeley admissions data, from Bickel-et al:75) );

```

The output file, berkeley.lsp looks like this:

```

(DATA "berkeley"
:TITLE "berkeley"
:ABOUT "Berkeley admissions data, from Bickel-et al:75"
:FREQ t
:VARIABLES (QUOTE ( "Gender" "Dept" "Admit" "Freq"))
:TYPES (QUOTE ( "Category" "Category" "Category" "Numeric"))
:LABELS (QUOTE (
  "M-A-Admit" "M-A-Reject" "F-A-Admit" "F-A-Reject" "M-B-Admit" "M-B-Reject"
  "F-B-Admit" "F-B-Reject" "M-C-Admit" "M-C-Reject" "F-C-Admit" "F-C-Reject"
  "M-D-Admit" "M-D-Reject" "F-D-Admit" "F-D-Reject" "M-E-Admit" "M-E-Reject"
  "F-E-Admit" "F-E-Reject" "M-F-Admit" "M-F-Reject" "F-F-Admit" "F-F-Reject"
))
:DATA (QUOTE (
  "M" "A" "Admit" 512
  "M" "A" "Reject" 313
  "F" "A" "Admit" 89
  "F" "A" "Reject" 19
  "M" "B" "Admit" 353
  "M" "B" "Reject" 207
  "F" "B" "Admit" 17
  "F" "B" "Reject" 8
  "M" "C" "Admit" 120
  "M" "C" "Reject" 205
  "F" "C" "Admit" 202
  "F" "C" "Reject" 391
  "M" "D" "Admit" 138
  "M" "D" "Reject" 279
  "F" "D" "Admit" 131
  "F" "D" "Reject" 244

```

```

"M" "E" "Admit" 53
"M" "E" "Reject" 138
"F" "E" "Admit" 94
"F" "E" "Reject" 299
"M" "F" "Admit" 22
"M" "F" "Reject" 351
"F" "F" "Admit" 24
"F" "F" "Reject" 317
))
)

```

### GETTING STARTED - Entering Data - Importing Data (II .7.5)

IMPORT DATA opens a file and attempts to import data from it. If the information in the file is formatted as explained below, ViSta will load the data and display a data icon on the workmap.

For a file to be an importable datafile, it must be a multiple line file, where each line has the same number of pieces of information, called elements. If there are N variables, every line must have N+1 elements. An element can be either a number, a symbol, or a string. A symbol is an unquoted group of keyboard characters containing no white space, while a string is a double-quoted group of characters which may contain white space. Symbols are converted to upper case, strings retain original case. Missing data elements are represented by the symbol nil or NIL, but NOT by the string "nil" (regardless of case).

#### TYPES OF LINES IN THE FILE

There are three types of lines which may appear in an importable datafile. These lines are called the NAMES, TYPES and DATA lines. While there is some flexibility (see below), it is recommended that the DATA lines be preceded by a NAMES line and a TYPES line, each of which can only have elements which are character strings. Files which do not have these lines do not follow the recommended structure and may be more time consuming to process, or may not be importable.

**NAMES:** The NAMES line, which is first, begins with a character string which specifies the name of the exporter and of the dataset. This string is a two-part string: The first part identifies the exporter and the second part names the file. The parts are separated by a colon. Thus, two examples are "ViSta:Cars" and "Excel:Fish". The line then continues with N more character strings which are used to define variable names.

**TYPES:** The TYPES line, which is second, defines the dataset type (first) and the types of each variable (the rest). The NAMES and TYPES lines can only have elements which are character strings. The NAMES can be any string of characters. The DATA TYPES can be one of the following: "category", "univariate", "bivariate", "multivariate", "classification", "frequency", "freqclass", "crosstabs", "general", "missing", "symmetric" or "asymmetric". The VARIABLE TYPES can be either "category", "ordinal", or "numeric". Capitalization is ignored for all data and variable types.

**DATA:** Each of the remaining lines of the file forms a row of the data. The first element of each row is a "label" which is used to identify a data observation (in the case of "symmetric", or "asymmetric" data the label identifies a matrix of data). These first element of a row may be a string, symbol or number, but since labels are strings, the first element of a data line is converted to a string. The remaining elements of each line are the data. A data element must be a number if its' variable is "numeric", but can be a number, symbol or string if its' variable is a "category" variable.

#### THE MEANING OF NUMERIC ELEMENTS

Elements of numeric variables are numbers, and numbers carry different meanings in different situations. Depending on the data being imported, you may be asked to indicate what the numbers represent. The variety of

meanings can be grouped together into three kinds of meanings, as follows:

**QUANTITY:** A number can represent the quantity or amount of some characteristic.

**FREQUENCY:** A number can also represent the frequency of something, or the count of something.

**ASSOCIATION:** A number can also tell of the degree of association or correlation between two things, or the distance between a pair of things.

## TYPES OF IMPORTABLE FILES

**VISTA EXPORT FILES:** Files which follow the recommended structure given above are called ViSta Export Files.

**ANONYMOUS EXPORT FILE:** These files follow the recommended structure given above, but the first character string in the file is a one-part string. It is assumed that the string names the dataset, and does not identify the exporting program.

**UNTYPED EXPORT FILE:** These files have only a NAMES line preceding the DATA lines. No TYPES line appears. When a file has only one initial line with elements which are all character strings is assumed to be an untyped export file. For such a file it is assumed that the first column of values (i.e., the first value on each line) is a LABEL. The data and variable TYPES are determined from the nature of the data (a variable is "numeric" if it only contains numbers, "category" otherwise). The dataset type is determined from the variable types.

**PLAIN EXPORT FILE:** These files only have DATA lines. There is no NAMES or TYPES line. If the initial record of the file contains values other than strings (i.e., contains at least one number or symbol) the file is assumed to be a "Plain" export file. That is, it is assumed that the initial name and type records are absent, and that the first record contains data elements for the first row of data. The first element of each line is assumed to be a LABEL. Default values are used for the variable names, and the variable types are determined from the data (a variable is "numeric" if it only contains numbers, "category" otherwise). The dataset type is determined from the variable types. The datafile name is used as the name for the dataset.

## **GETTING STARTED - Entering Data - Simulating Data (II .7.6)**

**SIMULATE DATA** lets you generate new data by simulating the process of sampling from a population. It also includes a visualization that demonstrates the central limit theorem.

When you use **SIMULATE DATA**, you will see a dialog box that asks you to select a distribution. This corresponds to the theoretical population distribution from which samples will be drawn.

The dialog box also asks for sample size and number of samples, and whether or not you wish to visualize the sampling process. For certain distributions, an additional dialog box will ask for information needed to completely specify the sampling process.

The visualization shows the shape of the population and sample distributions as well as the distribution of sample means and standard deviations. If you ask for more than one sample, the last sample distribution is displayed along with all sample means and standard deviations. The visualization gives you the choice of generating additional samples and seeing the updated distributions of means and variances. Each time you click on the visualization's "New Sample" button ViSta generates "n" additional samples, where "n" is the number of samples specified in the dialog box.

The simulation process creates a new multivariate data object. It is represented by an icon on the workmap.

**GETTING STARTED - Entering Data - Writing DataCode (II .7.7)****WRITING DATACODE WITH A TEXT EDITOR**

You can enter your data by using a text editor to write code in the Lisp programming language underlying ViSta. A brief introduction is given here. You can also get a good idea of the format by looking at any file in one of the data folders. See the User's guide for more information.

Briefly, the minimum format for multivariate data is:

```
(data "name"
:variables '("Variable A" "Variable B" "Variable C")
:data '(
1 2 3
4 5 6
7 8 9
10 11 12
))
```

Additional features are described in the User's Guide.

The DATA function creates a new data object or reports the names of all data objects or the object identification of a specific object. It can be used in three different ways:

1) To see a list of all data objects, type:

```
(DATA)
```

2) To see the object identification of data object NAME, type:

```
(DATA NAME)
```

3) To create a new data object from information contained within the DATA statement, the minimum syntax requires you to type:

```
(DATA 'NAME :VARIABLES <VARLIST> :DATA <DATALIST> )
```

**GENERAL ARGUMENTS:**

**&OPTIONAL NAME &KEY DATA VARIABLES TYPES LABELS FREQ ABOUT**

Defines ViSta data object NAME. NAME, DATA and VARIABLES are required arguments.

NAME must be a string or a symbol. If a symbol it must be preceded by a single quote.

DATA must be followed by a list of numbers, strings or symbols (or mix of such). Symbols are converted to uppercase strings. The number of data elements must conform to the information in other arguments.

VARIABLES must be followed by a list of strings or a single quoted list of symbols defining variable names (and, indirectly, the number of variables).

TYPES, optional, must be a list of the strings \"numeric\", \"ordinal\" or \"category\" (case ignored), or a list of the corresponding symbols. These strings or symbols specify whether the variables are numeric, ordinal or categorical (all numeric by default). Note that the ordinal datatype is seldomly used.

LABELS, optional, must be a list of strings or symbols specifying observation names (\"Obs1\", \"Obs2\", etc., by default). Symbols are converted to uppercase strings.

FREQ specifies that the values of the numeric variables are frequencies.

ABOUT is an optional string of information about the data.

Given these arguments above you can specify the following types of data:

- 1) MULTIVARIATE data are data which are not one of the other data types given below. These data include univariate (one numeric or ordinal variable) and bivariate (two numeric or ordinal variables) data.
- 2) CATEGORY data have one or more CATEGORY variables and no NUMERIC or ORDINAL variables. The N category variables define an n-way classification.
- 3) CLASSIFICATION data have one NUMERIC variable and one or more CATEGORY variables. The N category variables define an n-way classification. The numeric variable specifies an observation for a given classification.
- 4) FREQUENCY CLASSIFICATION data are classification data whose numeric variable specifies frequencies as indicated by using `FREQ`. The N category variables define an n-way classification, with the numeric variable specifying the co-occurrence frequency of a specific combination of categories.

#### ARGUMENTS FOR FREQUENCY TABLE DATA:

ARGUMENTS: `&KEY ROW-LABEL`, `COLUMN-LABEL`, and `FREQ`

For `FREQUENCY TABLE` data, the `ROW-LABEL` and `COLUMN-LABEL` arguments must be used: These data have `NUMERIC` variables whose values specify frequencies as indicated by using `FREQ`. The data are a two-way cross tabulation of the co-occurrence frequency of the row and column entities. The `ROW-LABEL` and `COLUMN-LABEL` identify the data as two-way array (table) data, with the numeric variables in the data represent columns of a two-way table rather than variables of a multivariate dataset. `ROW-LABEL` and `COLUMN-LABEL` each have a string value which labels the rows or columns (ways) of the array. Observation labels are used to label the row-levels and variable names the column-levels.

#### ARGUMENTS FOR MATRIX DATA

ARGUMENTS: `&KEY MATRICES SHAPES`

These arguments are used to identify matrix data. `MATRICES`, required for matrix data only, must be a list of strings specifying matrix names (and, indirectly, the number of matrices). `SHAPES`, optional for matrix data only, must be a list of strings `"symmetric"` or `"asymmetric"` (case ignored), specifying the shape of each matrix (all are symmetric by default). Matrix arguments cannot be used with array data.

#### EFFICIENCY ARGUMENTS: `&KEY MISSING-VALUES`, `STRINGS`

- 1) If you know the data do or do not contain missing values, specifying `MISSING-VALUES` as `T` or `NIL` eliminates the time required to check for missing values.
- 2) Specify `STRINGS T` if you know that all category values are represented by strings (values inside double-quote marks) then the need to check for non-string category values is eliminated, greatly increasing efficiency, especially for large data.

The preceding arguments are all of those concerning the definition of data within ViSta. There are additional arguments which concern the creation of data by programming. These are described below.

#### ARGUMENTS FOR PROGRAMMING

ARGUMENTS: `&KEY PROGRAM`, `USE`

If you wish to write a data program, use these arguments:

- 1) `PROGRAM` (required) specifies that a program follows which computes N new variables. The program must return a list of N lists corresponding to the N variables in the `:VARIABLES` keyword. Causes the new variables to be bound.
- 2) `USE` (optional) specifies the data object whose variables are input to the program. This must be a data object with bound variables. If not, specify `:USE (BIND-VARIABLES DOB)`

For example, assume there is a dataobject named `pcaexmpl` containing variables `X` and `Y`. Then you can type: .

```
(data "PCAExmpl2"
 :use pcaexmpl
 :variables ("X" "Y" "A" "B" "C")
 :program
 (let* ((A (+ X Y))
 (B (+ (* 5 X) Y))
 (C (+ (* -2 X) Y)))
 (list x y a b c)))
```

Which creates a new data object containing variables a b and c as well as the original x and y.

Consider this example (which assumes the variables hmw1 through hmw4 already exist in the points dataobject):

```
(data "WeightedPoints"
  :use points
  :program
  (let* ((h1 (* hmwk1 5/10))
        (h2 (+ (* hmwk2a 6/10) (* hmwk2b 4/10)))
        (h3 (+ (* hmwk3abd 7/10) (* hmwk3c 3/10)))
        (h4 (* hmwk4 15/10))
        (hmwksum (+ h1 h2 h3 h4))
        (total (+ hmwksum midterm1 attend vis)))
    (list attend vis h1 h2 h3 h4 hmwksum total))
  :variables '("attend" "visual" "h1" "h2" "h3" "h4" "htot" "total"))
```

This program takes the variables hmw1 through hmw4 in the points dataobject and combines them together for a total homework score, plus adding other variables in SCORE to get a point total for a class.

---

Finally, there are arguments that effect the operation of the system. These should only be used by knowledgeable system developers. These arguments are:

&KEY CREATED CREATOR-OBJECT SUBORDINATE ICONIFY DATASHEET-ARGUMENTS NEW-DATA ALL-TYPES-IN-DATA-ARRAY

## PART III – UNDERSTANDING DATA

### TOPIC 8 MANAGING DATA

#### UNDERSTANDING DATA - Managing Data - Managing Data (III.8.1)

##### ABOUT VARIABLES AND DATA

An important aspect of statistical data analysis and visualization involves creating new variables, calculating new variables from existing ones, and modifying the values of existing variables. Such new variables become the basis of new data objects, which in turn can be used for data analysis and visualization.

This help file presents an overview of variables and data objects, including information about ViSta's lists of variable and data object names; about the difference between short names and long-names; about how you use the lists of names to refer to variable and data objects; about how you create new variable and data objects; and about variable types and data types.

##### ABOUT SYMBOLS

You manipulate variables by typing simple arithmetic or algebraic statements in the listener window. These statements involve symbols for the mathematical operations and symbols for the variables. The mathematical operation symbols are the familiar ones you have always used: + for addition, / for division, etc. The variable symbols are the variable names you supplied when you created the variables.

##### NAMES - SHORT AND LONG

Throughout ViSta, variables and data objects are referred to by names that you supply. However, there is always the possibility that a name might be repeated. If it is, the name does not unambiguously identify the variable or data object. To avoid this problem, ViSta constructs and uses "long-names" which are unambiguous. In contrast, the "short-names" are the names you supplied, which can be ambiguous.

##### DATA-OBJECT NAMES

Data objects have long names which consist of three parts.

- 1) The first part is the name you supply
- 2) The second part is a three character "extension"
- 3) The third part is the "version number".

The extension identifies the datatype of the data object. The version number is supplied by ViSta and is used to uniquely identify the data object. The number is 1 if the name is unique, 2 if there is already another data object with the same name, etc.

The long name has the syntax

`dataname.ext#n`

That is, the long name of a data object consists of the NAME you supplied, followed by a dot, then the EXTENSION, followed by # (number sign) and the VERSION. The extension identifies the datatype of the data object.

##### VARIABLE NAMES

The long-name of a variable is constructed from the name you gave the variable and the name of the data object that contains the variable. Specifically, a variable's long-name is the long name of the data object (including extension and version number) followed by a dot and the name you gave to the variable;

`dataname.ext#n.varname`

For example, if the data object MYDATA contains a variable named GPA, then the variable's long-name is MYDATA.GEN#1.GPA when there is no other data object named MYDATA, and when the data object's datatype is "general".

Variables do not have to be in data objects, but may be "free-standing" variables. When you first create and manipulate new variables they are free-standing, and are called "free" variables. Variables in data-objects are called "attached" variables. Free-standing variables have long-names which are the same as their given name since they are not in any data object. Note that there is the possibility of duplicate names.

There is always a "current" data object. This is the data object that is the current focus of data analysis and visualization. It is represented on the workmap by the highlighted data icon.

## REFERING TO VARIABLES

In order to manipulate a variable, you must be able to specify which variable you wish to manipulate. A variable or data object can be referred to by its name, including either its' short-name or long-name. If you use a short-name, the most recently created variable with that name will be the one referred to.

ViSta maintains information about the data and variable objects that have been defined during the session. The information can be referred to by various names. In addition to being able to refer to a data object by its' name (with or without the #n suffix), you can refer to the information by various symbols, all beginning with \$. The symbols include:

SYMBOL	INFORMATION
\$	the current data object
\$data	all data objects
\$vars	the variables in the current data
\$all-vars	all variables, free and attached
\$data-vars	all attached variables
\$free-vars	all free variables
\$NAME-vars	all variables in data object NAME (NAME must be a long data object name)

By typing one of these symbols in the listener you can see which data and variable objects have been defined. New variables are usually constructed from specific variables on these lists. In addition, new variables may involve calculations performed on variables on these lists.

NOTE: Due to fundamental differences in the nature of data objects whose datatype is MATRIX (extension .mat), these data objects not included in the lists above. Furthermore, the variables in data objects whose datatype is MATRIX are not included in the features discussed below.

## CREATING VARIABLES WITH VIVA AND VAR

ViSta provides two ways of creating new variable objects, known as ViVa and Var.

ViVa is ViSta's Variable language, and VAR is a function for creating variables. Each creates new variables which can be further manipulated with ViVa and Var, or can be made into data objects, using the DATASET function. The data object can then be analyzed and visualized by ViSta.

ViVa calculates variables using statements like ordinary arithmetic or algebraic statements. VAR calculates new variables using expressions in the Lisp language.

The variables created by ViVa and Var are called "free" variables because they are not contained in any data object.

ViVa and VAR are described in the CREATING VARIABLES help item.

## THE DATASET FUNCTION

The "free" variable objects created by ViVa and VAR must be placed in a data object so that they can be analyzed and visualized. The DATASET function creates a new data object from a group of variables. It is described in the ABOUT MAKING DATA help file.

## TYPES OF VARIABLE AND DATA OBJECTS

Variables have a property known as their "variable type", a property which determines what kinds of calculations can be done on their values and what kinds of statistics and visualizations are appropriate. The types recognized by ViSta are "numeric", "ordinal" and "category".

> Numeric variables are regular numbers, supporting ordinary arithmetic and the statistics and visualizations based on arithmetic.

> Ordinal variables are those whose values only specify order, not number. Such variables are seldomly used in ViSta.

> Category variables are those whose values only specify category membership. These variables are used in appropriate places in ViSta for determining the kinds of statistics and visualizations that can be computed.

By default, ViSta assumes variables are numeric. The VAR function's :TYPES keyword allows you to specify non-numeric variables. ViVa variables are always numeric.

## **UNDERSTANDING DATA - Managing Data - Variable and Data Types (III.8.2)**

### VARIABLE TYPES AND DATATYPES

Each variable in a ViSta data object has a "type", referred to as the "variable type" of the variable. Each data object also has a "type", called the "datatype".

Don't confuse these two "types". Variable types are fundamental. They are immutable characteristics of the variables. On the other hand, the datatype is an immutable characteristic of a dataset, which is derived from the variable types and from other information.

### VARIABLE TYPES

One of the basic pieces of information that ViSta needs for every aspect of what it does is the "type" of each variable in the data. The "variable type" must be ONLY one of the following three choices:

**CATEGORY** - The information provided by each observation of a variable only specifies the category of the observation, nothing more... not the order of the observations nor the values of the observations.

**ORDINAL** - The information provided by each observation of a variable specifies the order as well as the category of the observation, but not the numeric value.

**NUMERIC** - The information provided by each observation of a variable specifies the numeric value as well as the order and category of the observation.

Note that currently ViSta treats ORDINAL variables as though they are NUMERIC (with rare, but obvious, exceptions).

### DATATYPES

Each ViSta data object has a datatype. The datatype determines ViSta's default analyses and visualizations, and limits the choice of actions you can take to those that are likely to be reasonable.

Datatypes are meant to simplify the user's experience... they allow ViSta to make a more educated guess about

what should be done with the data, and provide an unobtrusive way of guiding the user by limiting choices.

The datatype depends on

- 1) whether the data contain missing values;
- 2) the mix of variable types; and
- 3) whether the values for numeric variables represent quantity, frequency or relatedness.

Note that the definition of "frequency" data has been retro-fitted to ViSta and is a bit of a kludge. Specifically data are frequencies if explicitly declared so in the datacode, or if all numeric variables are named "Freq".

There are 11 data types recognized by ViSta. They include:

A) MISSING - The data have one or more NIL elements

B) MATRIX - A datatype where the basic datum is a relation. The observations and variables refer to the same things, the data elements are relational, specifying the relation (correlation, distance, covariance) between pairs of the things.

C) Six datatypes where the basic datum is a quantity:

- CATEGORY - There are only category variables
- UNIVARIATE - There is one variable and it is non-frequency numeric
- BIVARIATE - There are two variables and they are both non-frequency numeric
- MULTIVARIATE - There are more than two variables. All are non-frequency numeric
- CLASSIFICATION - There is exactly 1 non-frequency numeric variable and there are 1 or more category variables
- GENERAL - When none of the above definitions is satisfied the datatype is "general"

D) Three datatypes where the basic datum is a frequency:

- FREQUENCY - All the variables are numeric frequency variables
- FREQCLASS - There is exactly 1 numeric frequency variable and there are 1 or more category variables
- CROSSTABS - There are 1 or more numeric frequency variables 1 and or more category variables

Formally, the datatypes are defined to be:

A) MISSING if the data contain one or more NIL elements.

B) MATRIX if matrices are present without NIL elements.

C) If neither of the above is true, then the datatype is defined according to the number of category and numeric variables in the data, and by whether the data are frequencies or not. Specifically:

Number of Category Variables			Number of Numeric Variables			
			0	1	2	>2
0	nil	error	univariate	bivariate	multivariate	
	t	error	freq	freq	freq	
>0	nil	category	class	general	general	
	t	category	freqclass	crosstabs	crosstabs	

NOTES:

- a) TABLE datatype is no longer used.
- b) ORDINAL variables are treated as NUMERIC.
- c) Defined on ALL variables, NOT active variables. This makes the datatype a characteristic of the data, not the active data. I am considering changing this in the future.

### **UNDERSTANDING DATA - Managing Data - Manipulating Variables (III.8.3)**

#### **MANIPULATING VARIABLES**

A variable is a specific type of information (either numeric or non-numeric) that is observed many times, and

which can vary from one observation to the next. Data is a collection of variables, and is the information that you wish to visualize and analyze with ViSta.

Before visualizing or analyzing your data, and often times during the course of a statistical investigation, it is necessary to manipulate your data. ViSta has three data manipulation tools:

1) ViVa, ViSta's Variable Language, creates and modifies variables using statements that are like algebraic equations. ViVa is easy to learn and use, but is limited in scope and can only be used in the listener. It is a special purpose language intended for manipulating ViSta variables.

2) VAR creates and modifies variables using expressions in the Lisp language. VAR is faster and much more general than ViVa and can be entered from the listener or from files. Although VAR is based on Lisp, which is much harder to learn than ViVa, you only need to learn a small subset of Lisp to manipulate variables. But, since Lisp is a general purpose computing language and ViVa is not, VAR can be much more powerful than ViVa.

3) DATASET creates data from a collection of variables. Note that ViVa and VAR create variables which can be further manipulated by ViVa and VAR. However, they cannot be visualized, analyzed or permanently saved until they are placed into a dataset with DATASET.

## DOLLAR FUNCTIONS

ViSta has several functions that tell you what variables are available for use by ViVa, VAR and DATASET. These functions all start with the \$ character, and so are called "dollar functions".

To find out what variables are available for creating new variables and forming new data objects, type

NAME	INFORMATION
\$vars	list of variables in the current data
\$all-vars	list of all variables
\$data-vars	list of all data object variables
\$free-vars	list of all free variables
\$NAME-vars	list of all variables in data object NAME

### ViVa Example:

ViVa uses algebraic equations to manipulate and create variables. The equations must be enclosed in brackets. For example, if you have measurements of temperature on the Fahrenheit scale used in the United States and you wish to convert them to the Celsius scale used by the rest of the world, you would type:

```
[celsius = (5 / 9) * (fahrenheit - 32)]
```

The variable on the left of the equal sign is created and assigned the value of the expression on the right.

### VAR Example:

You can also use VAR to manipulate existing variables and calculate new ones. VAR statements have the form:

```
(VAR variable formula)
```

This statement creates a new variable named "variable" whose value is the result of the calculations performed by "formula". Here's how you do the temperature scale conversion example with VAR:

```
(var celsius (* (/ 5 9) (- fahrenheit 32)))
```

Notice that functions (such as \* / and -) always precede their arguments.

### DATASET Example:

DATASET takes a collection of variables and makes a new dataobject from them. The dataobject persists throughout the session with ViSta and can be permanently saved to a file. The variables in the dataset can be

visualized and analyzed with ViSta.

For example, to put the two temperature scales in a datafile you type:

```
(dataset both-scales celsius fahrenheit)
```

The general form of the DATASET statement is

```
(DATASET NAME VAR1 VAR2 ...
  :TYPES ("numeric" "category" ...)
  :LABELS ("A" "B" "C" "D" ...)
)
```

The :TYPES and :LABELS keywords are used to specify variable types and observation names. These arguments are optional, although you must use :TYPES when you have categorical variables.

### **UNDERSTANDING DATA - Managing Data - Using ViVa (III.8.4)**

ViVa: ViSta's Interactive Variable Application

ViVa, ViSta Interactive Variable Application, is an algebra-like language which calculates values for new variables and makes the variables available for further manipulation.

ViVa consists of algebra-like expressions. If the expression involves assignment to a variable, the variable is assigned the value of the expression, and the variable is saved for the duration of the session with ViSta.

ViVa can be used in 3 ways. These ways are outlined briefly here. Examples of ViVa expressions are given below.

1) At the Listener, type a ViVa expression enclosed in brackets. For example, if x is a variable known to Lisp from, say, typing

```
> (var x '(1 2 3))
(1 2 3)
```

Then you can type the ViVa expression, in brackets, at the Lisp prompt:

```
> [y = 2 + 3*x]
(5 8 11)
```

The expression on the right side of the = sign is evaluated, the resulting value is assigned to y, and the value is returned. Thus:

```
> [x=4.5]
4.5
> [y = 2 + 3*x]
15.5
```

2) At the Listener, type a left bracket and a return. Lisp's > prompt is replaced by ViVa's ? prompt. Then you can type a series of ViVa expressions, each followed by a return. Each expression is evaluated when you type a return. Finally, type a right bracket to return to regular Lisp. The value of the last expression is returned. For example:

```
> [
? s = list(2,4,6)
(2 4 6)
? u = 3
3
? v = 2
2
? r = u + v*s
(7 11 15)
```

```
? ]
(7 11 15)
>
```

3) Enter a bracket enclosed ViVa statement in the middle of lisp. It is evaluated and returns its value to Lisp.  
Thus:

```
> (list 1 2 [sqrt(9)] 4)
(1 2 3.0 4)
```

#### LIMITATIONS:

- 1) ViVa can only be entered from the Listener, not from files. Thus, ViVa scripts or applets are not possible.
- 2) When using ViVa, variable names cannot contain embedded characters interpreted as mathematical operators (i.e., - + = / \* , etc). These signs are always interpreted as being a mathematical operator. In particular, the Lisp convention of variable or function names containing dashes, such as principal-components or normal-rand, is not allowed in ViVa. Underscores may be substituted for dashes (see next point).
- 3) Underscores may only be used as a substitute for dashes (see previous point).

#### UNDERSCORES:

Since dashes are very commonly used in the names Lisp functions, you may substitute underscores for dashes. Thus, you can enter principal\_components or normal\_rand. ViVa will translate appropriately.

#### VERBOSE:

Type (viva-verbose t) or (viva-verbose nil) at the Lisp > prompt to control the amount of output generated by ViVa.

#### VIVA EXPRESSIONS:

ViVa expressions conform to a subset of the syntax for the C programming language. Specifically, ViVa supports all syntax defined in section 18.1 of the original Kernighan and Ritchie book, plus all C numerical syntax including floats and radix syntax (i.e. 0xnnn, 0bnnn, and 0onnn). ViVa supports multiple array subscripts.

#### ABOUT VIVA AND PARCIL

ViVa uses PARCIL, copyright (c) 1992 by Erann Gat. Parcil is used under the terms of the GNU General Public License as published by the Free Software Foundation. PARCIL parses expressions in the C programming language into Lisp. ViVa then takes the parsed statements and creates an interactive environment in which they are evaluated. Thanks to Sandy Weisberg for providing Parcil.

#### ViVa EXAMPLES

Examples of each way of using ViVa are given below.

1) At the Listener, type a ViVa expression enclosed in brackets. It is evaluated and the value returned. If the expression involves assignment to a variable, the variable is assigned the value, and the variable is saved for the duration of the session. The variable can be entered into a data object, using the DATASET function, and then saved permanently.

```
Here is an example:
> [A = 2 + (3*4)]
14
```

Here, the user has typed the ViVa expression  $A=2+(3*4)$  enclosed in brackets. ViSta responds with 14, the appropriate value, using the standard rules of operator precedence. A new Lisp variable A has been created:

```
> a
14
>
```

You can retrieve the list of all saved variables created by ViVa by typing

```
> $viva-vars
```

These variables, and those created by the VAR function, are free variables (not in a data object). You can get a list of all free variables by typing:

```
> $free-vars
```

You can also retrieve the list of all variables in all data objects by typing:

```
> $data-vars
```

Note that any Lisp function may be used in ViVa expressions, though in standard algebraic syntax. For example:

```
> [a=3*iseq(4)]
(0 3 6 9)
```

where `iseq(4)` is the ViVa equivalent of Lisp's `(iseq 4)`, which generates the sequence of numbers (0 1 2 3).

These functions can use any \$data-vars as arguments:

```
> [b=a^2]
> (0 9 36 81)
```

A Lisp expression involving a function with multiple arguments is written as a ViVa function involving arguments that are comma-delimited. Thus

```
> [L=list(1,2,3)]
```

corresponds to the Lisp expression `(setf L (list 1 2 3))`.

Note that a ViVa expression may be a compound expression:

```
> { ( x=1,y=2,print(x+y),sin(pi/2) ) }
3
1.0
> x
1
> y
2
>
```

2) At the Listener, type a left bracket and a return. Lisp's `>` prompt is replaced by ViVa's `?` prompt. Then you can type a series of ViVa expressions, each being evaluated when you type a return. Finally, type a right bracket to return to regular Lisp. Notice that functions with multiple arguments have their arguments separated by commas. Also, notice that vector and matrix algebra are supported. Here is an example interaction:

```
> [
? 2+3
5
? a
(0 3 6 9) ;this value for A is left from above
? a=2+3
5
? a ;a new value for A has been defined
5
? b=sqrt(a)
2.23606797749979
? a=iseq(4) ;yet another value for A
(0 1 2 3)
? b=sqrt(a) ;vector arithmetic is supported
(0.0 1.0 1.4142135623730951 1.7320508075688772)
? ] ;the value of the last expression is returned
(0.0 1.0 1.4142135623730951 1.7320508075688772)
>
```

If you do not wish to see so much output, type:

```
[viva_verbose=not(t)]
```

For example:

```
> [viva_verbose=not(t)
? a=normal_rand(10)
? b=a^2
? c=iseq(10)
? d=c*b
? ]
(0.0 0.12086435903614712 0.12465600120144842 0.05113395472276512 0.8572483580685174
2.8192853102290143 8.009152301820079 33.31659449692597 0.9436767853187026 1.1345846087676839)
> d
(0.0 0.12086435903614712 0.12465600120144842 0.05113395472276512 0.8572483580685174
2.8192853102290143 8.009152301820079 33.31659449692597 0.9436767853187026 1.1345846087676839)
>
```

Lisp's matrix language may be used with ViVa. Here is an example:

```
?a=matrix ( list(2,2), list(1, 2 ,3 ,4))
#2A((1 2) (3 4))
?b=matrix ( list(2,3), list(1,2,3,4,5,6))
#2A((1 2 3) (4 5 6))
?c=matmult(a,b)
#2A((9.0 12.0 15.0) (19.0 26.0 33.0))
```

3) Enter a bracket enclosed ViVa statement in the middle of lisp. It is evaluated and returns its value to Lisp:

```
>(list 1 3 [sqrt(25)] 7)
(1 3 5.0 7)
```

## TWO EXAMPLES

The first example involves calculating points in an Introductory Statistics course. The data are in P3099pts.lsp datafile. You can open these data with the OPEN DATA menu item in the FILE menu. In this example, ViVa is used to create variables, then the DATASET function is used to create a data object from them.

```
[
? homework_sum=homework1+homework2+homework3
(12.0 14.24 13.399999999999999 13.11 15.36 12.96 12.22 13.82 14.31 13.09 13.98 13.67 7.45 15.33 12.1 12.92
5.57)
? total=homework_sum+midterm1
(170.0 173.24 168.4 189.11 197.36 119.96 159.22 135.82 194.31 189.09 197.98 189.67 159.45 194.33 153.1
176.92 173.57)
? ]
(170.0 173.24 168.4 189.11 197.36 119.96 159.22 135.82 194.31 189.09 197.98 189.67 159.45 194.33 153.1
176.92 173.57)
> (dataset summary homework_sum total)
#<Object: a9e82c, prototype = MV-DATA-OBJECT-PROTO>
>
```

The second example involves calculating grades from points earned during the course.

```
(load (strcat *data-dir-name* "p3099pts.lsp"))

[homeworksum = homework1 + homework2 + homework3]
[midpts = homeworksum + midterm1]

(let ((grades (mapcar #'(lambda (pts)
  (cond ((> pts 190) "A")
        ((< 170 pts 190) "B")
        ((< 150 pts 170) "C")
        ((< 140 pts 150) "C-")
        ((< 0 pts 150) "D"))))
      midpts))))

(dataset homework1 homework2 homework3 homeworksum
  midterm1 midpts grades)
```

## UNDERSTANDING DATA - Managing Data - Making Variables (III.8.5)

MAKING VARIABLES with VAR - ViSta's Variable-Object Maker

The VAR function creates a XLS+ variable-object.

Syntax: (var variable &optional form &key (type numeric))

VAR represents the XLS+ variable object by two variables: VARIABLE and \$VARIABLE.

VARIABLE is an XLS variable whose name is bound to the value of the result of evaluating form. The name of the new XLS variable is added to the \$free-vars and \$all-vars lists.

The variable type can be specified as being 'category, 'numeric, 'ordinal, 'freq, or 'label. If not specified, the variable type will be 'numeric when all values in the result of evaluating FORM are numeric, otherwise the type will be 'category.

If the name VARIABLE involves dashes, an additional XLS variable is created where the dashes are replaced by underscores, facilitating ViVa processing.

\$VARIABLE is an XLS+ variable-object whose name is bound to the variable object. The new XLS+ variable name is not added to any lists, nor is it checked for dashes. You can send the :TYPE :NAME and :VALUE messages to \$VARIABLE.

If either VARIABLE or \$VARIABLE is already bound and the global variable \*ASK-ON-REDEFINE\* is not nil then you are asked if you want to redefine the variable.

The form (VAR VARIABLE) reports the value of VARIABLE, if it exists.

Example: uses VAR VIVA and DATASET together

```
> (var final-points (* 2 (list 78 45 67 93 89)))
(156 90 134 186 178)
> final-points
(156 90 134 186 178)
> final_points
(156 90 134 186 178)
> $final-points
#<Object: b1ff10, prototype = VAR-PROTO>
> (send $final-points :value)
(156 90 134 186 178)
> (send $final-points :type)
NUMERIC
> (send $final-points :name)
FINAL-POINTS
> [midterm=list(23,10,14,30,28)]
(23 10 14 30 28)
> [total=final-points+midterm]
"; evaluation error: The variable FINAL is unbound."
> [total=final_points+midterm]
(179 100 148 216 206)
> (defun points-to-grade (points)
  (mapcar #'(lambda (score)
    (cond ((< 199 score) "A")
          ((< 174 score 200) "B")
          ((< 149 score 175) "C")
          ((< 124 score 150) "D")
          ((< 0 score 125) "F")))
    points))
POINTS-TO-GRADE
> (var grades (points-to-grade total))
("B" "F" "D" "A" "A")
> grades
("B" "F" "D" "A" "A")
>
> $grades
```

```
#<Object: d8da98, prototype = VAR-PROTO>
> (send $grades :type)
CATEGORY
> (dataset overall midterm final-points total grades
   :types (numeric numeric numeric category))
OVERALL
>
```

## UNDERSTANDING DATA - Managing Data - Making Data (III.8.6)

### DATA OBJECTS AND THE DATASET MACRO

DATA OBJECTS are created by the DATASET macro from a collection of variables. DATASET can also report the names of data objects and can be used for more advanced data manipulation as well. We discuss the macro in this help file.

---

### CREATING DATA OBJECTS

---

DATASET is used to create a data object from a collection of variables. This is done by typing:

```
(DATASET NAME VAR1 VAR2 ...)
```

where NAME is the name of the new data object being created, and VAR1 VAR2, etc, are the names of numeric variables. When the variables are not all numeric, you must include the :TYPES keyword, followed by the types:

```
(DATASET NAME VAR1 VAR2 :TYPES (NUMERIC CATEGORY))
```

The variables must all have the same number of observations. You may use long or short names (see the ABOUT VARIABLES AND DATA help file). While short names are easier to type, they introduce the possibility of duplicate variable names. If there are duplicates, the most recently created variable is usually used, though ambiguity is possible. Long names, while more cumbersome, prevent the possibility of duplicate names. The variables may be "free" variables (i.e., those which were created by ViVa or VAR) or "data" variables (those already in data objects). Variables which are in more than one data object are distinct variables: Changes made to one will not cause changes in the other.

To find out what variables are available to form a new data object, type

```
NAME      INFORMATION
$vars     list of variables in the current data
$all-vars list of all variables
$data-vars list of all data object variables
$free-vars list of all free variables
$NAME-vars list of all variables in data object NAME
```

You may label your observations by using the :LABELS keyword, which must be followed by a list of strings specifying observation names ("Obs1", "Obs2", etc., by default).

```
(DATASET NAME VAR1 VAR2
 :TYPES (NUMERIC CATEGORY)
 :LABELS ("A" "B" "C"))
```

If the data are frequency data, you must use the :FREQ keyword followed by T (for true) to specify that the values of the numeric variables are frequencies. For example

```
(dataset freqdata frequency center treatment
 :types (numeric category category)
 :freqs t)
```

You may use the :ABOUT keyword, followed by an optional string of information about the data.

Given the arguments discussed above you can specify:

- 1) MULTIVARIATE data are data which are not one of the other data types given below. These data include univariate (one variable) and bivariate (two variables) data.
- 2) CATEGORY data have one or more CATEGORY variables and no NUMERIC or ORDINAL variables. The N category variables define an n-way classification.
- 3) CLASSIFICATION data have one NUMERIC variable and one or more CATEGORY variables. The N category variables define an n-way classification. The numeric variable specifies an observation for a given classification.
- 4) FREQUENCY CLASSIFICATION data are classification data whose numeric variable specifies frequencies as indicated by using FREQ. The N category variables define an n-way classification, with the numeric variable specifying the co-occurrence frequency of a specific combination of categories.

#### ADVANCED USES: FREQUENCY TABLE DATA AND MATRIX DATA

Frequency table data are data whose observations and variables are used to form the rows and columns of a two-way table. That is, the data are a two-way cross tabulation of the co-occurrence frequency formed from the observations and variables. The data elements must be frequencies. The variables must be NUMERIC and :FREQ T must be specified. In addition, :ROW-LABEL and :COLUMN-LABEL must be used. Each must be followed by a string. The string is used to label the rows or columns of the table.

MATRIX data are data whose observations and variables refer to the same things. These things are used to form a square, usually symmetric matrix with the same number of rows and columns, the rows and columns identifying the same things. The values might be correlations, covariances, distances, etc. Optionally, there can be more than one matrix in a given data object. All matrices must have rows and columns identifying the same things. The keyword :MATRICES, used only for matrix data, must be followed by a list of strings specifying matrix names (and, indirectly, the number of matrices). :SHAPES, optional for matrix data only, must be a list of strings "symmetric" or "asymmetric" (case ignored), specifying the shape of each matrix (all are symmetric by default).

#### THE LONG-FORM OF THE DATASET MACRO

The complete "long-form" of the DATASET function creates a new data object from information contained within the DATASET statement. The minimum required syntax is:

```
(DATASET NAME
  :VARIABLES (VARLIST)
  :DATA (DATALIST) )
```

For example:

```
(dataset example
  :variables (abc def)
  :data (1 2 3 4 5 6))
```

The required arguments are discussed next, with optional long-form arguments following:

REQUIRED ARGUMENTS:  
NAME &KEY :DATA :VARIABLES

NAME must be a string or a symbol. This is the name of the newly defined data object.

:VARIABLES must be followed by a list of strings or symbols defining variable names (and, indirectly, the

number of variables).

:DATA must be followed by a list of numbers, strings or symbols (symbols are converted to uppercase strings). The number of data elements must conform to the information in the other arguments.

GENERAL OPTIONAL ARGUMENTS:  
&KEY :TYPES :LABELS :FREQ :ABOUT

:TYPES must be followed by a list of strings "numeric", "ordinal" or "category" (case ignored), or symbols (same as strings, but no quotes) specifying whether the variables are numeric, ordinal or categorical (all numeric by default).

:LABELS must be followed by a list of strings specifying observation names ("Obs1", "Obs2", etc., by default).

:FREQ must be followed by T to specify that the values of the numeric variables are frequencies.

:ABOUT is followed by an optional string of information about the data.

---

#### ADDITIONAL USES FOR THE DATASET MACRO

---

The dataset macro may also be used to find out information about data objects. In particular, to see a list of all data objects, type:

(DATASET)

and to see the object identification of data object NAME, type:

(DATASET NAME)

Finally, you can create a new data object from a program contained within the DATASET macro by typing:

(DATASET NAME FORM)

where NAME is the name of the new data object, and FORM is a Lisp form, as described by Tierney (1990) or Steele (1990).

## TOPIC 9 - GRAPHING DATA

### Plot Menu Overview (III.9.2)

#### PLOT MENU SUMMARY

The PLOTS menu allows you to generate individual plots of your data. The items of the menu generate plots whose nature is summarized here. For more information, choose a plot and then use the plot's help button.

HOLLOW-HISTOGRAM makes hollow frequency histogram plots (and frequency polygons or dynamic histograms) from raw data, based on frequencies calculated by binning values of a numeric variable. The formation of the bins is under the control of the user and may be changed dynamically via buttons on the plot. The plot cannot be linked to other plots.

DYNAMIC -HISTOGRAM makes dynamic frequency histogram plots (and frequency polygons or hollow histograms) from raw data, based on frequencies calculated by binning values of a numeric variable. The formation of the bins is under the control of the user and may be changed dynamically via buttons on the plot. The plot cannot be linked to other plots.

LINKABLE -HISTOGRAM makes static frequency histogram plots based on frequencies calculated by binning values of a numeric variable. The plot consists of individual tiles which can be linked to other plots.

DISTRIBUTION-PLOT makes frequency distribution polygons (and dynamic or hollow histograms) from raw data, based on frequencies calculated by binning values of a numeric variable. The formation of the bins is under the control of the user and may be changed dynamically via buttons on the plot. The plot cannot be linked to other plots.

CUMULATIVE-PLOT makes quantile plots and normal probability plots of numeric variables to show the shape of the data's cumulative distribution.

COMPARISON-PLOT makes quantile-quantile plots which can be used to compare the shape of the distributions of two numeric variables.

DOT-PLOT makes a scatterplot of the first variable versus the observation sequence number. Dotplots can be used to determine if there is a relationship between a variable and the order in which it appears in the data. The two variables are represented by the X and Y axes. The observations are represented as points in the plot.

SCATTER-PLOT makes a scatterplot of the first two variables in the data. Scatterplots can be used to display the relationship between two variables. The two variables are represented by the X and Y axes. The observations are represented as points in the plot.

SPINNING-POINTS makes a three-dimensional spinnable scatterplot of the first three numeric variables in the data. Scatterplots can be used to display the relationship between variables. The variables are represented by the X, Y and Z axes. The observations are represented as points in the 3D space.

ORBITING-POINTS makes a six-dimensional spinnable scatterplot of the first six numeric variables in the data. Scatterplots can be used to display the relationship between variables. The variables are represented by the six axes. The observations are represented as points in the 6D space which are orbiting around their centroid. Points which are close together in 6D space will have similar orbits.

LINE -PLOT makes a connected scatterplot of the first variable versus the observation sequence number. Line plots can be used to determine if there is a relationship between a variable and the order in which it appears in the data. The two variables are represented by the X and Y axes. The observations are represented as connected points in the plot.

BOX-PLOT makes boxplots from the numeric variables in your data. A boxplot is a schematic of a set of observations based on the variable's median and quartiles. The schematic can give you insight into the shape of

the distribution of observations, and allows you to compare distribution shapes of different variables.

DIAMOND-PLOT makes diamond plots from the numeric variables in your data. A diamond plot is a schematic of a set of observations based on the variable's mean and standard deviation. The schematic can give you insight into the shape of the distribution of observations, and allows you to compare distribution shapes of different variables.

GROUPED BOX PLOT makes boxplots for each group in you data, using a categorical variable to form the groups and a numeric variable to form the boxplots. The boxplots can give you insight into the shape of the distribution of observations in each group, and allows you to compare distribution shapes of different groups.

BAR-GRAPH makes bar-graphs of as many as 4 categorical variables. The first several categorical variables are used. A bar graph consists of vertical bars whose height represents the frequency of crosstabulated categories of the data. Can create side-by-side frequency or probability bar graph of 1- to 4-way data, or stacked bar graph for 2- or 3-way data.

MOSAIC -PLOT makes mosaic plots of as many as 4 categorical variables. The first several categorical variables are used. A mosaic plot consists of tiles whose area represents the frequency of crosstabulated categories of the data.

### **UNDERSTANDING DATA - Graphing Data - The Box, Diamond and Dot Plots (III.9.3)**

The Box, Diamond and Dot plot uses boxes, diamonds and dots to form a schematic of a set of observations. The schematic can give you insight into the shape of the distribution of observations. Some Box, Diamond and Dot plots have several schematics. These side-by-side plots can also help you see if the distributions have the same average value and the same variation in values.

The plot always displays dots. They are located vertically at the value of the observations shown on the vertical scale. (The dots are 'jittered' horizontally by a small random amount to avoid overlap).

The plot can optionally display boxes and diamonds. Boxes summarize information about the quartiles of the variable's distribution. Diamonds summarize information about the moments of the variable's distribution. The BOX and DIAMOND buttons at the bottom of the graph control whether boxes or diamonds (or both) are displayed.

The box plot is a simple schematic of a variable's distribution. The schematic gives you information about the shape of the distribution of the observations. The schematic is especially useful for determining if the distribution of observations has a symmetric shape. If the portion of the schematic above the middle horizontal line is a reflection of the part below, then the distribution is symmetric. Otherwise, it is not.

In the box plot, the center horizontal line shows the median, the bottom and top edges of the box are at the first and third quartile, and the bottom and top lines are at the 10th and 90th percentile. Thus, half the data are inside the box, half outside. Also, 10% are above the top line and another 10% are below the bottom line. The width of the box is proportional to the total number of observations.

The diamond plot is another schematic of the distribution, but it is based on the mean and standard deviation. The center horizontal line is at the mean, and the top and bottom points of the diamond are one standard deviation away from the mean. The width is proportional to the number of observations. The diamond is always symmetric, regardless of whether the distribution is symmetric.

In side-by-side plots, both the box plot and diamond plot can be used to see if the distributions have the same central tendency and the same variation. If the several medians, as well as the several means, are all about the same, then the central tendency for each distribution is about the same. If the diamonds are all approximately the same size vertically, and if the boxes are also all about the same size vertically, then the distributions have about the same variation.

**UNDERSTANDING DATA - Graphing Data - Histogram and Distribution Plots (III.9.4)**

We are sorry, but help on this topic is not yet available.

**UNDERSTANDING DATA - Graphing Data - Cumulative Plot (III.9.5)**

The Cumulative Distribution Plots menu item gives you access to two plots designed to show you a variable's cumulative distribution. These plots are:

## THE QUANTILE PLOT

## THE NORMAL PROBABILITY PLOT

The Quantile plot (Q-Plot) pictures a variable's cumulative distribution by plotting the value of a specific datum versus the fraction of the data that is smaller than the specific datum. The jagged line represents the variable's cumulative distribution.

The Normal Probability Plot (NP-Plot) pictures a variable's cumulative distribution by plotting the value of a specific datum versus the Z-score that would be obtained for the datum under the assumption of normality.

Note that the NP -Plot is the same as the the Q-Plot except that the Q-plot's Fraction of Data (empirical probability) is converted into the NP -Plot's Z -Scores, where the Z-Scores has the stated probability.

## ABOUT THE Q-PLOT

The Q-Plot is used to help decide on the symmetry of a variable's distribution. Symmetry is not displayed in the usual sense. Rather, for a symmetric distribution, the points in the upper half of the plot will stretch out toward the upper right the same way the points in the bottom half stretch out toward the lower left.

There are several reasons why symmetry is important for data analysis: 1) The center of a distribution is unambiguous for a symmetric distribution. 2) Symmetric distributions are easier to understand (the upper part is like the lower part); and 3) Symmetric distributions are amenable to stronger statistical analysis than asymmetric distributions.

When you click on the Y button at the top of the graph you will be presented with a list of variables to display. Clicking on a variable will change the plot to display that variable on the Y-axis. (If there are only two variables, it toggles between them.)

Clicking on the X button at the top of the graph toggles the X-axis between "Fraction of Data", and "Z-Score of Fraction of Data". It also toggles the entire graph between a Quantile Plot and a Normal Probability Plot.

## ABOUT THE NP -PLOT

In the NP -Plot, the jagged line represents the variable's distribution and the straight line represents a normal distribution. If the jagged line is roughly linear, so that it approximately follows the straight line, the variable has an approximately normal distribution.

Systematic departures from a straight line indicate non-normality. Such departures include large deviations, which indicate outliers; asymmetric departures at one end or the other, indicating skewness; and horizontal segments, plateaus or gaps, which indicate discrete data.

Normality is important because very many inferential statistical procedures assume that the data are normally distributed. The normal-probability plot gives us a visual approach to checking on this critical assumption.

When you click on the Y button at the top of the graph you will be presented with a list of variables to display. Clicking on a variable will change the plot to display that variable on the Y-axis. (If there are only two variables, it toggles between them.)

Clicking on the X button at the top of the graph toggles the X-axis between "Fraction of Data", and "Z-Score of Fraction of Data". It also toggles the entire graph between a Quantile Plot and a Normal Probability Plot.

### **UNDERSTANDING DATA - Graphing Data - Comparison Plot (III.9.6)**

#### **COMPARISON PLOT**

The COMPARISON PLOT uses what is usually known as a Quantile-Quantile plot (QQ-Plot) to aid you in comparing the distributions of two numeric variables. In the QQ-plot, the quantiles of two variables are plotted against each other, forming the jagged blue line. This line represents the relationship between the two distributions. Since, for these data, the two variables have the same number of observations, the jagged blue line is simply a plot of one sorted variable against the other sorted variable.

The blue line on the QQ-Plot tells us whether the two variables have distributions that have the same shape. If the line is roughly straight, the two variables have roughly the same shape. This is important to know, since many analyses assume that the variables are "identically" distributed, which means they have the same shape. When two variables are normally distributed, for example, they have the same shape.

#### **CENTER AND SPREAD:**

The straight dashed black line represents two identically distributed variables (this line does not appear when the centers of the two variables are very different). The straight red line represents two variables whose distributions are the same shape and which have measures of center and spread which are like those of the observed variables. Such distributions are geometrically "similar", since they have the same shape.

When the dashed and red lines are parallel but not near each other, the measures of spread of the observed distributions are about the same, but the centers are different. When the two lines are near each other but not parallel, then the observed distributions have roughly the same centers, but different spreads.

The measures of center and spread that are compared in this plot are the mean and variance of the quantiles. If the jagged blue line is systematically different from a straight line the distributions of the two variables do not have the same shape, and are not geometrically similar. Outliers appear as large deviations from the straight line.

If the jagged blue line is roughly straight, the two variables have approximately the same shaped distributions. If the blue line approximately follows the dashed line, then the two distributions are roughly identical. If it approximately follows the red line, but not the dashed line, the two distributions are "similar", but have different centers and spreads.

When you click on the X or Y buttons at the top of the graph you will be presented with a list of variables to display. Clicking on a variable will change the plot to display the variable on the X or Y axis.

### **UNDERSTANDING DATA - Graphing Data - Mosaic Plot and Bar Chart (III.9.7)**

#### **MOSAIC PLOTS (AND BAR GRAPHS)**

A mosaic plot shows the frequencies in an n-way table by nested rectangular regions whose area is proportional

to the frequency in a cell or marginal subtable. The display uses color and shading to represent the sign and magnitude of standardized residuals from a specified model.

#### Comparison of Mosaic Plots and Bar Graphs

A mosaic plot presents the same information as is presented by a stacked bar-graph: The frequencies of combinations of categories of two variables.

1. A mosaic plot consists of rectangles laid out in a mosaic. The rectangles are like the sub-bars in a stacked bar-graph.
2. In a mosaic plot, each column of rectangles represents a category of the variable on the horizontal axis.
3. In a stacked bar-graph, each bar represents the overall frequency of a category of the variable plotted on the horizontal axis. In a mosaic, the several column of tiles are all the same height, representing 100%. Thus each tile in a mosaic represents a proportional frequency of a category combination.
4. Whereas a stacked bar-graph's sub-bars representing the joint frequency of a category of each of the two variables, in a mosaic plot each rectangle represents the joint probability of a category of each of the two variables.

#### **UNDERSTANDING DATA - Graphing Data - Scatter Plot (III.9.8)**

The scatterplot is designed to display the relationship between two variables. The variables are represented by the X-axis and Y-axis. The observed values on the two variables are represented by points in the scatterplot. Each point represents the values for (usually) one observation on two variables. The value can be approximately determined by seeing what value the point is above on the X-axis, and to the right of on the Y-axis.

Two normally distributed variables will have a scatterplot which has the greatest density in the middle, is roughly elliptical in shape, and has no obvious outliers.

#### **UNDERSTANDING DATA - Graphing Data - Spinning Points Plot (III.9.9)**

The Spin plot presents a three-dimensional scatterplot which you can spin to observe the distribution of your data in three dimensions.

The Up/Dn, C/CC, and L/R buttons at the bottom of the window control spinning direction (the button names mean Up/Down, Clockwise/Counterclockwise and Left/Right). Click on a button to change direction. The SPEED buttons control the speed of spinning. Hold down a button to gradually increase or decrease the speed.

The SPIN button starts and stops rotation. You can also spin the space by using the HAND ROTATE tool (whose cursor is a hand) and dragging your cursor around the space. If your cursor is not a hand, you can get this tool by clicking on the MOUSE button at the top of the window. Finally, you can spin the space by holding down the shift key while you click on the rotate buttons, or while you use the HAND ROTATE tool: The spinning will continue by itself until you click on the SPIN button.

The HOME button returns the space to its original orientation and stops rotation. The ROCK button reverses the direction of rotation: Repeatedly clicking on this button rocks the space back and forth, giving you a better view of the detailed location of points in the space. The CLIP button clips off garbage that sometimes appears at the edges of the window. Holding down a ZOOM button will gradually zoom the space in or out.

At the top of the window are buttons that give you this help message, turn color on and off, and allow you to change the mouse-mode. In addition, the BOX button allows you to add or remove a three-dimensional box that encompasses the data points. The X, Y and Z buttons allow you to change which variable is shown on the X, Y, or Z axis.

**UNDERSTANDING DATA - Graphing Data - Scatter Matrix (III.9.10)**

The scatterplot matrix is designed to display the relationship between all pairs of several variables. The plot matrix consists of plotcells containing little scatterplots formed from a pair of variables. The variables are represented by the X-axis and Y-axis of each plotcell. The observed values on the two variables are represented by points in the little scatterplot. Each point represents the values for (usually) one observation on two variables.

Normally distributed variables will have scatterplots which have the greatest density in the middle, are roughly elliptical in shape, and have no obvious outliers.

The scatterplot matrix can be used as a control panel for selecting variables, pairs of variables and triples of variables. When you click on a plotcell, that pair of variables is selected. When you click on a diagonal cell, a single variable is selected. Shift-clicking will select more variables.

When you use the scatterplot matrix to select one, two or three variables, various changes will appear in other plots. The specific changes depend on the nature of the SpreadPlot.

**UNDERSTANDING DATA - Graphing Data - Orbiting Points Plot (III.9.11)****TOUR CONTROLS**

The Guided Tour window presents a spinning 6-dimensional scatterplot. Initially, the space corresponds to the first 6 variables in your data, and it is spinning in the 3D space defined by the first 3 variables.

High-dimensional rotation is called a 'tour' of high-dimensional space. The tour is initiated by clicking on the TOUR button. The speed of the tour is controlled by the HSpd and VSpd buttons, which control the speed of rotation of the cloud in the Horizontal and Vertical directions. The minimum speed is zero. Holding down these buttons gradually increases the speed. Note that when the cloud is spinning both horizontally and vertically, the result is a four-dimensional tour which appears as moving points, not as rotation.

The beginning position of the point cloud is the same as that shown in the First Target window. When the horizontal and vertical dimensions are rotated 270 degrees the position is the same as that shown in the Second Target window. Further rotation by 90 degrees brings the cloud back to the First Target position.

The NEW button creates a new 6 dimensional space for a new tour. It does this by using the 3 dimensions of the current position shown in the window as the new 'First Target', and by defining the highest variance 3-dimensional space which is orthogonal to the new First Target as being a new Second Target.

**SPIN CONTROLS**

The description given in this section is from the general help about spinning plots. The description also applies to the tour plot, except that the HOME and ROCK buttons have been redefined to work within a six-dimensional context.

The Spin plot presents a three-dimensional scatterplot which you can spin to observe the distribution of your data in three dimensions.

The Up/Dn, C/CC, and L/R buttons at the bottom of the window control spinning direction (the button names mean Up/Down, Clockwise/Counterclockwise and Left/Right). Click on a button to change direction. The SPEED buttons control the speed of spinning. Hold down a button to gradually increase or decrease the speed.

The SPIN button starts and stops rotation. You can also spin the space by using the HAND ROTATE tool (whose cursor is a hand) and dragging your cursor around the space. If your cursor is not a hand, you can get this tool by clicking on the MOUSE button at the top of the window. Finally, you can spin the space by holding down the shift key while you click on the rotate buttons, or while you use the HAND ROTATE tool: The spinning will continue by itself until you click on the SPIN button.

The HOME button returns the space to its original orientation and stops rotation. The ROCK button reverses the direction of rotation: Repeatedly clicking on this button rocks the space back and forth, giving you a better view of the detailed location of points in the space. The CLIP button clips off garbage that sometimes appears at the edges of the window. Holding down a ZOOM button will gradually zoom the space in or out.

At the top of the window are buttons that give you this help message, turn color on and off, and allow you to change the mouse-mode. In addition, the BOX button allows you to add or remove a three-dimensional box that encompasses the data points.

## TOPIC 10: Analyzing Data

### UNDERSTANDING DATA - Analyzing Data - Analysis of Variance ... (III.10.2)

Analysis of Variance (ViSta-ANOVA) is a technique for comparing means of several populations. ViSta can perform one-way, two-way, or n-way analysis of variance with optional two-way interactions. The samples from the populations may be all the same size (balanced) or may be different sizes (unbalanced). There must be at least one observation in each cell (the data must be complete).

In one-way ANOVA samples are drawn from each population and the data are used to test the null hypothesis that the population means are equal.

In two-way ANOVA the populations are classified in two ways. In n-way ANOVA the populations are classified in three or more ways. For these types of ANOVA, the analysis still compares the means of populations. However, in two-way and n-way ANOVA several comparisons of the sample means are possible since there are several classifications. In particular, comparisons are made for the classifications themselves (called "main effects") and for the interactions of each pair of classifications (called "two-way interaction effects"). ViSta-ANOVA does not compare interaction higher than two-way.

ViSta-ANOVA provides significance tests to test the null hypothesis that the group means, as classified, are all equal. For the significance tests to be accurate we must assume that the samples are drawn independently from normally distributed populations which have the same standard deviations.

The ViSta-ANOVA visualization presents plots to help you assess the assumptions and to help you understand the results of the significance tests.

### UNDERSTANDING DATA - Analyzing Data - Regression Analysis ... (III.10.3)

Regression Analysis (ViSta-Regres) is a technique for predicting one response variable from one or more predictor variables. ViSta-Regres does simple and multiple regression. These are called univariate regression because only one response is being predicted. If you wish to predict more than one response, use ViSta-MulReg, which does multivariate multiple regression.

OLS (ordinary least squares) regression is the most common type of regression. It finds the strongest linear relationship between a linear combination of the predictors and the response. OLS regression assumes that errors are normally distributed and independent, and that predictors are measured without error.

OLS regression is the statistically best method when the assumptions are satisfied. However, when the data fail to meet the assumptions or when there are problems such as outliers or colinearity in the data, the OLS coefficients may be inaccurate estimates of the population parameters.

The ViSta-Regres visualization helps you check on the validity of the OLS assumptions. It also help show you outliers. If the data fail to meet the assumptions of OLS regression, you may use the robust or monotonic regression methods to analyze the data. Robust regression produces regression coefficients which are not influenced by outliers. Monotonic regression is useful when the relationship between the response variable and the predictor variables is nonlinear. You can compare the results provided by OLS, robust, and monotonic regression to determine which is most appropriate for your data.

### UNDERSTANDING DATA - Analyzing Data - Univariate Analysis ... (III.10.4)

Univariate Analysis (ViSta-UniVar) provides techniques for comparing means of two populations. ViSta-UniVar can compare two sets of data whether they are independent or paired (dependent). It tests whether the means of the two groups are significantly different, and reports the confidence interval for the difference in means.

For samples from independent populations ViSta-UniVar computes Student's T-test and the Mann-Whitney test. For paired (dependent) samples the paired-samples T-Test and the Wilcoxon Signed Rank Test are computed. Student's T-test is used when there is a single sample. ViSta-UniVar can also use the T-test to compare the mean of one population to a pre-specified hypothetical mean. If the population variance is known, then the Z-test is substituted for the T-test.

The T-test (and Z-test) test the null hypothesis that the means of the populations from which the data are sampled are equal. The Mann-Whitney U-test and the Wilcoxon Signed Rank Test use the null hypothesis that both populations are identically distributed.

The ViSta-UniVar visualization presents plots to help you assess the normality assumption.

#### UNDERSTANDING DATA - Analyzing Data - Frequency Analysis ... (III.10.5)

The Frequency Distribution Plots menu item gives you access to three plots designed to show you the shape of a variable's distribution. These plots are:

- THE POLYGON PLOT
- THE HISTOGRAM PLOT
- THE HOLLOWGRAM PLOT

You can use the PLOTS button at the bottom of the graph to cycle through three ways of plotting the frequency information: Histogram, Hollow Histogram and Frequency Polygon.

All three of these plots are based on the same approach for determining the shape of a variable's distribution. The three plots differ in how that shape is drawn and shown to you.

These three plots determine the shape of a variable's distribution by breaking the range of the variable's values into equal-sized intervals called BINS. They then determine the number of observations that fall into the interval (are in the BIN) and display the shape of the distribution as follows:

- The HISTOGRAM draws a BAR for each BIN
- The HOLLOWGRAM draws a HORIZONTAL LINE for each BIN
- The POLYGON draws a JAGGED LINE for each BIN

For each plot there is a set of red dots. These red dots are located so that their height is proportional to the frequency in the interval. The higher the red dot, the greater the frequency in the bin. The red dots are located above the midpoint of each bin, and to the right of the frequency of the bin. You can brush your cursor over the red dots to see the frequency and midpoint of the bin.

#### BINWIDTH

Unfortunately, all three plots are notorious for conveying an impression of the shape of the variable's distribution that is strongly dependent on the number of bins chosen. Changing the number of bins may radically change the apparent shape of the distribution. Even more unfortunately, there is no entirely satisfactory way to solve this problem.

For this reason there are two buttons on the graph that help you control the number of bins. These are the two BINWIDTH buttons at the bottom and the NEWBINS button at the top.

The two BINWIDTH buttons can be used to dynamically change the bin widths, and, consequently, the number of bins. By putting your cursor on the button and holding your mouse button down, these buttons allow you to watch the graph change in an animated way. Clicking on the NEWBINS button gives you a dialog box that lets you

customize the bin widths and midpoints (as well as the x-axis) to get a better distribution.

We recommend that you first use the BINWIDTH buttons to get a better impression of the distribution's shape, and then the NEWBINS button to choose a "nice" bin width and midpoint. "Nice" means that the distribution adequately portrays the shape of the distribution, and the bin widths, midpoints and axis details use sensible numbers.

The CURVES button can be used to add or remove several different distribution curves, including the normal distribution and several curves called "kernel density distribution curves". The kernel density distribution curves provide several alternate ways of approximating the shape of the population distribution. If the kernel density curves roughly approximate the normal distribution curve, then the variable's distribution approximates normality.

When you click on the X button at the top of the graph you will be presented with a list of variables to display (if there are only two variables, it will switch to the other variable). Clicking on a variable will change the plot to display that variable's Frequency Polygon. When you click on the Y button at the top of the graph the y axis will switch between frequency and probability.

Finally, when you click on the DATA button at the bottom of the graph, you will create a cumulative frequency table dataobject. It contains several variables specifying frequencies and cumulative frequencies, percentages and cumulative percentages, and limits and midpoints.

### **UNDERSTANDING DATA - Analyzing Data - Log linear analysis ... (III.10.6)**

Help for loglinear analysis

Log-linear models provide a method for analyzing associations between two or more categorical variables. The method has become widely accepted as a tool for researchers during the last two decades.

The visualization for log-linear models help to specify the possible models for a group of variables and assess the fit of these models. The list of terms (the variables and all the interactions between the variables) in the visualization gives a simple and intuitive way of indicating the terms to enter in a model.

If the list of terms is in hierarchical model, all the terms below a selected one are automatically introduced in the model. If the list of terms is not in hierarchical model, terms can be selected individually. The rest of plots help to assess the fit of the model like mosaic plots for observed and predicted values or distance of Cook v. leverages scatterplots. The parameter plot sorts out the parameters of the model and provides a interpretation in terms of frequencies of the original data of each parameter. Finally, the history plot keeps information about the chi square statistic that can be used for returning to previous successful models

### **UNDERSTANDING DATA - Analyzing Data - Multivariate Regression ... (III.10.7)**

Multivariate Multiple Regression (ViSta-MulReg) is a data analysis technique for predicting the values of a many response variables simultaneously from the values of two or more predictor variables. If you have just one response variable, you should use ViSta-Regres, not ViSta-MulReg.

ViSta-MulReg uses OLS (ordinary least squares) regression techniques to find the strongest possible linear relationship between a linear combination of the predictors and one of the response variables. A different linear combination of the predictors is obtained for each response variable. This is the same as repeatedly doing many (univariate) multiple regressions. Multivariate Regression then combines all of the separate (univariate) multiple regressions together to obtain an overall multivariate test of the significance of simultaneously predicting all of the response variables.

If you choose the Redundancy option you fit a Redundancy Analysis model to your data. This model obtains the single linear combination of the predictor variables which simultaneously predicts all of the response variables optimally, in the sense of maximizing the mean squared correlation between the linear combination and each response. This mean squared correlation is higher than that obtained by Multivariate Regression.

The ViSta-MulReg visualization shows the relationship between responses, the linear combinations of predictors, and the redundancy variables. No regression diagnostics are shown (use ViSta-Regres for diagnostics).

### **UNDERSTANDING DATA - Analyzing Data - Principal Components ... (III.10.8)**

Principal Component Analysis (ViSta-PrnCmp) is a statistical analysis and visualization method for picturing the variance in a set of continuous multivariate data. Only the numeric variables in the data are used.

The ViSta-PrnCmp visualization shows the maximum variance picture of the data: The first principal component is the linear combination of the variables that has the maximum variation. The second principal component is the linear combination of the variables that is orthogonal (at right angles) to the first one, and has the maximum remaining variation. The third is orthogonal to the first two, and has maximum variance. The scree plot is also shown.

There are two major decisions to be made:

Before the analysis, you must decide whether to obtain the principal components from the correlations or covariances of the variables. The choice of covariances can only make sense if your variables are all on the same scales. If they are not, you must choose correlations. If they are all on the same scales, you may choose either, but correlations is usually most reasonable. Choosing covariances means that the original differences in variance between variables will effect the results. Choosing correlations means this difference in variance will not effect the results.

After the analysis, you must to decide how many principal components are needed to adequately summarize the data. The interpretation for the model will help you make this decision.

### **UNDERSTANDING DATA - Analyzing Data - Item Analysis ... (III.10.9)**

#### ITEM ANALYSIS

Item Analysis provides psychometric methods for analyzing items of a test where the methods are based on Classical Test Theory. These methods include split-half reliability, Cronbach's alpha, standard error of measurement, confidence intervals for observed and estimated scores, as well as psychometric indexes for the items.

Item Analysis assumes that there is one numerical variable for each item of a test. You may analyze these data or you may create different item scores from them. The item scores include summated scores, mean scores, transformed scores, estimated scores with their confidence intervals, etc.

### **UNDERSTANDING DATA - Analyzing Data - Correspondence Analysis ... (III.10.10)**

Correspondence Analysis (ViSta-Coresp) is a statistical analysis and visualization method for picturing the associations between the levels of a two-way contingency table. The name is a translation of the French "Analyses des Correspondances", where the term correspondance denotes a "system of associations" between the elements of two sets.

In a two-way contingency table, the observed association of two traits is summarized by the cell frequencies. A typical inferential aspect is the study of whether certain levels of one characteristic are associated with certain levels of another.

Correspondence analysis is a geometric technique for displaying the rows and columns of a two-way contingency table as points in a low-dimensional space, such that the positions of the row and column points are consistent with their associations in the table. The goal is to have a global view of the data that is useful for interpretation.

The ViSta-Coresp visualization displays the graphs for the rows and columns of the contingency table, and lets you interactively modify the estimated values.

### **UNDERSTANDING DATA - Analyzing Data - Homogeneity Analysis ... (III.10.11)**

Homogeneity Analysis (HOMALS) is a statistical visualization technique for picturing the associations between the levels of a set of categorical variables, and the similarities between the objects which these categories are applied too. The goal is to have a global view of the data that is useful for exploratory proposes. Only the categorical variables are included in the analysis.

Homogeneity Analysis assigns scores to the objects and it quantifies categories (optimal scaling). These scores and quantifications allow to construct a representation of the data estructure in a low dimensional space (data reduction).

Categories and objects are represented as points in a joint space. The positions of the object points are related with their similarities. Objects with similar profile are located closely in the space. A category point is the centroid of the objects that belong to this category.

Homogeneity Analysis visualization displays the graphs for the category quantification and the objects scores, and let you interactively modify the dimensions of the HOMALS solution. Also, a plot for evaluating the fit of the solution and the discrimination measures of the variables is included.

### **UNDERSTANDING DATA - Analyzing Data - Metric Averaged MDS ... (III.10.12)**

Multidimensional Scaling (ViSta-MDScal) is a data analysis and visualization technique for analyzing distance-like data (proximities, dissimilarities) and displaying a low-dimensional Euclidean space that summarizes the distance information in the data.

The data consist of distance-like information between all pairs of a set of objects. There may be several matrices of such data, and the data matrices may be asymmetric or symmetric. The data must be dissimilarities, not similarities (i.e., large numbers mean large distance). There may not be any missing data.

ViSta-MDScal locates points in a Eucliden space which has a point for every object in the data. The points are located so that those that are close together have data which are similar, and those which are far apart have data which are different. The user must decide on the appropriate dimensionality of the Euclidean space.

The ViSta-MDScal visualization is based on the initial estimates of the best locations for the points. The fit of the points to the data can be improved by using the iterate button. The visualization provides several views of the multidimensional Euclidean space, as well as a fit plot to help decide on dimensionality.

**UNDERSTANDING DATA - Analyzing Data - Multidimensional Scaling ... (III.10.13)**

Multidimensional Scaling (ViSta-MDScal) is a data analysis and visualization technique for analyzing distance-like data (proximities, dissimilarities) and displaying a low-dimensional Euclidean space that summarizes the distance information in the data.

The data consist of distance-like information between all pairs of a set of objects. There may be several matrices of such data, and the data matrices may be asymmetric or symmetric. The data must be dissimilarities, not similarities (i.e., large numbers mean large distance). There may not be any missing data.

ViSta-MDScal locates points in a Euclidean space which has a point for every object in the data. The points are located so that those that are close together have data which are similar, and those which are far apart have data which are different. The user must decide on the appropriate dimensionality of the Euclidean space.

The ViSta-MDScal visualization is based on the initial estimates of the best locations for the points. The fit of the points to the data can be improved by using the iterate button. The visualization provides several views of the multidimensional Euclidean space, as well as a fit plot to help decide on dimensionality.

**UNDERSTANDING DATA - Analyzing Data - Cluster Analysis ... (III.10.14)**

No Help Available

**UNDERSTANDING DATA - Analyzing Data - Impute Missing Data ... (III.10.15)**

The IMPUTE MISSING DATA menu item is available to you when there are missing values in your data. Since data with missing values cannot be used in ViSta's analysis methods, some way of pre-processing the data must be provided. The menu item provides three of the most common methods used to deal with missing data:

- 1) Listwise deletion: Any observation with a missing value is deleted from the dataset.
- 2) Pairwise deletion: Correlation/covariance matrices are computed on the basis of cases which do not have pairs of missing values.
- 3) Maximum Likelihood: An iterative process attempts to obtain maximum likelihood estimates of the missing values. These estimates are used to replace the missing values so that no data has to be thrown out.

## Topic 11: Modeling Data

### **UNDERSTANDING DATA - Modeling Data - About The Analysis (III.11.2)**

The ABOUT THESE DATA and ABOUT THE ANALYSIS menu items present information about data objects and analysis methods.

You can enter this information for your data objects. To do this:

1) Click on the data icon for which you wish to enter information.

2) Type:

(about-these-data "INFORMATION")

in the listener window, replacing INFORMATION with the information about the data that you wish to enter. The information must be inside double quote marks. If the information itself contains double quote marks, they must be preceded by a back-slash.

### **UNDERSTANDING DATA - Modeling Data - Visualize Model (III.11.3)**

The VISUALIZE DATA and VISUALIZE MODEL items produce a visualization of your data or model.

The data and model visualizations are designed to help you see what your data --- or the model of your data --- seem to say.

Visualizations consist of several plot windows that work together. The group of plot windows is called a SpreadPlot. Help about the spreadplot is available from the help menu's SPREADPLOT HELP item. In addition, help may be obtained for the individual plots by using the help menu's WINDOW HELP item. Finally, many plot windows have a help button which provides help about the individual plot window.

When you are finished with the visualization, use any of the close boxes. All of the windows will close together.

Each model has its own visualization. In addition, there are several different data visualizations, depending on the selection of variable types active at the time the visualization is chosen.

When all variables are numeric, there are four possible visualizations: A multivariate visualization for data with 3 or more numeric variables; a bivariate visualization for data with two numeric variables; a univariate visualization for data with one numeric variable; and a Guided Tour visualization for data which have 6 or more numeric variables.

Finally, there is a classification visualization for data which have a numeric variable and one or more category variables; a frequency visualization for frequency data (data which have numeric variables that specify frequency values); and a category visualization for category data (data which have one or more category variables and no numeric variables).

### **UNDERSTANDING DATA - Modeling Data - Report Model ... (III.11.4)**

REPORT MODEL produces a listing of numeric information about your model. The details of the report depend on which model you have chosen.

### **UNDERSTANDING DATA - Modeling Data - Interpret Model (III.11.5)**

INTERPRET MODEL

The INTERPRET MODEL menu item provides an interpretation of your model. The interpretation is a paragraph describing the results of the analysis that you have done, and the assumptions underlying the results.

### **UNDERSTANDING DATA - Modeling Data - Save Model As ... (III.11.6)**

SAVE MODEL saves your ViSta model as a file on your computer so that it can be used again at a later time.

If you are using guidemaps, then at this point you should save the model before proceeding. You do that by using the !! side of the SAVE MODEL button.

#### **UNDERSTANDING DATA - Modeling Data - Delete Model (III.11.7)**

DELETE MODEL deletes the current model object AND all associated objects, if any.

The current model object is the one highlighted on the workmap and checkmarked in the model menu.

Associated objects are:

- 1) All those "below" the current model object. An object is "below" the current model object if a path of connecting lines exists from the lower edge of the current model object to the "below" object. A "below" object may be a data object, model object, or analysis object.
- 2) Those "emmediately above" the current model object. An object is "emmediately above" the current model object if it is an analysis object which is connected directly to the upper edge of the model object. An "above" object may only be an analysis object.

#### **UNDERSTANDING DATA - Modeling Data - Create Data ... (III.11.8)**

CREATE DATA creates new data from the current model. For some models, many different kinds of data can be created, and dialog box will be presented for you to select the kinds you want. Other models cannot create new data. A message will tell you if this is the case.

#### **UNDERSTANDING DATA - Modeling Data - Print Data Analysis ... (III.11.9)**

PRINT DATA - This item prints a listing of your data on your printer.

## **PART IV: ENHANCING VISTA**

### **Topic 12: Enhancing ViSta**

ViSta is a MODERATED OPEN software system. That means that parts of its code is OPEN to those who wish to enhance the system. The items in this help topic are designed to help you understand how to take advantage of this aspect of ViSta.

#### **ENHANCING VISTA - Enhancing ViSta - Application Development (IV.12.2)**

##### HOW THE DEVELOPMENT EFFORT WORKS

ViSta consists of a core engine plus plugins and addons. This design lets ViSta be both stable and expandable: The core is stable, while the plugins and addons provide the path to growth. This architecture also provides for an obvious organization of ViSta developers into Application developers and System developers. Applications developers develop new plugins and addons, whereas system developers can also enhance ViSta's core engine.

##### APPLICATION DEVELOPMENT

If you are a system or application developer, all of the code and tools that you will need to be a ViSta Application Developer have just been downloaded onto your machine. You can proceed to develop your plugin or addon application without coordinating your efforts with those of other application developers.

However, please check out [www.visualstats.org/developer](http://www.visualstats.org/developer) for information on what other folks are doing, and to leave information about what you are doing. This way, duplication of effort can be avoided, and the user and developer community can be kept up-to-date.

If you need professional software development support, please contact [forrest@visualstats.org](mailto:forrest@visualstats.org) for information about our professional development and programming services.

##### SUBMITTING YOUR APPLICATION FOR DISTRIBUTION

If you wish, you can submit your application to us for distribution by [visualstats.org](http://visualstats.org). You are, of course, free to distribute your app independently from [visualstats.org](http://visualstats.org).

Submitting an application to us is much like submitting a paper for publication. When your app is ready, just email the installation module to [app-devel@visualstats.org](mailto:app-devel@visualstats.org)

The installation module should include code, data, examples and documentation. The ViSta Editorial Board will review and make a recommendation concerning distribution. If it is approved for distribution it will be included on the [visualstats.org](http://visualstats.org) website (and on our mirror sites), and links will be made so that it can be downloaded.

**ENHANCING VISTA - Enhancing ViSta - System Development (IV.12.3)****SYSTEM DEVELOPMENT**

ViSta System Developers have access to the entire source code, and can make changes to any portion of the system. But this isn't a free-for-all. Rather, because of the critical nature of systems development, and the importance of the core engine to the entire system, the system development effort is a coordinated effort. The effort is coordinated through the use of CVS, a version control system.

CVS permits individuals who are part of a widely distributed development effort to work independently and simultaneously on a common set of code. The code is on your machine, where your CVS client coordinates your development with that of other developers, all the while permitting you to work independently from other developers. When you have completed your changes, the central CVS server will review all code changes to detect changes which conflict with those made by other system developers. You will then need to resolve these conflicts before the changes are accepted.

Contact Forrest Young if you wish to download ViSta from the ViSta CVS server at the University of North Carolina at Chapel Hill.

**ENHANCING VISTA - Enhancing ViSta - Writing Applets (IV.12.4)**

## About ViSta Applets

Applets are code written in Lisp, ViSta's underlying programming language. Applets provide the user with a way to run ViSta and have it perform a specialized set of calculations specified by of the applet, while providing the programmer control over the initial appears of ViSta. Applets can also cause an already running instance of ViSta to perform the calculations.

Applets can be delivered over a network to a client and can be run without the standard ViSta logo and desktop windows being shown.

An applet file can contain exactly one function: the applet function. This is not a limitation, however, since the function can contain any desired code. The APPLET function has the following form:

```
(applet name args forms)
```

Where NAME is the name of the applet, ARGS is a list of arguments, and FORMS is a series of Lisp statements. For example:

```
(applet @see-data (&key (exit t) (hide-desktop t)
                 (hide-logo nil) )
  (load-data)
  (visualize-data :dialog nil)
  (when exit
    (send (send *current-spreadplot* :container)
          :make-close-menu
          (list (send menu-item-proto
                    :new "Show DeskTop"
                    :action (lambda ()
                            (send *spreadplot-container*
                                  :hide-window)
                            (show-desktop))))
              (send menu-item-proto
                    :new "Exit ViSta"
                    :action (lambda () (vista -exit))))
          )))
```

specifies an applet named @see-data which has arguments that cause the desktop and logo to be hidden, and which has statements that load data and visualizes it.

NAME is the symbolic name of the applet. By convention, the name starts with the @ character.

ARGS is a list of arguments. In addition to whatever arguments you wish to include, you may take advantage of built-in, pre-defined keyword arguments which control the display of ViSta's DeskTop, XLispStat and (at startup only) Logo Window. These arguments are: Hide-Desktop, Hide-Logo, and Hide-XLispStat (and their anti-aliases Show-DeskTop, Show-Logo and Show-XLispStat). The defaults of these arguments are the values of the global system variables \*hide-desktop\*, \*hide-logo\*, and \*hide-xlispstat\*. The default values for these global variables cause the DeskTop and Logo windows to be seen at startup, while the XLispStat window is not shown.

FORMS is any series of Lisp statements.

**ENHANCING VISTA - Enhancing ViSta - Writing Plots (IV.12.5)**

## Using ViSta's PLOT objects and PLOTS menu

ViSta's PLOTS menu is used to generate individual plots. The nature of each plot is discussed in the help information for the individual plot itself.

The plots are generated by a function called the plot constructor function. You can also generate the plot by typing the function into the listener. Whereas the item in the PLOT menu generates a default plot, you can gain a lot of control over the nature of each plot by typing the plot's constructor function. The details of how to do this

are discussed below.

The name of each of the various plot constructor functions is identical to the menu-item's name, except that spaces in the menu-item name are replaced by dashes in the constructor function. That is, the "Scatter Plot" menu item corresponds to the SCATTER-PLOT constructor function.

Each constructor function may have an optional DATA argument and may have several KEYWORD arguments. If no arguments are used, a default plot is constructed which is the same as the plot generated by the PLOTS menu's menu item. This default is constructed from information in \$, the current data. The information that is used is the first few variables in the list of all variables. Only the appropriate variable types are used. The number of vectors that are used depends on the details of the plot, as does the definition of the appropriate variable types. The default plot will include variable-labels and point-labels (or way-labels and level-labels), and will have a button-bar, a pop-up menu, a title and legends. The plot will appear in its' own window (i.e., not inside a container window), although the menu system may override this default.

All plots share the following arguments and syntax:

```
(PLOT-CONSTRUCTOR-NAME
 &optional DATA
 &key KEYWORD-ARGUMENTS)
```

#### ABOUT THE DATA ARGUMENT

The optional DATA argument may be omitted or may be NIL, or it may be a data object, a list of numeric or string elements, a list of variable objects, a list of equal-length lists, a vector, a list of equal-length vectors, or a matrix. If omitted or NIL, DATA is assumed to be \$, the current dataobject. In all cases, DATA is converted into a list of equal-length vectors. one vector for each of the OK-VAR-TYPES (see below) variables in the referenced data object.

The plot is constructed from the first few vectors in the list of vectors, the number of vectors depending on the details of the plot. If you need to use any of the keyword arguments, the data must be specified, and it must be mentioned before the keywords.

#### ABOUT THE KEYWORD ARGUMENTS

The keyword arguments consist of STANDARD keyword arguments (which are used by all of the plots and which are discussed here) and UNIQUE keyword arguments (which are unique to each plot and which are discussed in the help for each plot). If you need to use any of the keyword arguments, the data must be specified (perhaps as nil or \$) before the keywords are specified.

The standard keyword arguments (with defaults in parentheses, if appropriate) are:

ok-var-types a list of symbols or strings  
of usable variable types

variable-labels a list of strings, one for each variable

point-labels a list of strings, one per point

in (unused) T, NIL, (UNUSED) or CONTAINER -  
sets the container window.

linked (T) T or NIL for linked with other plots  
via stat object (not for comparison-plot,  
mosaic-plot or bar-graph)

connect (NIL) T or NIL for connecting points with lines  
(not for histogram-plot, mosaic-plot or bar-graph)

show (T) T or NIL controls if plot shown when created

top-most (T) T or NIL sets whether plots are on top

location     a list (x y) locating the plot's upper-left corner

size         a list (w h) of the plot's width & height

title        a string shown in the graph window title bar

legend1     a string for the first line of the legend

legend2     a string for the second line of the legend

content-only  T or NIL sets if only plot content shown

#### VARIABLE-LABELS and POINT-LABELS

Variables and points can be labeled with the VARIABLE-LABELS and POINT-LABELS keywords. (bar-graphs and mosaic plots use WAY-LABELS and LEVEL-LABELS). Each argument is followed by a list of the appropriate number of strings.

#### IN

By default, each plot will appear as an independent window (i.e., not inside a container window) when it is created. If :IN is used, the new graphic will appear inside a container window. If a menu item is used to create the graphic, IN will be set by the menuing system to direct the graph to the appropriate container. The container will be:

- 1: CONTAINER if IN is CONTAINER
- 2: XLISPSTAT if IN is NIL
- 3: \*ACTIVE-CONTAINER\* if IN and \*ACTIVE-CONTAINER\* are T

Note that for case 3 it can be tricky to determine where the graph appeared if the container is not visible.

#### SHOW, LOCATION, SIZE, TOP-MOST

The plot will not be shown when :SHOW is NIL until it is sent the message :SHOW-WINDOW. When the window appears it will be at LOCATION (which is relative to the window's container, if there is one) and be of size SIZE. The graphic will be :TOP-MOST by default. Note that the graphic will not be seen when it appears if its containing window is closed, minimized, located off-screen, or obscured by another window.

#### TITLE, MENU, LEGEND1, LEGEND2 and CONTENT-ONLY

Each plot can have a menu, title, and two legends. When :CONTENT-ONLY is T there will be no overlays, buttons, legends, etc., which is useful if the graph is part of a large spreadplot.

#### ABOUT THE MENUS

All old-style graphics and all new-style plots now use the message :make-two-plot-menus, to invoke a graph-PROTO method which creates the several plots menus and the do-click method that causes the menus to appear. This message and its associated method are invoked by a rewritten version of Luke's :new-menu message/method (see page 290 of Tierney). Since :new-menu is invoked by every graph-PROTO's :isnew method, so is :make-two-plot-menus.

Note that the :make-two-plot-menus method really only constructs one menu, since a window can only have one menu. It looks like two menus are made, but really it is one menu which has two sets of items, one presented as a popup to a right-click, the other as a pull-down from the menu-hotspot. The do-click method determines which set of items should be presented. The code for :make-two-plot-menus and the revised code for :new-menu is in runtime\graphic1.lsp.

The :make-two-plot-menus method makes two sets of menu items. The selection of menu items in the set is determined by the :menu-template and :popup-menu-template methods, which return a selection of menu template symbols. You can change the selection of menu items by modifying a plot's :menu-template and :popup-menu-template methods, as outlined by Luke. You can also use the :make-two-plot-menus message itself. For example, scatterplots might have the following use of the message:

```
(send self :make-two-plot-menus
  "Scatter"
  :hotspot-items '(help dash new-x new-y dash link dash
```

```

    on-top maximize dash print save copy))
:popup-items '(showing-labels mouse resize-brush dash
    erase-selection focus-on-selection view-selection
    select-all show-all dash
    symbol color dash
    selection slicer))

```

The hotspot-items and popup-items lists are lists of symbols which are defined in accordance with Tierney's discussion of menu templates. You can get a list of the hotspot menu items defined for a graph by typing (send graph :menu-template). The popup-items for a graph can be listed by typing (send graph :popup-menu-template). You can get a list of the entire set of items which I have defined up to this point by typing (menu-symbols). This list is not updated if you add more. Not all items work. The code is in file runtime\graphics0.lsp.

To add items to the menus, you should add symbol definitions to the make-menu-template method near the end of the runtime\graphics1.lsp file. It is also a good idea to add the symbol to the (menu-symbols) function nearby. Of course, you also have to define the method or function used by the item. Then you have to add the symbol to the appropriate template as summarized below. You have to do for each plot or graphic that is to have the new item.

For the old-style graphics, the message appears in the :isnew method (or sometimes in another method used by the :isnew method) for each graph or in a specifically mentioned template. The message's location, for each old-style graph is given here:

#### GRAPH CONSTRUCTOR FUNCTIONS WHERE TO ADD MENU ITEMS

```

histogram
histofreq          histofreq-proto :isnew in histfrq1.lsp
quantile-plot      quantile-plot-proto :isnew in qplotobj.lsp
normal-probability-plot  quantile-plot-proto :isnew in qplotobj.lsp
quantile-quantile-plot  quantile-quantile-plot-proto :isnew in qplotobj.lsp
plot-points        vista-scatterplot-proto :revised-look-and-feel and
                  scatterplot-proto :vista-look-and-feel in scatter1.lsp
spin-plot          spinplot-proto :isnew in spinplot.lsp
scatterplot-matrix  scatmat-proto :menu-template in scatmat.lsp

```

mosaic-plot

For the new-style plots, the message appears in the constructor function, overriding the old-style graphics usage of the method to reflect the added features of the new plots. The message's location, for each plot, is given here:)

#### PLOT CONSTRUCTOR FUNCTIONS WHERE TO ADD MENU ITEMS

```

histogram-plot
distribution-plot  distribution-plot function in plots002.lsp
cumulative-plot  cumulative-plot function in plots002.lsp
comparison-plot  comparison-plot function in plots002.lsp
dot-plot         dot-plot function in plots001.lsp
scatter-plot
all-scatter-plots  all-scatter-plots function in plots001.lsp
spinning-plot     spinning-plot function in plots001.lsp
orbit-plot        orbit-plot function in plots001.lsp
line-plot         line-plot function in plots001.lsp
box-plot          box-or-diamond-plot function in plots004.lsp
diamond-plot      box-or-diamond-plot function in plots004.lsp
subgroups-plot    subgroups-plot function in plots004.lsp
parallel-coords-plot  box-or-diamond-plot function in plots004.lsp
bar-graph         bar-graph-new function in plots003.lsp
mosaic-plot       mosaic-plot-new function in plots003.lsp

```



**ENHANCING VISTA - Enhancing ViSta - Writing Spreadplots (IV.12.6)**

The SPREADPLOT function creates and returns a spreadplot. The arguments are:

Required argument: PLOTMATRIX

Keyword arguments (with default values):

(CONTAINER \*SPREADPLOT-CONTAINER\*)

(SPAN-RIGHT NIL)

(SPAN-DOWN NIL)

(REL-WIDTHS NIL)

(REL-HEIGHTS NIL)

(LOCAL-LINKS T)

(STATISTICAL-OBJECT NIL)

(MENU-TITLE "SpreadPlot")

(SHOW NIL)

(SIZE (EFFECTIVE-SCREEN-SIZE))

(SUPPLEMENTAL-PLOT NIL)

PLOTMATRIX is a matrix of plot or dialog box objects (called plot-objects). The plot-objects in PLOTMATRIX are assumed to already exist and to have all been created while CONTAINER was enabled (all must have the same container, which must be CONTAINER). SPREADPLOT then arranges these plots inside CONTAINER in a rectangular array according to the details described below.

Usually there is one object per matrix cell, in which case the objects are laid out adjacent to each other, one per cell, in a rectangular array. There may be more than one object per matrix cell. When there is, the objects in the cell are laid out on top of each other, the last one being shown initially. To allow for cell-spanning (see below) there may be no object in a matrix cell (i.e., the matrix cell may be nil). The first row and column cannot contain nil values.

SPAN-RIGHT and SPAN-DOWN are each a matrix whose elements specify whether a plot occupies more than a single plotcell within a row or column. Each entry must be a non-negative integer indicating how many cells are spanned, counting from the current plotcell. Plotcells which are nil (are spanned and have no plot) have a span value of zero.

Plots are all same width and height unless REL-WIDTHS or REL-HEIGHTS are used (REL-WIDTHS and REL-HEIGHTS must be a list of nr or nc values, respectively, where nr=number of spreadplot rows and nc=number of columns). Column widths are proportional to the REL-WIDTHS values, row-heights to the REL-HEIGHTS values.

The spreadplot is contained within CONTAINER whose size is not greater than SIZE. Due to aspect-ratio constraints the size may be smaller than SIZE. By default, SIZE is equal to the usable portion of the screen (the part not occupied by MS-Windows toolbars). This size is calculated by the (EFFECTIVE-SCREEN-SIZE) function. The plots are shown if SHOW is T (the default is NIL).

SUPPLEMENTAL-PLOT is a plot or dialog box object. SUPPLEMENTAL-PLOT is located adjacent to the upper right edge of the PLOTMATRIX. The spreadplot cells consist of objects identified in PLOTMATRIX, plus, if not nil, SUPPLEMENTAL-PLOT. Note that containers and the supplemental plot do not yield an aesthetically pleasing result!

When LOCAL-LINKS is T (the default value), the spreadplot plots can be linked only to themselves, and not to other plots. They can be linked to other plots otherwise (even hidden plots).

STATISTICAL-OBJECT is an optional object identification of the model or data object that is creating the spreadplot (nil, the default, when there is none).

MENU-TITLE specifies the title of the spreadplot menu (nil for no menu).

Here is the general idea for how spreadplot messaging works:

When a plot in a spreadplot experiences some change, it sends a message to the spreadplot about the details of the change. Then, if the message needs to be processed by the spreadplot's statistical object (i.e., its model, transf, or data object) the spreadplot in turn relays this same message to the statistical object so that it knows that

a change has been made and what the change is. Then the statistical object processes the message and sends the results to the spreadplot. Then the spreadplot broadcasts either the original (when the statistical object isn't used) or the results of the processed original message (when the statistical object was used) to every plot in the spreadplot.

The specific messages that do this, and the sequence of actions, is as follows:

1) An individual plot begins the process by sending the  
`:UPDATE-SPREADPLOT (i j &rest args &key (use-statobj nil))`  
 message to the spreadplot. If USE-STATOBJ is T then the message is relayed to the statistical object which processes it and sends the results back via another UPDATE-SPREADPLOT message in which USE-STATOBJ must be NIL, and which must have the same values for values I and J. When USE-STATOBJ is NIL, the message is broadcast via the update-plotcell message to all plots so that they can update themselves. The arguments I and J are two numbers which identify the message uniquely, usually the row and column of the sending plotcell, but if a plotcell can send more than one type of message than another identifying method must be used. All of the REST arguments must be a union of arguments needed by all receiving plotcells, each of which must decide which arguments it needs.

Steps 2 and 3 are skipped unless USE-STATOBJ is T

2) When USE-STATOBJ was T in the preceeding step, the spreadplot object sends  
`:UPDATE-STATOBJ (I J &REST ARGS)`  
 to STATOBJ with the same arguments as were sent to the spreadplot (except for USE-STATOBJ). The particular statistical object needs to have an `:update-statobj` method written.

3) The statistical object sends the  
`:UPDATE-SPREADPLOT (i j &rest args)`  
 to the speadplot. Here I and J must be the same as were received by the statistical object, but args has changed. USE-STATOBJ must not be used.

4) The spreadplot object broadcasts the  
`:UPDATE-PLOTCELL (i j &rest args)`  
 message to all of its plots. Each plot that needs to respond to the message must have a unique UPDATE-PLOTCELL message written. They need to know how to respond to only certain I-J messages, not all. Args is either what was sent by the originating plotcell or the statistical object. Near the beginning of the sprdplot.lsp file is a dummy `:update-plotcell` method which is defined for all graphs and dialogs and which does nothing except ignore any messages that are sent."

## ENHANCING VISTA - Enhancing ViSta - Writing Plugins (IV.12.7)

### ON WRITING A VISTA PLUGIN

The ViSta plugin interface consists of two short pieces of code.

a) One piece is the **PLUGIN LOADER FUNCTION**, which comprises the entire executable contents of the **PLUGIN LOADER FILE**. The loader file is located in ViSta's **PLUGINS** directory. This is the file that interfaces your plugin's code with ViSta and its plugin system. All files in the **PLUGINS** directory are automatically loaded in by ViSta, and all these files must be **PLUGIN LOADER** files only.

b) The other piece is the **plugin PLUGIN CONSTRUCTOR FUNCTION**. This function is an even smaller piece of code which comprises a part of the executable contents of the **PLUGIN CONSTRUCTOR FILE**. This function is the main entry point to the rest of the code. The constructor file must contain, in the stated order:

- 1 >> your plugin's constructor function
- 2 >> your plugin's defproto statement
- 3 >> your plugin's isnew method
- 3 >> load functions that load additional code
- 3 >> additional code if appropriate

The order of the last three is not important, but they must all follow the defproto.

The main body of your plugin's code resides in a subdirectory of the PLUGIN directory. The code in a subdirectory is not automatically loaded by ViSta: You have to write the code to make it load.

## EXAMPLES OF PLUGINS

It is recommended that you read the code for the frequency analysis plugin, since it is simple and short and since it is used to illustrate the plugin interface. The Log Linear analysis, Homogeneity analysis and both MDS plugins are also written according to the rules given here, and serve as more complex examples. Other plugins are written according to older, more complex rules and should not be used as models of how to write plugins.

## CODE RESTRICTION

Your code must not contain any symbols which redefine symbols in ViSta. It is best if your code consists entirely of object-oriented code. In particular, avoid the use of DEFUN. If you must use DEFUN, make sure first that there isn't already a function with the same name.

---

STEP 1: DEFINE THE PLUGIN SYSTEM VARIABLES  
AND PREPARE VISTA FOR YOUR PLUGIN

---

The following seven plugin system variables **MUST** be local variables that are bound as described:

1) **PLUGIN-SUBDIRECTORY** - A string of up to 8 characters specifying the name of the subdirectory containing the plugin code and the name of the file that contains the **PLUGIN CONSTRUCTOR** function. The filename must simultaneously follow the rules of directory naming for the MSDOS, MACINTOSH and UNIX operating systems.

2) **PLUGIN-FILE** - A string of up to 8 characters specifying the name of the file that contains the **PLUGIN CONSTRUCTOR** function. The filename must simultaneously follow the rules of directory naming for the MSDOS, MACINTOSH and UNIX operating systems. ViSta uses the **PLUGIN-FILE** to construct the global system variable \*NAME-plugin-path\*, where NAME is replaced with the **TOOLBAR-BUTTON** specified below.

The **PLUGIN-FILE** file **MUST** be located in the **PLUGIN-SUBDIRECTORY** subdirectory of the plugin directory. The file must contain the plugin's constructor function (which replaces the constructor loader function) and whatever other code is needed by the plugin. If all the code doesn't fit in one file, the plugin-constructor-file must load in the additional files that are needed.

3) **MENU-ITEM** - Specifies the plugin's menu item name (a string of up to about 20 characters).

4) **TOOLBAR-BUTTON** - Specifies the name of the plugin's button on the workmap's toolbar (a string of 5 or 6 characters).

5) **WORKMAP-ICON-PREFIX** - Specifies the plugin icon's name on the workmap (a string of 3 characters)

6) **OK-VARIABLE-TYPES** - Specifies the data types that are used by the plugin. The possible ok-variable-types are "numeric" and "category". The "ordinal" type is not supported.

7) **OK-DATA-TYPES** - Specifies the variable types that are used by the plugin (a string list). The eleven datatypes are divided into:

- a) a datatype for data with missing values:  
"missing"
- b) a datatype where the basic datum is a relation:  
"matrix"
- c) six datatypes where the basic datum is a quantity:  
"univariate" "bivariate" "multivariate"

```
"category" "class" "general"
```

d) three datatypes where the basic datum is a frequency:

```
"frequency" "freqclass" "crosstabs"
```

The message (send \$ :determine-data-type) determines and returns the datatype of \$, the current data. More information about datatypes is available from the help topics."

These seven variables are then used to prepare ViSta's plugin environment for your plugin. The following statement defines the variables and prepares the environment for the frequency analysis module:

```
(let* ((plugin-subdirectory "freq")
      (plugin-file "freqmain.lsp")
      (menu-item "Frequency Analysis")
      (toolbar-button "Freqs")
      (workmap-icon "Frq")
      (data-types '("class" "category"
                  "freq" "freqclass"
                  "crosstabs" "general"))
      (variable-types '(numeric category)))
      )
```

```
(send *vista* :prepare-plugin-environment
      plugin-subdirectory
      plugin-file
      menu-item
      toolbar-button
      workmap-icon
      data-types
      variable-types ))
```

ViSta uses PLUGIN-SUBDIRECTORY, PLUGIN-FILE and TOOLBAR -BUTTON to construct the global system variables \*NAME-plugin-creator-file\* and \*NAME-plugin-path\*, where NAME is replaced with TOOLBAR-BUTTON.

For example, assume that (where = means "bound to")

```
PLUGIN-SUBDIRECTORY = "freq",
```

```
PLUGIN-FILE = "freqmain.lsp", and
```

```
*PLUGIN-PATH* = "C:\Program Files\VisualStats\ViSta7\"
```

then ViSta creates the following two global variables and binds them as shown:

```
*FRQNCY-PLUGIN-PATH* = "C:\Program Files\VisualStats\ViSta7\plugins\freq\"
```

```
*FRQNCY-PLUGIN-CONSTRUCTOR-FILE* = "C:\Program Files\VisualStats\ViSta7\plugins\freq\freqmain.lsp"
```

---

ABOUT THE PLUGIN'S CONSTRUCTOR AND LOADER FUNCTIONS

---

You must write two functions. These two functions must have identical names and arguments. but they will have different bodies. One of these functions is loaded when ViSta is run, while the other is loaded the first time the user uses the plugin.

**PLUGIN LOADER FUNCTION** - The function that is loaded when ViSta is run is called the **PLUGIN LOADER FUNCTION** since it's primary purpose is to load the plugin's code the first time the user needs to use the plugin. The loader function is a small piece of code written according to rules specified below. It is contained in your plugin interface file (the file you place in the PLUGINS directory).

## PLUGIN-CONSTRUCTOR-FUNCTION

This function is the entry point to your plugin's code. The function issues the :ISNEW message which in turn constructs a new instance of the plugin, and then uses the new instance to do the analysis called for. The constructor function must be located in the PLUGIN CONSTRUCTOR file discussed earlier. The constructor file must also contain code to load in any additional code files.

The plugin system assumes that when the user selects an analysis menu item that there is a model constructor function which is the same as the menu item's name (but with spaces replaced by dashes).

Thus, the names of the two functions must not only be identical, but they must also each be identical to MENU-ITEM specified above (with spaces in menu-item replaced by dashes in the function names).

Because the loader and constructor functions have the same name, and because the code read in by the loader function contains the constructor function, the loader function modifies itself to become the constructor function (i.e., the loader function is self-modifying). The architecture of the constructor function is as it is so that code to do the analysis is not loaded until needed. Note, however, that the menu item, tool and model prefix are automatically loaded at startup.

Here are the rules for writing the loader and constructor function.

- (a) The loader and constructor functions MUST have the same names.
- (b) The loader/constructor name must be the same as the plugin's item name in the analysis menu (with menu-item spaces replaced by function-name dashes).
- (c) The loader and constructor functions MUST have the same arguments. The arguments must be keyword arguments.
- (d) The loader and constructor functions MUST have the following two keyword arguments:
  - DATA a symbol which, by default, is bound to \*current-data\*
  - DIALOG a logical bound whose value must be NIL, even if the analysis dialog is to shown
- (e) The loader and the constructor function MAY have any number of additional keyword arguments as is needed by your analysis.
- (f) The entire set of required and additional arguments must be the same for the loader and constructor functions.

---

## STEP 2: DEFINE THE PLUGIN'S LOADER FUNCTION

---

The loader function is located in the PLUGIN LOADER FILE which is itself located in the PLUGINS directory. Its filename must follow the usual naming conventions. The function MUST take these following three actions. The actions MUST take place in the order specified:

- (a) display a copyright notice (optional, of course)
- (b) load code containing the constructor function
- (c) invoke the constructor function.

Here is the plugin-loader-function for the frequency analysis module:

```
(defun frequency-analysis
  (&key
   (table-variables nil)
   (control-variables nil)
   (data *current-data*)
   (dialog nil))
  "Args: &key table-variables control-variables data title name dialog
```

ViSta Frequency Analysis Plugin to calculate chi-square and related statistics for n-way frequency data. The analysis is based on the frequency array for the :TABLE -VARIABLES (a variable name string or a list of variable name strings). By default, the first two (or one) ways of the frequency array become the table variables. Additional ways are treated as the control variables, unless they are explicitly specified as :CONTROL-VARIABLES (a variable name string or a list of variable name strings). The remaining keywords have their usual meaning."

```
(format t "; CopyRt: FREQ Copyright (c) 1998-2002, by Forrest W. Young & Dominic Moore~%> ")
(load *freqs-plugin-creator-file*)
(frequency-analysis
 :table-variables table-variables
 :control-variables control-variables
 :data data
 :dialog dialog))
```

---

### STEP 3: DEFINE THE PLUGIN'S CONSTRUCTOR FUNCTION

---

The constructor function must be located in the file specified by \*PLUGIN-FILE\* which must be located in the subdirectory specified by PLUGIN-SUBDIRECTORY, The function MUST have name and arguments that are identical to those of the loader function. The function does just one thing: It invokes the plugin's NEW method.

Here is the plugin-creator function for frequency analysis:

```
(defun frequency-analysis
 (&key
 (table-variables nil)
 (control-variables nil)
 (data *current-data*)
 (dialog nil))
 "Args: &key table-variables control-variables data title dialog
```

ViSta Frequency Analysis Plugin to calculate chi-square and related statistics for n-way frequency data. The analysis is based on the frequency array for the :TABLE -VARIABLES (a variable name string or a list of variable name strings). By default, the first two (or one) ways of the frequency array become the table variables. Additional ways are treated as the control variables, unless they are explicitly specified as :CONTROL-VARIABLES (a variable name string or a list of variable name strings). The remaining keywords have their usual meaning."

```
(send freq-plugin-object-proto :new "Frequency Analysis"
 data dialog table-variables control-variables))
```

---

### STEP 4: DEFINE YOUR PLUGIN'S PROTOTYPE OBJECT

---

Your plugin's defproto function must define your proto in such a way as that it inherits from vista-analysis-plugin-object-proto. Here is the frequency analysis defproto statement:

```
(defproto freq-plugin-object-proto
 '(table-vars control-vars freq-var nways chisq
 phi binomial cmh
 table-labels control-labels
 observed-data-matrices expected-data-matrices
 matrix-index-list freqclass-data-matrix)
 ()
 vista-analysis-plugin-object-proto)
```

---

 STEP 5: DEFINE YOUR PLUGIN'S :ISNEW METHOD
 

---

The plugin constructor function issues the :NEW message. This message invokes the plugin's isnew method which proceeds to create a new instance of the plugin object.

Here are the rules for writing the plugin's :new message and its corresponding :isnew method:

- (a) The new message and the :isnew method MUST have at least three arguments. The first three arguments must be, in order: (a) A string which is identical to MENU-ITEM. (b) DATA; and (c) DIALOG
- (b) You can follow the initial three arguments with whatever arguments are required for your plugin.
- (c) These additional arguments must be processed (or stored for later processing) within the isnew method.
- (d) A CALL-NEXT-METHOD function must appear at the end of the constructor function, It must have exactly three arguments: MENU-ITEM, DATA, and DIALOG, in that order.

Here is the isnew method for frequency analysis:

```
(defmeth freq-plugin-object-proto :isnew
  (title data dialog table-variables control-variables)
  (unless (equal data *current-data*)(setcd data))
  (unless (send data :array)
    (error-message "Wrong kind of data for frequency analysis."))
  (let* ((vars (send data :active-array-variables))
        (nvars (length vars))
        (ntvars (min nvars 2))
        (1col (and (= 2 (length (array-dimensions (send data :data-array))))
                   (= 1 (second (array-dimensions (send data :data-array))))))
        )
    (flet ((check-vars (check-variables vars type)
            (when (stringp check-variables)
              (setf check-variables (list check-variables)))
            (when (not (listp check-variables))
              (fatal-message "~a variables must be specified as either a variable name string or as a list of one or
two variable name strings." type))
            (mapcar #'(lambda (var)
                        (if (not (member var vars))
                            (fatal-message "The variable ~s, specified as a ~a variable, is not found in these data." type))
                        check-variables))))
        (cond
         (table-variables
          (check-vars table-variables vars "Table")
          (when (length (> table-variables 2))
            (fatal-message "Too many table variables. Maximum is 2.)))
         (t (setf table-variables (select vars (iseq ntvars))))))
    (if control-variables
        (check-vars control-variables vars "Control")
        (when (> ntvars nvars)
          (setf control-variables (select vars (iseq ntvars (1- nvars))))))
    (send self :table-vars table-variables)
    (send self :control-vars control-variables)
    (send self :freq-var (send self :freq-var? data))

    (call-next-method title data dialog)
  )))
```



## ENHANCING VISTA - Enhancing ViSta - Writing Updates (IV.12.8)

### UPDATING VISTA

You can place files in a subdirectory of the update directory to fix bugs and update features.

The files that you place in this subdirectory are not automatically loaded by ViSta. Rather, you must include them in a load statement which is included in one of the five files in the update directory that are automatically loaded by ViSta during the startup process. These files are called `phase0.lsp` `phase1.lsp` `phase2.lsp` `phase3.lsp` `phase4.lsp`. These five files are used to determine when the files that you have placed in the update directory are loaded during the startup process. You must include the name of each file you have placed in the update directory in the load statement in one and only one of the `phase*.lsp` files.

The startup phases are as follows:

#### PHASE 0 - START

These files are used to change aspects of the system used before the initialization. You will need to use this phase for changes to verbose system (file `verbose.lsp`), changes to the initialization system (files `vista.lsp`, `vista1.lsp` and `vista2.lsp`) and for changes you include in other phases which should but don't take effect because the to-be-changed version has already loaded before phases 1, 2, 3 or 4.

#### PHASE 1 - INITIALIZATION

These files are used to change the initialization. Often, these files are changes to `lspsrc` or `maketime` files. They are loaded before anything else.

#### PHASE 2 - RUNTIME

These files are loaded immediately after the runtime files are loaded, but before anything is done to create the initial user environment. These files are, essentially, additional runtime files. Do not use these files to override the distributed runtime files. If you wish to modify the runtime files you should make the modifications and place the modified files in the runtime directory, keeping the original filenames.

#### PHASE 3 - DESKTOP GUI

These files are used to alter the desktop after it is created. Often, these files are changes to source files. They are loaded after the desktop has been created.

#### PHASE 4 - SOURCE

These files are used to alter the workspace. Often, these files are new features. They are loaded after initial display is completely prepared but before it is seen and before any `-f` or `dde` files are read.

#### FILENAMES

Please use this naming convention for files in the update directory (where `xxx` is the original file name, when possible, and `-mm-dd-yy` is month-day-year of fix)

```
xxx-slt-mm-dd-yy.lsp  slot changes
xxx-fix-mm-dd-yy.lsp  bug fixes
xxx-bet-mm-dd-yy.lsp  betterments
xxx-new-mm-dd-yy.lsp  new features
xxx-mod-mm-dd-yy.lsp  modifications
```

#### SUBDIRECTORY NAME

Please name the subdirectory to identify the date of release of the changes. You must create a variable `*update-subpath*` whose value is a string which names the subdirectory. Do this with the following statement, substituting your subdirectory name (note use of back-quote and comma):

```
(setf *update-subpath*
  (make-pathname
    :directory `(:relative ,*update-path* "Sep17-2002"))))
```

## LOADING FILES

You should use the following statements to load files from the update subdirectory, substituting the appropriate filenames. Please include a comment explaining the update.

```
(mapcar
 #'(lambda (filename)
   (load (strcat *update-subpath* filename)))
 (list "datasmry-bet-09-03-02.lsp" ;changes about-these-data
       "datatype-fix-09-03-02.lsp" ;remove debugging message
       ))
```

**ENHANCING VISTA - Enhancing ViSta - About DataTypes (IV.12.9)****WHY HAVE DATATYPES?**

Each ViSta data object has a datatype. The datatype determines ViSta's default analyses and visualizations, and limits the choice of actions you can take to those that are likely to be reasonable.

Datatypes are meant to simplify the user's experience... they allow ViSta to make a more educated guess about what should be done with the data, and provide an unobtrusive way of guiding the user by limiting choices.

To determine the datatype of the current dataobject, type:

(datatype?)

For more information on the details of datatypeing the current dataobject, type

(current-datatype)

in the listener.

**WHAT ARE THE DATATYPES?**

The datatype depends on variable types and on whether the data are frequencies, are relational, or contain missing values (data are frequencies if explicitly declared so in the datacode, or if all numeric variables are named "Freq").

The 12 data types recognized by ViSta are:

**CATEGORY** - There are only category variables

**UNIVARIATE** - There is one variable and it is non-frequency numeric

**BIVARIATE** - There are two variables and they are both non-frequency numeric

**MULTIVARIATE** - There are more than two variables. All are non-frequency numeric

**CLASSIFICATION** - There is exactly 1 non-frequency numeric variable and there are 1 or more category variables.

**FREQUENCY** - All the variables are numeric frequency variables

**FREQCLASS** - There is exactly 1 numeric frequency variable and there are 1 or more category variables

**CROSSTABS** - There are 1 or more numeric frequency variables 1 and or more category variables

**GENERAL** - When none of the above definitions is satisfied the datatype is "general".

**MISSING** - The data have one or more NIL elements

**MATRIX** - The observations and variables refer to the same things, the data elements are relational, specifying the relation (correlation, distance, covariance) between pairs of the things.

**NEW** - Data created by the NEW DATA menu item, and not yet saved as a dataobject.

**DEFINITION OF DATATYPES:**

Somewhat more formally, the datatypes are defined to be:

A) NEW for data which have just be generated by NEW DATA menu item.

B) MISSING if the data contain one or more NIL elements.

C) MATRIX if matrices are present without NIL elements.

D) If none of the above is true, then the datatype is defined according to the number of category and numeric variables in the data, and by whether the data are frequencies or not. Specifically:

Number of Category Variables	 [freq?]	Number of Numeric Variables			
		0	1	2	>2
0	nil	error	univariate	bivariate	multivariate
	t	error	freq	freq	freq
>0	nil	category	class	general	general
	t	category	freqclass	crosstabs	crosstabs

## NOTES:

- TABLE datatype is no longer used.
- ORDINAL variables are treated as NUMERIC.
- Unless you specify otherwise, by default the datatype is defined on ALL variables, NOT active variables. This makes the datatype a characteristic of the data, not the active data.
- In addition, datatype may be temporarily set to "EnAbl" "DisAbl" "ReEnAbl".

## USEFUL DATATYPE FUNCTIONS AND MESSAGES:

To determine the (generalized) datatype of the current data object, type:

```
(datatype?)
```

To determine the (generalized) datatype of a data object DOB, type:

```
(datatype? DOB)
```

These functions use the following function which may be used is no data object:

```
(datatype types freq new missing matrix)
```

where types is a list of variable types, and freq, new, missing, and matrix are logical variables indicating whether the data are frequencies, new, have missing values, or are matrix data.

The full set of possible datatypes, etc, are given by

```
(possible-datatypes)
```

```
(possible-data-extensions)
```

```
(possible-data-abbreviations)
```

To see if data satisfy a particular datatype, send a message of the form

```
(send $ :datatype?)
```

where "datatype" is replaced by the datatype name for each datatype. E.g.:

```
(send $ :freq?)
```

```
(send $ :crosstabs?)
```

```
etc.
```

For a help message about the datatype aspects of the current dataobject, type

```
(current-datatype)
```

**ENHANCING VISTA - Enhancing ViSta - About FileTypes (IV.12.10)****ABOUT FILE EXTENSIONS AND FILETYPES**

The FILE EXTENSION of a file is specified by the three letters at the end of the name of the file. ViSta recognizes the .VDF, .VAF, .VIS, .LSP, .WKS, and .FSL file extensions.

All of the file extensions given above are associated with LISPBOSS, an input file managing program within the ViSta system. Using the file EXTENSION, along with other information, LISPBOSS determines the FILETYPE of the file and then determines which of the two other programs within the ViSta system should process the file. These other two programs are VISTA, which is the visual statistics program, and LSPEDIT, which is a Lisp editor. (File extensions are all associated with LISPBOSS during installation by the filetype.bat file, which resides in the ViSta home directory.)

The FILETYPE of a file depends on the file extension. Sometimes it also depends on other information. The ViSta system distinguishes DATA, APPLETT, and PROGRAM files (it also distinguishes WORKSPACE and BYTECODE filetypes which are not considered here.)

**DATAFILE**

A datafile defines the data which are the focus of a statistical visualization and analysis. The file consists of a single DATA function which ViSta uses to define a data object. Nothing else is permitted to be in a datafile. The file can reside on the local client computer or can be served to the local client computer by a remote server or via the internet.

**APPLET FILE**

An applet file defines an applet. An applet is a small piece of code written in Lisp, ViSta's underlying programming language. Applets provide the user with a way to run ViSta and have it perform a specialized set of calculations specified by the applet. Applets can also cause an already running instance of ViSta to perform the stated calculations. Applets can cause ViSta to run without the standard ViSta logo and desktop windows being shown. The applet file consists of a single APPLETT function whose body contains the applet code. Nothing else can be in an applet file. The file can reside on the local client computer or can be served to the local client computer by a remote server or via the internet.

**PROGRAM FILE**

A program file is a file which contains Lisp programs. Program files are lisp-containing files which are recognized by ViSta and which are neither data files nor applet files. These files can be loaded by ViSta or edited by LspEdit depending on the value of \*edit-lisp-files\*. Any lisp code can be in a program file.

**RELATION BETWEEN FILE EXTENSION AND FILETYPE.**

A file with the .LSP file extension can be either a DATAFILE or a PROGRAM FILE. It is treated as being a DATAFILE if it is in a directory whose path contains a directory named DATA, DATALIBRARY, MYDATA or MY-DATA (capitalization ignored). If it is not in such a directory, it is still treated as being a DATAFILE if the environmental variable \*edit-lisp-files\* is NIL. Otherwise, a .LSP file is treated as being a PROGRAM FILE.

The remaining file extensions have a one-to-one mapping onto file types, as follows:

.VDF files are DATAFILES

.VAF files are APPLETT FILES.

.VIS files are PROGRAM FILES.

WORKSPACE and BYTECODE files.

These filetypes are created by the Lisp compiler system and are contained in .WKS and .FSL files respectively. These files need not concern us here.

**NO EXTENSION**

If there is no extension specified, it is assumed that the extension should be either .LSP or .FSL, and looks for the most recent of these two.



## APPENDIX: VISTA'S MENUS

VISTA'S MENUS - About The Menus - The Menu Bar (A.13.1)

ABOUT VISTA'S MENUBAR MENUS:

The menus on ViSta's menubar fall into three groups:

- File menus: FILE and EDIT
- Analysis menus: DATA, PLOTS, VIEWS, TRANSFORM, ANALYZE and MODEL
- Support menus: OPTIONS, HELP, DESKTOP and WINDOW

---

The File Menus  
FILE and EDIT menus

---

You use the FILE menu to get data into ViSta, to print results, and to output information from ViSta.

For inputing data into ViSta, the File menu has items to create NEW DATA; to OPEN DATA for analysis by ViSta; to SIMULATE DATA according to statistical models; and to IMPORT DATA from text files.

For printing results, the File menu has items to print files, the contents of a window, and the contents of a pane of a graphics window.

For outputing information, the File menu has items for exporting data and for saving data and models.

The EDIT menu is not directly focused on data analysis, but provides the usual copying, pasting, etc.

---

The Data Analysis and Visualization Menus  
DATA - PLOTS - VIEWS - TRANSFORM - ANALYZE - MODEL

---

The analysis menus provide you access to ViSta's statistical visualization and analysis capabilities. These six menus are the six major steps that you take to gain understanding of your data, and their arrangement order reflects the order in which these steps occurs when looking at your data. A typical data analysis is a cycle repeating these six steps, with the cycle spiralling in on a deeper understanding of the data.

**DATA:**

Use the DATA menu "to see what the data seem to say", to manipulate the data, and to get quick summary information, both visual and numeric, about them.

**PLOTS**

Use the PLOTS menu to obtain simple visual information about the data.

**VIEWS**

Use the VIEWS menu to get more advanced and complete visual information about the data.

**TRANSFORM:**

Use the TRANSFORM menu to transform the data (whether this is necessary depends on "what you see" when looking at your data).

**ANALYZE:**

Use the ANALYZE menu to create a model of what your "data seem to say".

**MODEL:**

Use the MODEL menu to look at the model resulting from the analysis (again, "to see what the data seem to say",

but this time we look at the view the model provides).

---

### The Support Menus OPTIONS - HELP - DESKTOP - WINDOW

---

The OPTIONS, HELP, DESKTOP and WINDOW provide support facilities that effect how ViSta interacts with you. Briefly, the OPTIONS menu allows you to alter ViSta's data analysis environment; the HELP menu provides help about using ViSta; the DESKTOP menu allows you to alter the way the desktop window works with you; and the WINDOW menu provide access to ViSta's other major windows. Each of these menus has help information which you can access through the HELP menu.

### APPENDIX: VISTA'S MENUS - About The Menus - Pop-Up Menus (A.13.2) DeskTop PopUp Menus

You can pop-up a menu by right-clicking the desktop or one of the desktop's icons. These menus give you access to nearly all of the capabilities that can be accessed via the menubar. The menu you get depends on exactly what you have clicked on, as is summarized below.

#### RIGHT-CLICKING A DATA ICON

Icon Body: Data Menu  
Icon Cap: About Menu  
Icon Buttons:  
Up Left: Report Menu  
Up Right: Visualization Menu  
Lo Left: Transformations Menu  
Lo Right: Analysis Menu

#### RIGHT-CLICKING A DATASHEET ICON

Icon Cap: About Menu  
Icon Body DataSheet Menu

#### RIGHT-CLICKING AN ANALYSIS ICON

Anywhere: Analysis Options  
(not implemented)

#### RIGHT-CLICKING A TRANSFORMATION ICON

Anywhere: Transformation Options  
(not implemented)

#### RIGHT-CLICKING A MODEL ICON

Main Body: Model Menu  
Cap: About Menu  
Icon Buttons:

Left: Report Menu  
Right: Visualization Menu\

#### RIGHT-CLICKING THE DESKTOP

Anywhere Desktop menu

#### RIGHT-CLICKING THE TOOLBAR

Anywhere Change the number and function of buttons.

#### APPENDIX: VISTA'S MENUS - The File Menu - The File Menu (A.14.1)

This topic presents you with the help information for the items of the File Menu. Choose an item to see its help.

#### APPENDIX: VISTA'S MENUS - The File Menu - New Data ... (A.14.2)

NEW DATA lets you enter new data into ViSta.

When you use NEW DATA, you will see a dialog box that asks you to specify the DataType of the data you wish to enter, and which suggests names for specific information that is needed for each datatype. See the VARIABLE AND DATA TYPES subtopic of the help panel's MANAGING DATA topic for further information about datatypes.

The three datatype choices are:

##### FREQUENCY TABLE DATA

These data have entirely NUMERIC variables whose values specify frequencies. The values must all be non-negative (or missing). The data may be a one-way classification of the frequencies of the row (or column) entities, or a two-way cross-classification (crosstabs) of the co-occurrence frequency of the row and column entities. These data may be converted into FREQUENCY CLASSIFICATION or CATEGORY data.

##### MATRIX DATA

These data have entirely NUMERIC variables whose values specify the strength of relation (correlation, covariance, co-occurrence probability, distance, etc.) between the row and column entities. The observation and variable entities must be the same. Each value species an element of a square matrix. There may be more than one matrix.

##### ALL OTHER TYPES

This choice (the default) is appropriate for all datatypes except the two choices above it. Note that this includes several specific datatypes other than the two specified. If you choose ALL OTHER TYPES, ViSta will determine which datatype is appropriate.

Depending on the DataType, ViSta suggests a name for the new data (and for the rows and columns of new frequency data). You may change these suggestions by typing a name in the appropriate place.

After you click OK, you will see a datasheet icon on the workmap, and a new datasheet. If this datasheet doesn't have enough rows and columns for your data, use the datasheet's popup menu to ADD ROWS AND/OR COLUMNS. See the ENTERING DATA topic for more information on using the datasheet.

**APPENDIX: VISTA'S MENUS - The File Menu - Open Data ... (A.14.3)**

OPEN DATA opens a datafile, loads it into ViSta, and displays it as a datasheet. It is the first step in a data analysis.

When you use OPEN DATA, ViSta will then open a file dialog, which you use in the usual way to find your data.

After you open your data you will see a datasheet of your data. You may use this datasheet to browse through your data. EDIT DATA lets you edit it.

When you are finished with the datasheet you should close it. You will then see that other ViSta windows have changed: The WorkMap will have a new Data Icon. The name of the Data Icon will correspond to the name of your data.

If you are using guidemaps, you are now ready to open your data. After you open your data the OPEN YOUR DATA guidemap will be replaced by the DATA ANALYSIS guidemap. Click on the !! side of the OPEN DATA button and use the file dialog to find your data.

**APPENDIX: VISTA'S MENUS - The File Menu - Simulate Data ... (A.14.4)**

SIMULATE DATA lets you generate new data by simulating the process of sampling from a population. It also includes a visualization that demonstrates the central limit theorem.

When you use SIMULATE DATA, you will see a dialog box that asks you to select a distribution. This corresponds to the theoretical population distribution from which samples will be drawn.

The dialog box also asks for sample size and number of samples, and whether or not you wish to visualize the sampling process. For certain distributions, an additional dialog box will ask for information needed to completely specify the sampling process.

The visualization shows the shape of the population and sample distributions as well as the distribution of sample means and standard deviations. If you ask for more than one sample, the last sample distribution is displayed along with all sample means and standard deviations. The visualization gives you the choice of generating additional samples and seeing the updated distributions of means and variances. Each time you click on the visualization's "New Sample" button ViSta generates "n" additional samples, where "n" is the number of samples specified in the dialog box.

The simulation process creates a new multivariate data object. It is represented by an icon on the workmap.

**APPENDIX: VISTA'S MENUS - The File Menu - Import Data ... (A.14.5)**

IMPORT DATA opens a file and attempts to import data from it. If the information in the file is formatted as explained below, ViSta will load the data and display a data icon on the workmap.

For a file to be an importable datafile, it must be a multiple line file, where each line has the same number of pieces of information, called elements. If there are N variables, every line must have N+1 elements. An element can be either a number, a symbol, or a string. A symbol is an unquoted group of keyboard characters containing no white space, while a string is a double-quoted group of characters which may contain white space. Symbols

are converted to upper case, strings retain original case. Missing data elements are represented by the symbol nil or NIL, but NOT by the string "nil" (regardless of case).

## TYPES OF LINES IN THE FILE

There are three types of lines which may appear in an importable datafile. These lines are called the NAMES, TYPES and DATA lines. While there is some flexibility (see below), it is recommended that the DATA lines be preceded by a NAMES line and a TYPES line, each of which can only have elements which are character strings. Files which do not have these lines do not follow the recommended structure and may be more time consuming to process, or may not be importable.

**NAMES:** The NAMES line, which is first, begins with a character string which specifies the name of the exporter and of the dataset. This string is a two-part string: The first part identifies the exporter and the second part names the file. The parts are separated by a colon. Thus, two examples are "ViSta:Cars" and "Excel:Fish". The line then continues with N more character strings which are used to define variable names.

**TYPES:** The TYPES line, which is second, defines the dataset type (first) and the types of each variable (the rest). The NAMES and TYPES lines can only have elements which are character strings. The NAMES can be any string of characters. The DATA TYPES can be one of the following: "category", "univariate", "bivariate", "multivariate", "classification", "frequency", "freqclass", "crosstabs", "general", "missing", "symmetric" or "asymmetric". The VARIABLE TYPES can be either "category", "ordinal", or "numeric". Capitalization is ignored for all data and variable types.

**DATA:** Each of the remaining lines of the file forms a row of the data. The first element of each row is a "label" which is used to identify a data observation (in the case of "symmetric", or "asymmetric" data the label identifies a matrix of data). These first element of a row may be a string, symbol or number, but since labels are strings, the first element of a data line is converted to a string. The remaining elements of each line are the data. A data element must be a number if its variable is "numeric", but can be a number, symbol or string if its variable is a "category" variable.

## THE MEANING OF NUMERIC ELEMENTS

Elements of numeric variables are numbers, and numbers carry different meanings in different situations. Depending on the data being imported, you may be asked to indicate what the numbers represent. The variety of meanings can be grouped together into three kinds of meanings, as follows:

**QUANTITY:** A number can represent the quantity or amount of some characteristic.

**FREQUENCY:** A number can also represent the frequency of something, or the count of something.

**ASSOCIATION:** A number can also tell of the degree of association or correlation between two things, or the distance between a pair of things.

## TYPES OF IMPORTABLE FILES

**VISTA EXPORT FILES:** Files which follow the recommended structure given above are called ViSta Export Files.

**ANONYMOUS EXPORT FILE:** These files follow the recommended structure given above, but the first character string in the file is a one-part string. It is assumed that the string names the dataset, and does not identify the exporting program.

**UNTYPED EXPORT FILE:** These files have only a NAMES line preceding the DATA lines. No TYPES line appears. When a file has only one initial line with elements which are all character strings is assumed to be an untyped export file. For such a file it is assumed that the first column of values (i.e., the first value on each line) is a LABEL. The data and variable TYPES are determined from the nature of the data (a variable is "numeric" if it only contains numbers, "category" otherwise). The dataset type is determined from the variable types.

PLAIN EXPORT FILE: These files only have DATA lines. There is no NAMES or TYPES line. If the initial record of the file contains values other than strings (i.e., contains at least one number or symbol) the file is assumed to be a "Plain" export file. That is, it is assumed that the initial name and type records are absent, and that the first record contains data elements for the first row of data. The first element of each line is assumed to be a LABEL. Default values are used for the variable names, and the variable types are determined from the data (a variable is "numeric" if it only contains numbers, "category" otherwise). The dataset type is determined from the variable types. The datafile name is used as the name for the dataset.

#### APPENDIX: VISTA'S MENUS - The File Menu - Export Data ... (A.14.6)

EXPORT DATA exports your ViSta data as a text file containing your data in a form that can be imported into other programs (or into ViSta).

When you export data you will see a standard SAVE dialog box. Use it to identify where you wish to save your data. The exported data specify that they have been exported from ViSta, and specify the data's name and type, the variable names and types and the observation labels. For matrix data the exported data also identify the matrix names and shapes.

If there are "N" active variables, then all lines in the file contain N+1 values. All of the values on the first two lines of the export file are character strings. Each character string is contained inside double quotes and is followed by a white space separator.

The first line of the file starts with a string which names the data and identifies ViSta as the exporter (by including "ViSta:" as the first part of the string). The line has N more strings which are used to define variable names.

The second line of the file begins with a string specifying the datatype and continues with N more strings which define the variable types.

Each succeeding line of the file begins with an observation label and is followed by the N data values for that observation. There are a total of 2 + NOBS records in the file, where NOBS is the number of observations.

For matrix data each of the data lines begins with a string which is the matrix name. These strings are the same for every row of a matrix, but change from one matrix to the next. The shape of the data are specified by the data type: The datatype "symmetric" means that all matrices are symmetric, whereas the datatype "asymmetric" means that at least one matrix is asymmetric. Row labels are not stated and are formed by concatenating the matrix name with the appropriate variable name.

#### APPENDIX: VISTA'S MENUS - The File Menu - Save Data As ... (A.14.7)

SAVE DATA AS ... saves your ViSta data as a file on your computer so that it can be used again after you finish using ViSta.

Your data can be saved as one of several different types of files, as indicated by the choice of file extensions.

#### VISTA DATA FILE (.VDF)

Normally you would save the data as a ViSta Data File (.VDF). This file is recognized by Windows as being a ViSta file, and when you double-click it within Windows it will run ViSta (if it isn't already running) and then appear on ViSta's workmap.

**VISTA PROGRAM FILE (.VIS)**

You can also save your data as a ViSta Program File (.VIS). This file can be edited to contain additional Lisp code to pre- or post-process the datafile. It will be loaded by ViSta for processing when it is double-clicked.

**LISP FILE (.LSP)**

If you save your data as a .LSP file it will be treated as a data file (just like a .VDF file) if it is saved into a directory structure containing at least one directory whose name is DATA, MYDATA, MY-DATA or DATALIBRARY. In this case, Vista will process it when it is double-clicked. Otherwise it will be treated as if it is a .VIS file, except that it will run the Lisp Editor when it is double-clicked.

**TEXT FILE (.TXT)**

Finally, you can export your data for use by other programs if you save it as a .TXT file. It can also be re-imported into ViSta via the IMPORT DATA (or OPEN DATA) menu items.

**APPENDIX: VISTA'S MENUS - The File Menu - Save Model As ... (A.14.8)**

SAVE MODEL saves your ViSta model as a file on your computer so that it can be used again at a later time.

If you are using guidemaps, then at this point you should save the model before proceeding. You do that by using the !! side of the SAVE MODEL button.

**APPENDIX: VISTA'S MENUS - The File Menu - New Edit... (A.14.9)**

The NEW EDIT item runs the Lisp Editor with a new, empty file into which you may enter text. This does not have to be a Lisp or ViSta program, although the Lisp Editor is specially connected with the ViSta/XLispStat system so that you can evaluate statements in the editor to check them for errors and see their consequences.

**APPENDIX: VISTA'S MENUS - The File Menu - Open Edit ... (A.14.10)**

The OPEN EDIT item opens a file for editing with the Lisp Editor.

After using OPEN EDIT you will see a dialog box for selecting the file that you wish to edit. Use the dialog in the usual way to find the file.

**APPENDIX: VISTA'S MENUS - The File Menu - Load Edit ... (A.14.11)**

LOAD EDIT loads and evaluates (runs) a ViDAL or Lisp program file.

After using LOAD EDIT you will see a dialog box for selecting the file that contains the program you wish to load and evaluate. Use the dialog in the usual way to find your file.

APPENDIX: VISTA'S MENUS - The File Menu - Print File ... (A.14.12)  
PRINT FILE

The PRINT FILE menu item opens a dialog box for you to select a file, which is then printed.

APPENDIX: VISTA'S MENUS - The File Menu - Print Active Pane... (A.14.13)  
PRINT ACTIVE PANE

The PRINT ACTIVE PANE menu item prints an image of the active window pane (window, if the window has no panes).

A window pane is activated by clicking on it.

APPENDIX: VISTA'S MENUS - The File Menu - Print Entire Window (A.14.14)  
PRINT ENTIRE WINDOW

The PRINT ENTIRE WINDOW menu item prints an image of the active window.

A window is activated by clicking on it.

APPENDIX: VISTA'S MENUS - The File Menu - Exit (A.14.15)  
EXIT

The EXIT menu item terminates your session with ViSta. Please make sure that you have saved your work.

APPENDIX: VISTA'S MENUS - The Edit Menu - The Edit Menu (A.15.1)  
THE EDIT MENU

The edit menu has the usual edit functions. COPY can be used to copy any graphic that appears on the screen. The remaining items only apply to text in the Listener window.

## APPENDIX: VISTA'S MENUS - The Edit Menu - Cut (A.15.2)

## CUT

Cuts selected text from the Listener window's prompt line.

## APPENDIX: VISTA'S MENUS - The Edit Menu - Copy (A.15.3)

## COPY

Copies the active window (or window-pane, if the window has multiple panes) into the clipboard. Does not copy portions of the window (pane) which are not visible on the screen. The contents of the clipboard can be pasted into other graphical programs.

A window (pane) is made active by clicking on it.

## APPENDIX: VISTA'S MENUS - The Edit Menu - Paste (A.15.4)

## PASTE

Pastes text into the Listener window's prompt line.

## APPENDIX: VISTA'S MENUS - The Edit Menu - Clear (A.15.5)

## CLEAR

Clears selected text from the Listener window's prompt line.

## APPENDIX: VISTA'S MENUS - The Edit Menu - Copy-Paste (A.15.6)

## COPY -PASTE

Copies selected Listener text to the Listener's prompt line.

## APPENDIX: VISTA'S MENUS - The Data Menu - The Data Menu (A.16.1)

This topic presents you with the help information for the items of the Data Menu. Choose an item to see its help.

## APPENDIX: VISTA'S MENUS - The Data Menu - About These Data (A.16.2)

The ABOUT THESE DATA and ABOUT THE ANALYSIS menu items present information about data objects and analysis methods.

You can enter this information for your data objects. To do this:

1) Click on the data icon for which you wish to enter information.

2) Type:

(about-these-data "INFORMATION")

in the listener window, replacing INFORMATION with the information about the data that you wish to enter. The information must be inside double quote marks. If the information itself contains double quote marks, they must be preceded by a back-slash.

## APPENDIX: VISTA'S MENUS - The Data Menu - Data Header (A.16.3)

## APPENDIX: VISTA'S MENUS - The Data Menu - Visualize Data ... (A.16.4)

The VISUALIZE DATA and VISUALIZE MODEL items produce a visualization of your data or model.

The data and model visualizations are designed to help you see what your data --- or the model of your data --- seem to say.

Visualizations consist of several plot windows that work together. The group of plot windows is called a SpreadPlot. Help about the spreadplot is available from the help menu's SPREADPLOT HELP item. In addition, help may be obtained for the individual plots by using the help menu's WINDOW HELP item. Finally, many plot windows have a help button which provides help about the individual plot window.

When you are finished with the visualization, use any of the close boxes. All of the windows will close together.

Each model has its own visualization. In addition, there are several different data visualizations, depending on the selection of variable types active at the time the visualization is chosen.

When all variables are numeric, there are four possible visualizations: A multivariate visualization for data with 3 or more numeric variables; a bivariate visualization for data with two numeric variables; a univariate visualization for data with one numeric variable; and a Guided Tour visualization for data which have 6 or more numeric variables.

Finally, there is a classification visualization for data which have a numeric variable and one or more category variables; a frequency visualization for frequency data (data which have numeric variables that specify frequency values); and a category visualization for category data (data which have one or more category variables and no numeric variables).

#### APPENDIX: VISTA'S MENUS - The Data Menu - Summarize Data ... (A.16.5)

SUMMARIZE DATA displays summary statistics about your data.

These summary statistics include, for each numeric variable, the variable's mean, standard deviation, variance, skewness, kurtosis, and the number of observations.

In addition, the five number summary (the minimum, 2nd quartile, median, 4th quartile and maximum) is presented for each ordinal and numeric variable.

The SUMMARIZE DATA menu item lets you choose to also get information about ranges, interquartile ranges, correlations and covariances.

#### APPENDIX: VISTA'S MENUS - The Data Menu - List Data (A.16.6)

LIST DATA

The List Data menu item produces a window containing a listing of your active data.

#### APPENDIX: VISTA'S MENUS - The Data Menu - Create Stats Object (A.16.7)

The CREATE STATS OBJECT menu item creates a new data object which contains the summary statistics for the current data object.

#### APPENDIX: VISTA'S MENUS - The Data Menu - Create Data Object (A.16.8)

CREATE DATA creates new data from the current model. For some models, many different kinds of data can be

created, and dialog box will be presented for you to select the kinds you want. Other models cannot create new data. A message will tell you if this is the case.

#### APPENDIX: VISTA'S MENUS - The Data Menu - Delete Data Object ... (A.16.9)

DELETE DATA deletes the current data object AND all associated objects, if any.

The current data object is the one highlighted on the workmap and checkmarked in the data menu.

Associated objects are:

- 1) All those "below" the current data object. An object is "below" the current data object if a path of connecting lines exists from the lower edge of the current data object to the "below" object. A "below" object may be a data object, model object, or analysis object.
- 2) Those "immediately above" the current data object. An object is "immediately above" the current data object if it is an analysis object which is connected directly to the upper edge of the data object. An "above" object may only be an analysis object.

#### APPENDIX: VISTA'S MENUS - The Data Menu - Impute Missing Data ... (A.16.10)

The IMPUTE MISSING DATA menu item is available to you when there are missing values in your data. Since data with missing values cannot be used in ViSta's analysis methods, some way of pre-processing the data must be provided. The menu item provides three of the most common methods used to deal with missing data:

- 1) Listwise deletion: Any observation with a missing value is deleted from the dataset.
- 2) Pairwise deletion: Correlation/covariance matrices are computed on the basis of cases which do not have pairs of missing values.
- 3) Maximum Likelihood: An iterative process attempts to obtain maximum likelihood estimates of the missing values. These estimates are used to replace the missing values so that no data has to be thrown out.

#### APPENDIX: VISTA'S MENUS - The Data Menu - Edit Data (A.16.11)

##### EDIT DATA

The Edit Data menu item displays a new datasheet icon on your workmap, along with a datasheet of your data. The datasheet is ready for you to edit your data.

The menu item also turns off the menu items and many other functions so that you can safely edit your data without concern about what the rest of the system will do with a partially edited set of data.

If you wish to analyze the data as you have edited them, then you can  
- right-click the datasheet and choose the CREATE DATAOBJECT item  
- click on the workmap and then right-click on the datasheet icon, choosing the CREATE DATAOBJECT menu item.

If you wish to return to the data prior to editing, simply click on the data icon to which it is attached. The data in that icon have not been altered.

The editing you have done is not lost when you close the datasheet.

During a single session with ViSta you can return to your editing of a datasheet at any time. You can always open it again during the same session and continue from where you left off.

Note, however, that if your changes are not saved with the SAVE DATA AS item before you end the session with ViSta, that your changes are lost.

## EDITING YOUR DATA WITH THE DATASHEET EDITOR

Datasheets display your data and let you edit your data. Datasheets can be used to create a brand new data object, to change the data in an already existing data object, or to add new data to an already existing data object.

Datasheets support the standard editing functions. Click on a cell to edit it. Type the desired information. Use the delete key to delete what you have typed. Use the cursor keys or the return, home, end or tab keys to move to another cell.

Datasheets are not spreadsheets: They do not let you enter mathematical formulas in the cells. To do this, use a spreadsheet such as Excel. When you have the desired spreadsheet you can transfer it to ViSta as a datasheet, taking advantage of ViSta's advanced and extensive visualization and analysis features. When you make the transfer from Excel to ViSta, Excel's numerical and textual information is transferred as is, as are the values resulting from the Excel's formulas. The formulas themselves are not transferred.

### CONTENTS OF THE DATASHEET:

The datasheet contains OBSERVATION LABELS, VARIABLE NAMES, VARIABLE TYPES and DATA, as follows:

**OBSERVATION LABELS:** The left column of cells contains observation labels. These may contain any information that you wish. It is recommended that each label be unique.

**VARIABLE NAMES:** The top row of cells contains variable names. These may contain any information that you wish. It is recommended that each variable name be unique.

**VARIABLE TYPES:** The second row of cells contains variable types. ViSta recognizes two variable types: Numeric and Category. Type N for numeric, C for Category. Any other information typed in these cells is ignored.

**DATA:** The remaining cells of the datasheet contain the data. What you can type depends on the variable type specified for the variable (in the second header row).

**CATEGORY** variables form strings from the characters typed in a cell of the datasheet. Most characters are acceptable, although some are ignored and some are interpreted as movement characters.

**NUMERIC** and **ORDINAL** variables assume that their cells contain numbers. For this reason you can only enter the ten digits and the characters + - . and , (the . and , can only be typed at "appropriate" places, where, I regret to say, "appropriate" is according to American rules of numeric notation).

**SCIENTIFIC NOTATION** may be used, with the scientific-notation precision-type character (d, l, f, or s) being typed at the "appropriate" place. While you can type d (double-float) l (long-float) f (single-float) or s (short-float), ViSta only uses double-float and long-float, with the conversion details depending on the machine you are using. You can determine the precision by typing the global variable MACHINE-EPSILON. The value of this variable is the smallest floating point number for which  $(1 + \text{MACHINE-EPSILON})$  is not equal to 1.

**MISSING VALUES.** The datasheet can contain missing values. Missing values are entered as --- (three minus signs typed without spaces between them) and are displayed in the datasheet as --- or as NIL. For numeric variables --- has the numeric value equal to MACHINE-EPSILON. For character variables is is equal to NIL.

### REFORMATING THE DATASHEET:

You expand the datasheet (add rows, columns or matrices) by clicking on special cells in the datasheet. To add a new observation (row) to multivariate data, click on the "New Obs" cell located below the left-hand column. To add a new variable (column) to the data, click on the "New Var" cell located to the right of the top row. You expand matrix data in a similar fashion. You can also use the EXPAND button (or menu item) to add several rows or columns simultaneously. You can control the width of columns and the number of decimal places shown by using the FORMAT button on the button bar, or appropriate items of the DESKTOP menu.

### SAVING THE DATA:

You may use the button bar's SAVE button, or the FILE menu's SAVE DATA menu item to save the data. When

you save the data, the current data object will be updated, unless it is attached to another icon. If so, the old data object will be left unchanged, and a new one will be created from the information in the datasheet.

#### APPENDIX: VISTA'S MENUS - The Data Menu - Edit Source (A.16.12) EDITING THE DATA SOURCE CODE

You can enter your data by using a text editor to write code in the Lisp programming language underlying ViSta. A brief introduction is given here. You can also get a good idea of the format by looking at any file in one of the data folders. See the User's guide for more information.

Briefly, the minimum format for multivariate data is:

```
(data "name"
:variables ("Variable A" "Variable B" "Variable C")
:data '(
1 2 3
4 5 6
7 8 9
10 11 12
))
```

Additional features are described in the User's Guide.

The DATA function creates a new data object or reports the names of all data objects or the object identification of a specific object. It can be used in three different ways:

1) To see a list of all data objects, type:

```
(DATA)
```

2) To see the object identification of data object NAME, type:

```
(DATA NAME)
```

3) To create a new data object from information contained within the DATA statement, the minimum syntax requires you to type:

```
(DATA 'NAME :VARIABLES <VARLIST> :DATA <DATALIST> )
```

#### GENERAL ARGUMENTS:

&OPTIONAL NAME &KEY DATA VARIABLES TYPES LABELS FREQ ABOUT

Defines ViSta data object NAME. NAME, DATA and VARIABLES are required arguments.

NAME must be a string or a symbol. If a symbol it must be preceeded by a single quote.

DATA must be followed by a list of numbers, strings or symbols (or mix of such). Symbols are converted to uppercase strings. The number of data elements must conform to the information in other arguments.

VARIABLES must be followed by a list of strings or a single quoted list of symbols defining variable names (and, indirectly, the number of variables).

TYPES, optional, must be a list of the strings \"numeric\", \"ordinal\" or \"category\" (case ignored), or a list of the corresponding symbols. These strings or symbols specify whether the variables are numeric, ordinal or categorical (all numeric by default). Note that the ordinal datatype is seldomly used.

LABELS, optional, must be a list of strings or symbols specifying observation names (\"Obs1\", \"Obs2\", etc., by default). Symbols are converted to uppercase strings.

FREQ specifies that the values of the numeric variables are frequencies.

ABOUT is an optional string of information about the data.

Given these arguments above you can specify the following types of data:

- 1) MULTIVARIATE data are data which are not one of the other data types given below. These data include univariate (one numeric or ordinal variable) and bivariate (two numeric or ordinal variables) data.
- 2) CATEGORY data have one or more CATEGORY variables and no NUMERIC or ORDINAL variables. The N category variables define an n-way classification.
- 3) CLASSIFICATION data have one NUMERIC variable and one or more CATEGORY variables. The N category variables define an n-way classification. The numeric variable specifies an observation for a given classification.
- 4) FREQUENCY CLASSIFICATION data are classification data whose numeric variable specifies frequencies as indicated by using FREQ. The N category variables define an n-way classification, with the numeric variable specifying the co-occurrence frequency of a specific combination of categories.

#### ARGUMENTS FOR FREQUENCY TABLE DATA:

ARGUMENTS: &KEY ROW-LABEL, COLUMN-LABEL, and FREQ

For FREQUENCY TABLE data, the ROW-LABEL and COLUMN-LABEL arguments must be used: These data have NUMERIC variables whose values specify frequencies as indicated by using FREQ. The data are a two-way cross tabulation of the co-occurrence frequency of the row and column entities. The ROW-LABEL and COLUMN-LABEL identify the data as two-way array (table) data, with the numeric variables in the data represent columns of a two-way table rather than variables of a multivariate dataset. ROW-LABEL and COLUMN-LABEL each have a string value which labels the rows or columns (ways) of the array. Observation labels are used to label the row-levels and variable names the column-levels.

#### ARGUMENTS FOR MATRIX DATA

ARGUMENTS: &KEY MATRICES SHAPES

These arguments are used to identify matrix data. MATRICES, required for matrix data only, must be a list of strings specifying matrix names (and, indirectly, the number of matrices). SHAPES, optional for matrix data only, must be a list of strings \"symmetric\" or \"asymmetric\" (case ignored), specifying the shape of each matrix (all are symmetric by default). Matrix arguments cannot be used with array data.

#### EFFICIENCY ARGUMENTS: &KEY MISSING-VALUES, STRINGS

- 1) If you know the data do or do not contain missing values, specifying MISSING-VALUES as T or NIL eliminates the time required to check for missing values.
- 2) Specify STRINGS T if you know that all category values are represented by strings (values inside double-quote marks) then the need to check for non-string category values is eliminated, greatly increasing efficiency, especially for large data.

The preceding arguments are all of those concerning the definition of data within ViSta. There are additional arguments which concern the creation of data by programming. These are described below.

#### ARGUMENTS FOR PROGRAMMING

ARGUMENTS: &KEY PROGRAM, USE

If you wish to write a data program, use these arguments:

- 1) PROGRAM (required) specifies that a program follows which computes N new variables. The program must return a list of N lists corresponding to the N variables in the :VARIABLES keyword. Causes the new variables to be bound.
- 2) USE (optional) specifies the data object whose variables are input to the program. This must be a data object with bound variables. If not, specify :USE (BIND-VARIABLES DOB)

For example, assume there is a dataobject named pcaexmpl containing variables X and Y. Then you can type: .

```
(data "PCAExmpl2"
 :use pcaexmpl
 :variables '("X" "Y" "A" "B" "C")
 :program
 (let* ((A (+ X Y))
 (B (+ (* 5 X) Y))
 (C (+ (* -2 X) Y)))
 (list x y a b c)))
```

Which creates a new data object containing variables a b and c as well as the original x and y.

Consider this example (which assumes the variables hmw1 through hmw4 already exist in the points dataobject):

```
(data "WeightedPoints"
  :use points
  :program
  (let* ((h1 (* hmwk1 5/10))
        (h2 (+ (* hmwk2a 6/10) (* hmwk2b 4/10)))
        (h3 (+ (* hmwk3abd 7/10) (* hmwk3c 3/10)))
        (h4 (* hmwk4 15/10))
        (hmwksum (+ h1 h2 h3 h4))
        (total (+ hmwksum midterm1 attend vis)))
    (list attend vis h1 h2 h3 h4 hmwksum total))
  :variables '("attend" "visual" "h1" "h2" "h3" "h4" "htot" "total"))
```

This program takes the variables hmw1 through hmw4 in the points dataobject and combines them together for a total homework score, plus adding other variables in SCORE to get a point total for a class.

Finally, there are arguments that effect the operation of the system. These should only be used by knowledgeable system developers. These arguments are:

&KEY CREATED CREATOR-OBJECT SUBORDINATE ICONIFY DATASHEET-ARGUMENTS NEW-DATA ALL-TYPES-IN-DATA-ARRAY

#### APPENDIX: VISTA'S MENUS - The Data Menu - Edit Variables (A.16.13) EDIT VARIABLES:

The Edit Variables menu item displays a window in which you can enter statements using ViSta Interactive Variable Application, an algebra-like language which calculates values for new variables and makes the variables available for further manipulation.

ViVa consists of algebra-like expressions. If the expression involves assignment to a variable, the variable is assigned the value of the expression, and the variable is saved for the duration of the session with ViSta.

ViVa can be used in 3 ways. These ways are outlined briefly here. Examples of ViVa expressions are given below.

1) At the Listener, type a ViVa expression enclosed in brackets. For example, if x is a variable known to Lisp from, say, typing

```
> (var x '(1 2 3))
(1 2 3)
```

Then you can type the ViVa expression, in brackets, at the Lisp prompt:

```
> [y = 2 + 3*x]
```

```
(5 8 11)
```

The expression on the right side of the = sign is evaluated, the resulting value is assigned to y, and the value is returned. Thus:

```
> [x=4.5]
4.5
> [y = 2 + 3*x]
15.5
```

2) At the Listener, type a left bracket and a return. Lisp's > prompt is replaced by ViVa's ? prompt. Then you can type a series of ViVa expressions, each followed by a return. Each expression is evaluated when you type a return. Finally, type a right bracket to return to regular Lisp. The value of the last expression is returned. For example:

```
> [
? s = list(2,4,6)
(2 4 6)
? u = 3
3
? v = 2
2
? r = u + v*s
(7 11 15)
? ]
(7 11 15)
>
```

3) Enter a bracket enclosed ViVa statement in the middle of lisp. It is evaluated and returns its value to Lisp.

Thus:

```
> (list 1 2 [sqrt(9)] 4)
(1 2 3.0 4)
```

#### LIMITATIONS:

- 1) ViVa can only be entered from the Listener, not from files. Thus, ViVa scripts or applets are not possible.
- 2) When using ViVa, variable names cannot contain embedded characters interpreted as mathematical operators (i.e., - + = / \*, etc). These signs are always interpreted as being a mathematical operator. In particular, the Lisp convention of variable or function names containing dashes, such as principal-components or normal-rand, is not allowed in ViVa. Underscores may be substituted for dashes (see next point).
- 3) Underscores may only be used as a substitute for dashes (see previous point).

#### UNDERSCORES:

Since dashes are very commonly used in the names Lisp functions, you may substitute underscores for dashes. Thus, you can enter principal\_components or normal\_rand. ViVa will translate appropriately.

#### VERBOSE:

Type (viva-verbose t) or (viva-verbose nil) at the Lisp > prompt to control the amount of output generated by ViVa.

#### VIVA EXPRESSIONS:

ViVa expressions conform to a subset of the syntax for the C programming language. Specifically, ViVa supports all syntax defined in section 18.1 of the original Kernighan and Ritchie book, plus all C numerical syntax including floats and radix syntax (i.e. 0xnnn, 0bnnn, and 0onnn). ViVa supports multiple array subscripts.

#### ABOUT VIVA AND PARCIL

ViVa uses PARCIL, copyright (c) 1992 by Erann Gat. Parcil is used under the terms of the GNU General Public License as published by the Free Software Foundation. PARCIL parses expressions in the C programming language into Lisp. ViVa then takes the parsed statements and creates an interactive environment in which they are evaluated. Thanks to Sandy Weisberg for providing Parcil.

#### ViVa EXAMPLES

Examples of each way of using ViVa are given below.

1) At the Listener, type a ViVa expression enclosed in brackets. It is evaluated and the value returned. If the expression involves assignment to a variable, the variable is assigned the value, and the variable is saved for the duration of the session. The variable can be entered into a data object, using the DATASET function, and then saved permanently.

Here is an example:

```
> [A = 2 + (3*4)]
14
```

Here, the user has typed the ViVa expression  $A=2+(3*4)$  enclosed in brackets. ViSta responds with 14, the appropriate value, using the standard rules of operator precedence. A new Lisp variable A has been created:

```
> a
14
>
```

You can retrieve the list of all saved variables created by ViVa by typing

```
> $viva-vars
```

These variables, and those created by the VAR function, are free variables (not in a data object). You can get a list of all free variables by typing:

```
> $free-vars
```

You can also retrieve the list of all variables in all data objects by typing:

```
> $data-vars
```

Note that any Lisp function may be used in ViVa expressions, though in standard algebraic syntax. For example:

```
> [a=3*iseq(4)]
(0 3 6 9)
```

where  $iseq(4)$  is the ViVa equivalent of Lisp's  $(iseq 4)$ , which generates the sequence of numbers (0 1 2 3).

These functions can use any \$data-vars as arguments:

```
> [b=a^2]
> (0 9 36 81)
```

A Lisp expression involving a function with multiple arguments is written as a ViVa function involving arguments that are comma-delimited. Thus

```
> [L=list(1,2,3)]
```

corresponds to the Lisp expression  $(setf L (list 1 2 3))$ .

Note that a ViVa expression may be a compound expression:

```
> { ( x=1,y=2,print(x+y),sin(pi/2) ) }
3
1.0
> x
1
> y
2
>
```

2) At the Listener, type a left bracket and a return. Lisp's > prompt is replaced by ViVa's ? prompt. Then you can type a series of ViVa expressions, each being evaluated when you type a return. Finally, type a right bracket to return to regular Lisp. Notice that functions with multiple arguments have their arguments separated by commas. Also, notice that vector and matrix algebra are supported. Here is an example interaction:

```
> [
? 2+3
5
? a
(0 3 6 9) ;this value for A is left from above
? a=2+3
5
? a ;a new value for A has been defined
5
? b=sqrt(a)
2.23606797749979
? a=iseq(4) ;yet another value for A
(0 1 2 3)
```

```
? b=sqrt(a) ;vector arithmetic is supported
(0.0 1.0 1.4142135623730951 1.7320508075688772)
? ] ;the value of the last expression is returned
(0.0 1.0 1.4142135623730951 1.7320508075688772)
>
```

If you do not wish to see so much output, type:

```
[viva_verbose=not(t)]
```

For example:

```
> [viva_verbose=not(t)
? a=normal_rand(10)
? b=a^2
? c=iseq(10)
? d=c*b
? ]
(0.0 0.12086435903614712 0.12465600120144842 0.05113395472276512 0.8572483580685174
2.8192853102290143 8.009152301820079 33.31659449692597 0.9436767853187026 1.1345846087676839)
> d
(0.0 0.12086435903614712 0.12465600120144842 0.05113395472276512 0.8572483580685174
2.8192853102290143 8.009152301820079 33.31659449692597 0.9436767853187026 1.1345846087676839)
>
```

Lisp's matrix language may be used with ViVa. Here is an example:

```
?a=matrix ( list(2,2), list(1, 2 ,3 ,4))
#2A((1 2) (3 4))
?b=matrix ( list(2,3), list(1,2,3,4,5,6))
#2A((1 2 3) (4 5 6))
?c=matmult(a,b)
#2A((9.0 12.0 15.0) (19.0 26.0 33.0))
```

3) Enter a bracket enclosed ViVa statement in the middle of lisp. It is evaluated and returns its value to Lisp:

```
>(list 1 3 [sqrt(25)] 7)
(1 3 5.0 7)
```

## TWO EXAMPLES

The first example involves calculating points in an Introductory Statistics course. The data are in P3099pts.lsp datafile. You can open these data with the OPEN DATA menu item in the FILE menu. In this example, ViVa is used to create variables, then the DATASET function is used to create a data object from them.

```
[
? homework_sum=homework1+homework2=homework3
(12.0 14.24 13.399999999999999 13.11 15.36 12.96 12.22 13.82 14.31 13.09 13.98 13.67 7.45 15.33 12.1 12.92
5.57)
? total=homework_sum+midterm1
(170.0 173.24 168.4 189.11 197.36 119.96 159.22 135.82 194.31 189.09 197.98 189.67 159.45 194.33 153.1
176.92 173.57)
? ]
(170.0 173.24 168.4 189.11 197.36 119.96 159.22 135.82 194.31 189.09 197.98 189.67 159.45 194.33 153.1
176.92 173.57)
> (dataset summary homework_sum total)
#<Object: a9e82c, prototype = MV-DATA-OBJECT-PROTO>
>
```

The second example involves calculating grades from points earned during the course.

```
(load (strcat *data-dir-name* "p3099pts.lsp"))
```

```
[homeworksum = homework1 + homework2 + homework3]
[midpts = homeworksum + midterm1]
```

```
(let ((grades (mapcar #'(lambda (pts)
```

```
(cond ((> pts 190) "A")
      ((< 170 pts 190) "B")
      ((< 150 pts 170) "C")
      ((< 140 pts 150) "C-")
      ((< 0 pts 150) "D")))
      midpts))))
```

```
(dataset homework1 homework2 homework3 homeworksum
      midterm1 midpts grades)
```

#### APPENDIX: VISTA'S MENUS - The Data Menu - Print Data (A.16.14) PRINT DATA

This item prints a listing of your data on your printer.

#### APPENDIX: VISTA'S MENUS - The Data Menu - Print Source (A.16.15) PRINT DATA

This item prints the source code underlying your dataobject on your printer.

#### APPENDIX: VISTA'S MENUS - The Data Menu - Merge Variables (A.16.16)

MERGE VARIABLES enables you to merge the variables of one dataset with the variables of another dataset to create a third dataset that has both sets of variables in it.

The variables in the current dataset (the highlighted one) are merged with those in the previously current dataset (the one current before the current one). In the new dataset, the variables in the previously current dataset will appear before those in the current dataset.

The two datasets that are merged must have the same number of observations (rows). Usually, it does not make sense to merge variables unless the observations in the two datasets are the same, and in the same order, although ViSta does not check on this.

#### APPENDIX: VISTA'S MENUS - The Data Menu - Merge Observations (A.16.17)

MERGE OBSERVATIONS enables you to merge the observations of one dataset with the observations of another dataset to create a third dataset that has both sets of observations in it.

The observations in the current dataset (the highlighted one) are merged with those in the previously current dataset (the one current before the current one). In the new dataset, the observations in the previously current dataset will appear before those in the current dataset.

The two datasets that are merged must have the same number of variables (columns). Usually, it does not make sense to merge observations unless the variables in the two datasets are the same, and in the same order, although ViSta does not check on this.

#### APPENDIX: VISTA'S MENUS - The Data Menu - Merge Matrices (A.16.18)

MERGE MATRICES enables you to merge the matrices of one dataset with the matrices of another dataset to create a third dataset that has both sets of matrices in it.

The matrices in the current dataset (the highlighted one) are merged with those in the matrices current dataset (the one current before the current one). In the new dataset, the matrices in the previously current dataset will appear before those in the current dataset.

The two datasets that are merged must have conformable matrices (the matrices must have the same number of rows and columns). Usually, it does not make sense to merge matrices unless the rows and columns in the two datasets are the same, and in the same order, although ViSta does not check on this.

#### APPENDIX: VISTA'S MENUS - The Transform Menu - The Transform Menu (A.17.1)

This topic presents you with the help information for the items of the Transform Menu. Choose an item to see its help.

#### APPENDIX: VISTA'S MENUS - The Transform Menu - Sort-Permute ... (A.17.2)

The Sort-Permute transformation creates a new dataset in which the observations of a selected variable in the current dataset (or of the current dataset's labels, if desired) are sorted into order, and the remaining variables are permuted accordingly.

Sorting is done in ascending order (from small to large, or in the case of category variables, alphabetically) by default, or in descending order if desired.

#### APPENDIX: VISTA'S MENUS - The Transform Menu - Ranks (A.17.3)

The Ranks transformation creates a new dataset in which the values of each numeric or ordinal variable in the current dataset are converted into ranking numbers. Category variables are not included in the new dataset.

Ranks are values between 1 and N, the number of observations. Ranks start at 1, for the smallest value of a variable, and proceed to N for the largest. Ties in the values of a variable are converted into means of appropriate ranks.

#### APPENDIX: VISTA'S MENUS - The Transform Menu - Normal Scores (A.17.4)

The Normal Scores transformation creates a new dataset in which the values of each numeric variable in the current dataset are converted into normal scores. Ordinal and category variables are not included in the new dataset.

Normal scores are the normalized z-scores that would be obtained under the assumption that the variable is normally distributed, given the variable's mean and standard deviation.

#### APPENDIX: VISTA'S MENUS - The Transform Menu - Absolute Value (A.17.5)

The Absolute Value transformation creates a new dataset from the current dataset in which the values of the numeric variables in the new dataset are the absolute values of those in the current dataset.

Ordinal and category variables are not placed into the new dataset.

#### APPENDIX: VISTA'S MENUS - The Transform Menu - Rounding ... (A.17.6)

The Rounding transformation creates a new dataset from the current dataset in which the values of the numeric variables in the current dataset are rounded.

Rounding is converting decimal numbers to the nearest integer.

Ordinal and category variables are not placed into the new dataset.

#### APPENDIX: VISTA'S MENUS - The Transform Menu - Reciprocal (A.17.7)

The Reciprocal transformation creates a new dataset from the current dataset in which the values of the numeric variables in the new dataset are the reciprocals of those in the current dataset.

The reciprocal of the number A is  $1/A$ .

Ordinal and category variables are not placed into the new dataset.

**APPENDIX: VISTA'S MENUS - The Transform Menu - Exponential ... (A.17.8)**

The Exponential transformation creates a new dataset from the current dataset in which the values of the numeric variables in the new dataset are the exponentials of those in the current dataset. A choice of several specific exponential transformations is provided.

Ordinal and category variables are not placed into the new dataset.

**APPENDIX: VISTA'S MENUS - The Transform Menu - Logarithm ... (A.17.9)**

The Logarithm transformation creates a new dataset from the current dataset in which the values of the numeric variables in the new dataset are the logarithms of those in the current dataset. A choice of natural (base e) or base X logarithms is provided, where the value of X is specified in an additional dialog box.

Ordinal and category variables are not placed into the new dataset.

**APPENDIX: VISTA'S MENUS - The Transform Menu - Trigonometric ... (A.17.10)**

The Trigonometric transformation creates a new dataset from the current dataset in which the values of the numeric variables in the new dataset are trigonometric transformations of those in the current dataset. A choice of the familiar trigonometric functions is provided in an additional dialog box.

Ordinal and category variables are not placed into the new dataset.

**APPENDIX: VISTA'S MENUS - The Transform Menu - Correlations (A.17.11)**

The Correlations transformation creates a new dataset from the numeric variables in the current dataset. The values in the new dataset are the correlations between the numeric variables in the current dataset. The diagonal contains ones. The new dataset has the same number of observations as it has variables. The observations labels are constructed from the variable names.

Ordinal and category variables are not used in the calculations.

**APPENDIX: VISTA'S MENUS - The Transform Menu - Covariances (A.17.12)**

The Covariances transformation creates a new dataset from the numeric variables in the current dataset. The values in the new dataset are the covariances of the numeric variables in the current dataset. The variable variances are on the diagonal. The new dataset has the same number of observations as it has variables. The observations labels correspond to the variable names.

Ordinal and category variables are not used in the calculations.

APPENDIX: VISTA'S MENUS - The Transform Menu - Distances (A.17.13)

The Distances transformation creates a new dataset from the numeric variables in the current dataset. The values in the new dataset are the distances between the numeric variables in the current dataset. The diagonal contains zeros. The new dataset has the same number of observations as it has variables. The observations labels are constructed from the variable names.

Ordinal and category variables are not used in the calculations.

APPENDIX: VISTA'S MENUS - The Transform Menu - Transpose Data (A.17.14)

The Transpose Data transformation creates a new dataset from the current dataset in which the observations (rows) of the current dataset become the variables (columns) of the new one, and the variables (columns) of the current dataset become the observations (rows) of the new one.

APPENDIX: VISTA'S MENUS - The Transform Menu - Standardize Data ... (A.17.15)

The Standardize Data transformation creates a new dataset from the current dataset in which the values of the numeric variables in the current dataset are standardized to have a specified mean and standard deviation (0 and 1 by default).

Ordinal and category variables are not placed into the new dataset.

APPENDIX: VISTA'S MENUS - The Transform Menu - Orthogonalize Data (A.17.16)

The Orthogonalize Data transformation creates a new dataset from the current dataset in which the values of the numeric variables in the current dataset are orthogonalized. Orthogonalized variables have zero correlation. The Graham-Schmidt orthogonalization process is used.

Ordinal and category variables are not placed into the new dataset.

APPENDIX: VISTA'S MENUS - The Transform Menu - Folded Power (A.17.17)

The Folded Power transformation creates a new dataset where the variables have been transformed according to the following transformation:

$$\frac{(2Y)^p - (2-2Y)^p}{2p} \quad \text{for } p \neq 0$$

$$\frac{\log(Y) - \log(1-y)}{2} \quad \text{for } p = 0$$

Where  $p$  is controlled by a slider in a visualization specially designed to evaluate the asymmetry and linearity of the variables in the dataset after the transformation. This transformation is appropriate for proportions or percentages or for variables that need transformations that work on both sides of its distribution.

#### APPENDIX: VISTA'S MENUS - The Transform Menu - Box-Cox Power (A.17.18)

The Box Cox Power transformation creates a new dataset where the variables have been transformed according to the following transformation:

$$(Y^p - 1)/p \quad \text{for } p \text{ not equal to } 0$$

$$\log(Y) \quad \text{for } p \text{ equal } 0$$

Where  $p$  is controlled by a slider in a visualization specially designed to evaluate the asymmetry and linearity of the variables in the dataset after the transformation. When  $Y$  has values equal or below 0, a constant is added to the variable to make all the values in  $Y$  positive. This transformation is appropriate for continuous numeric asymmetric variables. Some interesting values of  $p$  are:

- 2.0 - square - power of two
- 1.0 - no transformation?
- 0.5 - square root
- 0.0 - logarithm
- 1.0 - inverse

#### APPENDIX: VISTA'S MENUS - The Transform Menu - Dummy coding (A.17.19)

The dummy code transformation creates a new dataset where the categorical variables with  $k$  categories are transformed into  $k$  binary variables where 1 indicates presence of a category and 0 absence. This transformation is appropriate for including categorical variables in certain statistical analysis that expect only numeric variables. For example,  $k-1$  dummy variables can be included as independent variables in a regression analysis.

**APPENDIX: VISTA'S MENUS - The Transform Menu - Split By Categories (A.17.20)**

The split by categories transformation creates a new dataset for each category of the selected category variables. This dataset contains the values of the cases of the numeric variables in the original dataset falling in the categories of the categorical variables. This transformation can be useful for carrying out separate analysis for cases in different categories, such as male and female.

**APPENDIX: VISTA'S MENUS - The Transform Menu - Join Category Variables (A.17.21)**

The join category variables transformation creates a new dataset with the values of the category variables selected in the original dataset concatenated. This transformation combines the categorical variables and creates new variables that can be more useful for the purpose of analysis.

**APPENDIX: VISTA'S MENUS - The Transform Menu - Program Editor (A.17.22)**

The Transform menu's Program Editor menu item lets you define your own transformation program. It can use variables that are in one or more existing data objects, use variables that are not in any data object, compute new variables from old ones, and place the new ones in a new data object. You can also create a new data object that does not involve variables in an old data object.

The menu item gives you the choice of using the Listener window to enter the calculations interactively, or of using the LispEdit program editor to create a script which can be run with the editor and/or saved for later use. The script can be run by using LispEdit's Eval menu item, or if it is placed in a file, it can be loaded and run using ViSta's Load Edit menu item. You can choose to use both approaches simultaneously, since it can be useful to use the listener to test statements in the program and the editor to construct and save the program.

It is important to note that the data objects which have been defined during the analysis session, and the variables in these data objects, are available to you (i.e., are bound in the local environment). Consequently, the menu item informs you of the data objects and variables currently available. These objects and variables are represented by the following symbols:

\$ the current data object (the one whose icon is high-lighted).  
\$data the names for all the data objects that have been defined.  
\$vars the names of all variables in \$.  
\$data-vars the names of all variables in all data objects.  
\$desk-vars the names of all variables that are not in data objects.

Note that the \$data-vars names are two-level names consisting of the name of the data object and the name of the variable, separated by a dash.

All \$vars and \$data\$vars are available for use in your program.

**APPENDIX: VISTA'S MENUS - The Analyze Menu - The Analyze Menu (A.18.1)**

This topic presents you with the help information for the items of the Analyze Menu. Choose an item to see its help.

#### APPENDIX: VISTA'S MENUS - The Analyze Menu - Analysis of Variance ... (A.18.2)

Analysis of Variance (ViSta-ANOVA) is a technique for comparing means of several populations. ViSta can perform one-way, two-way, or n-way analysis of variance with optional two-way interactions. The samples from the populations may be all the same size (balanced) or may be different sizes (unbalanced). There must be at least one observation in each cell (the data must be complete).

In one-way ANOVA samples are drawn from each population and the data are used to test the null hypothesis that the population means are equal.

In two-way ANOVA the populations are classified in two ways. In n-way ANOVA the populations are classified in three or more ways. For these types of ANOVA, the analysis still compares the means of populations. However, in two-way and n-way ANOVA several comparisons of the sample means are possible since there are several classifications. In particular, comparisons are made for the classifications themselves (called "main effects") and for the interactions of each pair of classifications (called "two-way interaction effects"). ViSta-ANOVA does not compare interaction higher than two-way.

ViSta-ANOVA provides significance tests to test the null hypothesis that the group means, as classified, are all equal. For the significance tests to be accurate we must assume that the samples are drawn independently from normally distributed populations which have the same standard deviations.

The ViSta-ANOVA visualization presents plots to help you assess the assumptions and to help you understand the results of the significance tests.

#### APPENDIX: VISTA'S MENUS - The Analyze Menu - Regression Analysis ... (A.18.3)

Regression Analysis (ViSta-Regres) is a technique for predicting one response variable from one or more predictor variables. ViSta-Regres does simple and multiple regression. These are called univariate regression because only one response is being predicted. If you wish to predict more than one response, use ViSta-MulReg, which does multivariate multiple regression.

OLS (ordinary least squares) regression is the most common type of regression. It finds the strongest linear relationship between a linear combination of the predictors and the response. OLS regression assumes that errors are normally distributed and independent, and that predictors are measured without error.

OLS regression is the statistically best method when the assumptions are satisfied. However, when the data fail to meet the assumptions or when there are problems such as outliers or colinearity in the data, the OLS coefficients may be inaccurate estimates of the population parameters.

The ViSta-Regres visualization helps you check on the validity of the OLS assumptions. It also help show you outliers. If the data fail to meet the assumptions of OLS regression, you may use the robust or monotonic regression methods to analyze the data. Robust regression produces regression coefficients which are not influenced by outliers. Monotonic regression is useful when the relationship between the response variable and the predictor variables is nonlinear. You can compare the results provided by OLS, robust, and monotonic regression to determine which is most appropriate for your data.

**APPENDIX: VISTA'S MENUS - The Analyze Menu - Univariate Analysis ... (A.18.4)**

Univariate Analysis (ViSta-UniVar) provides techniques for comparing means of two populations. ViSta-UniVar can compare two sets of data whether they are independent or paired (dependent). It tests whether the means of the two groups are significantly different, and reports the confidence interval for the difference in means.

For samples from independent populations ViSta-UniVar computes Student's T-test and the Mann-Whitney test. For paired (dependent) samples the paired-samples T-Test and the Wilcoxon Signed Rank Test are computed. Student's T-test is used when there is a single sample. ViSta-UniVar can also use the T-test to compare the mean of one population to a pre-specified hypothetical mean. If the population variance is known, then the Z-test is substituted for the T-test.

The T-test (and Z-test) test the null hypothesis that the means of the populations from which the data are sampled are equal. The Mann-Whitney U-test and the Wilcoxon Signed Rank Test use the null hypothesis that both populations are identically distributed.

The ViSta-UniVar visualization presents plots to help you assess the normality assumption.

**APPENDIX: VISTA'S MENUS - The Analyze Menu - Frequency Analysis ... (A.18.5)**

The Frequency Distribution Plots menu item gives you access to three plots designed to show you the shape of a variable's distribution. These plots are:

THE POLYGON PLOT  
THE HISTOGRAM PLOT  
THE HOLLOWGRAM PLOT

You can use the PLOTS button at the bottom of the graph to cycle through three ways of plotting the frequency information: Histogram, Hollow Histogram and Frequency Polygon.

All three of these plots are based on the same approach for determining the shape of a variable's distribution. The three plots differ in how that shape is drawn and shown to you.

These three plots determine the shape of a variable's distribution by breaking the range of the variable's values into equal-sized intervals called BINS. They then determine the number of observations that fall into the interval (are in the BIN) and display the shape of the distribution as follows:

The HISTOGRAM draws a BAR for each BIN  
The HOLLOWGRAM draws a HORIZONTAL LINE for each BIN  
The POLYGON draws a JAGGED LINE for each BIN

For each plot there is a set of red dots. These red dots are located so that their height is proportional to the frequency in the interval. The higher the red dot, the greater the frequency in the bin. The red dots are located above the midpoint of each bin, and to the right of the frequency of the bin. You can brush your cursor over the red dots to see the frequency and midpoint of the bin.

**BINWIDTH**

Unfortunately, all three plots are notorious for conveying an impression of the shape of the variable's distribution that is strongly dependent on the number of bins chosen. Changing the number of bins may radically change the apparent shape of the distribution. Even more unfortunately, there is no entirely satisfactory way to solve this problem.

For this reason there are two buttons on the graph that help you control the number of bins. These are the two

BINWIDTH buttons at the bottom and the NEWBINS button at the top.

The two BINWIDTH buttons can be used to dynamically change the bin widths, and, consequently, the number of bins. By putting your cursor on the button and holding your mouse button down, these buttons allow you to watch the graph change in an animated way. Clicking on the NEWBINS button gives you a dialog box that lets you customize the bin widths and midpoints (as well as the x-axis) to get a better distribution.

We recommend that you first use the BINWIDTH buttons to get a better impression of the distribution's shape, and then the NEWBINS button to choose a "nice" bin width and midpoint. "Nice" means that the distribution adequately portrays the shape of the distribution, and the bin widths, midpoints and axis details use sensible numbers.

The CURVES button can be used to add or remove several different distribution curves, including the normal distribution and several curves called "kernel density distribution curves". The kernel density distribution curves provide several alternate ways of approximating the shape of the population distribution. If the kernel density curves roughly approximate the normal distribution curve, then the variable's distribution approximates normality.

When you click on the X button at the top of the graph you will be presented with a list of variables to display (if there are only two variables, it will switch to the other variable). Clicking on a variable will change the plot to display that variable's Frequency Polygon. When you click on the Y button at the top of the graph the y axis will switch between frequency and probability.

Finally, when you click on the DATA button at the bottom of the graph, you will create a cumulative frequency table dataobject. It contains several variables specifying frequencies and cumulative frequencies, percentages and cumulative percentages, and limits and midpoints.

#### APPENDIX: VISTA'S MENUS - The Analyze Menu - Log linear analysis ... (A.18.6) Help for loglinear analysis

Log-linear models provide a method for analyzing associations between two or more categorical variables. The method has become widely accepted as a tool for researchers during the last two decades.

The visualization for log-linear models help to specify the possible models for a group of variables and assess the fit of these models. The list of terms (the variables and all the interactions between the variables) in the visualization gives a simple and intuitive way of indicating the terms to enter in a model.

If the list of terms is in hierarchical model, all the terms below a selected one are automatically introduced in the model. If the list of terms is not in hierarchical model, terms can be selected individually. The rest of plots help to assess the fit of the model like mosaic plots for observed and predicted values or distance of Cook v. leverages scatterplots. The parameter plot sorts out the parameters of the model and provides a interpretation in terms of frequencies of the original data of each parameter. Finally, the history plot keeps information about the chi square statistic that can be used for returning to previous successful models

#### APPENDIX: VISTA'S MENUS - The Analyze Menu - Multivariate Regression ... (A.18.7)

Multivariate Multiple Regression (ViSta-MulReg) is a data analysis technique for predicting the values of a many response variables simultaneously from the values of two or more predictor variables. If you have just one response variable, you should use ViSta-Regres, not ViSta-MulReg.

ViSta-MulReg uses OLS (ordinary least squares) regression techniques to find the strongest possible linear

relationship between a linear combination of the predictors and one of the response variables. A different linear combination of the predictors is obtained for each response variable. This is the same as repeatedly doing many (univariate) multiple regressions. Multivariate Regression then combines all of the separate (univariate) multiple regressions together to obtain an overall multivariate test of the significance of simultaneously predicting all of the response variables.

If you choose the Redundancy option you fit a Redundancy Analysis model to your data. This model obtains the single linear combination of the predictor variables which simultaneously predicts all of the response variables optimally, in the sense of maximizing the mean squared correlation between the linear combination and each response. This mean squared correlation is higher than that obtained by Multivariate Regression.

The ViSta-MulReg visualization shows the relationship between responses, the linear combinations of predictors, and the redundancy variables. No regression diagnostics are shown (use ViSta-Regres for diagnostics).

#### APPENDIX: VISTA'S MENUS - The Analyze Menu - Principal Components ... (A.18.8)

Principal Component Analysis (ViSta-PrnCmp) is a statistical analysis and visualization method for picturing the variance in a set of continuous multivariate data. Only the numeric variables in the data are used.

The ViSta-PrnCmp visualization shows the maximum variance picture of the data: The first principal component is the linear combination of the variables that has the maximum variation. The second principal component is the linear combination of the variables that is orthogonal (at right angles) to the first one, and has the maximum remaining variation. The third is orthogonal to the first two, and has maximum variance. The scree plot is also shown.

There are two major decisions to be made:

Before the analysis, you must decide whether to obtain the principal components from the correlations or covariances of the variables. The choice of covariances can only make sense if your variables are all on the same scales. If they are not, you must choose correlations. If they are all on the same scales, you may choose either, but correlations is usually most reasonable. Choosing covariances means that the original differences in variance between variables will effect the results. Choosing correlations means this difference in variance will not effect the results.

After the analysis, you must to decide how many principal components are needed to adequately summarize the data. The interpretation for the model will help you make this decision.

#### APPENDIX: VISTA'S MENUS - The Analyze Menu - Item Analysis ... (A.18.9)

##### ITEM ANALYSIS

Item Analysis provides psychometric methods for analyzing items of a test where the methods are based on Classical Test Theory. These methods include split-half reliability, Cronbach's alpha, standard error of measurement, confidence intervals for observed and estimated scores, as well as psychometric indexes for the items.

Item Analysis assumes that there is one numerical variable for each item of a test. You may analyze these data or you may create different item scores from them. The item scores include summated scores, mean scores, transformed scores, estimated scores with their confidence intervals, etc.

**APPENDIX: VISTA'S MENUS - The Analyze Menu - Correspondence Analysis ... (A.18.10)**

Correspondence Analysis (ViSta-Coresp) is a statistical analysis and visualization method for picturing the associations between the levels of a two-way contingency table. The name is a translation of the French "Analyses des Correspondances", where the term correspondance denotes a "system of associations" between the elements of two sets.

In a two-way contingency table, the observed association of two traits is summarized by the cell frequencies. A typical inferential aspect is the study of whether certain levels of one characteristic are associated with certain levels of another.

Correspondence analysis is a geometric technique for displaying the rows and columns of a two-way contingency table as points in a low-dimensional space, such that the positions of the row and column points are consistent with their associations in the table. The goal is to have a global view of the data that is useful for interpretation.

The ViSta-Coresp visualization displays the graphs for the rows and columns of the contingency table, and lets you interactively modify the estimated values.

**APPENDIX: VISTA'S MENUS - The Analyze Menu - Homogeneity Analysis ... (A.18.11)**

Homogeneity Analysis (HOMALS) is a statistical visualization technique for picturing the associations between the levels of a set of categorical variables, and the similarities between the objects which these categories are applied too. The goal is to have a global view of the data that is useful for exploratory proposes. Only the categorical variables are included in the analysis.

Homogeneity Analysis assigns scores to the objects and it quantifies categories (optimal scaling). These scores and quantifications allow to construct a representation of the data esturcture in a low dimensional space (data reduction).

Categories and objects are represented as points in a joint space. The positions of the object points are related with their similarities. Objects with similar profile are located closely in the space. A category point is the centroid of the objects that belong to this category.

Homogeneity Analysis visualization displays the graphs for the category quantification and the objects scores, and let you interactively modify the dimensions of the HOMALS solution. Also, a plot for evaluating the fit of the solution and the discrimination measures of the variables is included.

**APPENDIX: VISTA'S MENUS - The Analyze Menu - Metric Averaged MDS ... (A.18.12)**

Multidimensional Scaling (ViSta-MDScal) is a data analysis and visualization technique for analyzing distance-like data (proximities, dissimilarities) and displaying a low-dimensional Euclidean space that summarizes the distance information in the data.

The data consist of distance-like information between all pairs of a set of objects. There may be several matrices

of such data, and the data matrices may be asymmetric or symmetric. The data must be dissimilarities, not similarities (i.e., large numbers mean large distance). There may not be any missing data.

ViSta-MDScal locates points in a Euclidean space which has a point for every object in the data. The points are located so that those that are close together have data which are similar, and those which are far apart have data which are different. The user must decide on the appropriate dimensionality of the Euclidean space.

The ViSta-MDScal visualization is based on the initial estimates of the best locations for the points. The fit of the points to the data can be improved by using the iterate button. The visualization provides several views of the multidimensional Euclidean space, as well as a fit plot to help decide on dimensionality.

#### APPENDIX: VISTA'S MENUS - The Analyze Menu - Multidimensional Scaling ... (A.18.13)

Multidimensional Scaling (ViSta-MDScal) is a data analysis and visualization technique for analyzing distance-like data (proximities, dissimilarities) and displaying a low-dimensional Euclidean space that summarizes the distance information in the data.

The data consist of distance-like information between all pairs of a set of objects. There may be several matrices of such data, and the data matrices may be asymmetric or symmetric. The data must be dissimilarities, not similarities (i.e., large numbers mean large distance). There may not be any missing data.

ViSta-MDScal locates points in a Euclidean space which has a point for every object in the data. The points are located so that those that are close together have data which are similar, and those which are far apart have data which are different. The user must decide on the appropriate dimensionality of the Euclidean space.

The ViSta-MDScal visualization is based on the initial estimates of the best locations for the points. The fit of the points to the data can be improved by using the iterate button. The visualization provides several views of the multidimensional Euclidean space, as well as a fit plot to help decide on dimensionality.

#### APPENDIX: VISTA'S MENUS - The Analyze Menu - Cluster Analysis ... (A.18.14)

No Help Available

#### APPENDIX: VISTA'S MENUS - The Analyze Menu - Impute Missing Data ... (A.18.15)

The IMPUTE MISSING DATA menu item is available to you when there are missing values in your data. Since data with missing values cannot be used in ViSta's analysis methods, some way of pre-processing the data must be provided. The menu item provides three of the most common methods used to deal with missing data:

- 1) Listwise deletion: Any observation with a missing value is deleted from the dataset.
- 2) Pairwise deletion: Correlation/covariance matrices are computed on the basis of cases which do not have pairs of missing values.
- 3) Maximum Likelihood: An iterative process attempts to obtain maximum likelihood estimates of the missing values. These estimates are used to replace the missing values so that no data has to be thrown out.

**APPENDIX: VISTA'S MENUS - The Model Menu - The Model Menu (A.19.1)**

This topic presents you with the help information for the items of the Model Menu. Choose an item to see its help.

**APPENDIX: VISTA'S MENUS - The Model Menu - About The Analysis (A.19.2)**

The ABOUT THESE DATA and ABOUT THE ANALYSIS menu items present information about data objects and analysis methods.

You can enter this information for your data objects. To do this:

1) Click on the data icon for which you wish to enter information.

2) Type:

(about-these-data "INFORMATION")

in the listener window, replacing INFORMATION with the information about the data that you wish to enter. The information must be inside double quote marks. If the information itself contains double quote marks, they must be preceded by a back-slash.

**APPENDIX: VISTA'S MENUS - The Model Menu - Visualize Model (A.19.3)**

The VISUALIZE DATA and VISUALIZE MODEL items produce a visualization of your data or model.

The data and model visualizations are designed to help you see what your data --- or the model of your data --- seem to say.

Visualizations consist of several plot windows that work together. The group of plot windows is called a SpreadPlot. Help about the spreadplot is available from the help menu's SPREADPLOT HELP item. In addition, help may be obtained for the individual plots by using the help menu's WINDOW HELP item. Finally, many plot windows have a help button which provides help about the individual plot window.

When you are finished with the visualization, use any of the close boxes. All of the windows will close together.

Each model has its own visualization. In addition, there are several different data visualizations, depending on the selection of variable types active at the time the visualization is chosen.

When all variables are numeric, there are four possible visualizations: A multivariate visualization for data with 3 or more numeric variables; a bivariate visualization for data with two numeric variables; a univariate visualization for data with one numeric variable; and a Guided Tour visualization for data which have 6 or more numeric variables.

Finally, there is a classification visualization for data which have a numeric variable and one or more category variables; a frequency visualization for frequency data (data which have numeric variables that specify frequency values); and a category visualization for category data (data which have one or more category variables and no numeric variables).

APPENDIX: VISTA'S MENUS - The Model Menu - Report Model ... (A.19.4)  
REPORT MODEL produces a listing of numeric information about your model.

The details of the report depend on which model you have chosen.

APPENDIX: VISTA'S MENUS - The Model Menu - Interpret Model (A.19.5)  
INTERPRET MODEL

The INTERPRET MODEL menu item provides an interpretation of your model. The interpretation is a paragraph describing the results of the analysis that you have done, and the assumptions underlying the results.

APPENDIX: VISTA'S MENUS - The Model Menu - Save Model As ... (A.19.6)  
SAVE MODEL saves your ViSta model as a file on your computer so that it can be used again at a later time.

If you are using guidemaps, then at this point you should save the model before proceeding. You do that by using the !! side of the SAVE MODEL button.

APPENDIX: VISTA'S MENUS - The Model Menu - Delete Model (A.19.7)  
DELETE MODEL deletes the current model object AND all associated objects, if any.

The current model object is the one highlighted on the workmap and checkmarked in the model menu.

Associated objects are:

- 1) All those "below" the current model object. An object is "below" the current model object if a path of connecting lines exists from the lower edge of the current model object to the "below" object. A "below" object may be a data object, model object, or analysis object.
- 2) Those "immediately above" the current model object. An object is "immediately above" the current model object if it is an analysis object which is connected directly to the upper edge of the model object. An "above" object may only be an analysis object.

APPENDIX: VISTA'S MENUS - The Model Menu - Create Data ... (A.19.8)  
CREATE DATA creates new data from the current model. For some models, many different kinds of data can be created, and dialog box will be presented for you to select the kinds you want. Other models cannot create new data. A message will tell you if this is the case.

#### APPENDIX: VISTA'S MENUS - The Model Menu - Print Data Analysis ... (A.19.9) PRINT DATA

This item prints a listing of your data on your printer.

#### APPENDIX: VISTA'S MENUS - The Options Menu - The Options Menu (A.20.1)

This topic presents you with the help information for the items of the Options Menu. Choose an item to see its help.

#### APPENDIX: VISTA'S MENUS - The Options Menu - Preferences ... (A.20.2)

The PREFERENCES menu item changes the way that ViSta works. The changes you make are saved and will be in effect when you use ViSta again. Preference settings include:

##### START/EXIT PREFERENCES

**HIDE XLISPSTAT** - Hide XLispStat window when checked. The XLispStat window contains a large version of the listener that is handy for programming.

**HIDE DESKTOP** - Hide DeskTop window when checked. This is useful if you want to make ViSta invisible at startup and wish to run a script which only shows analysis results.

**HIDE VISTA LOGO** - Hide ViSta Logo window when checked. Note that you can make ViSta startup completely invisibly by hiding all three startup windows.

**SPIN VISTA LOGO** - Show Spinning ViSta Logo when checked. Shows the initial splash screen with the Logo spinning.

**SHOW VISTA LOGO** - Show Static ViSta Logo when checked. Shows the initial splash screen with the Logo nopt spinning.

**SHOW EXIT DIALOG** - Shows "DO YOU WANT TO EXIT" dialog at exit.

##### DEVELOPER PREFERENCES

**DEVELOPER MODE** - Switch to the developer's mode. This shows a menu of tools that are useful for developers.

**VERBOSITY LEVEL** - Set ammount of listener output that helps with debugging.

## SESSION PREFERENCES

OPEN DATA SHOWS DATASHEET - Automatically see the data's datasheet when you first open it.

SHOW SPINPLOTS SPINNING - Show spinable plots spinning the first time they appear.

AUTOSHOW MODEL REPORTS - Automatically show reports when making a new model.

AUTOSHOW MODEL SPREADPLOTS - Automatically show the model's spreadplot after a model is constructed.

## APPENDIX: VISTA'S MENUS - The Options Menu - Run Excel ... (A.20.3)

### RUN EXCEL

This item runs Excel, adding a VISTA menu to Excel. The menu has items which can be configured by the experienced ViSta user. See the help item for these features.

## APPENDIX: VISTA'S MENUS - The Options Menu - Record Listener (A.20.4)

RECORD LISTENER lets you record all of the messages shown in the listener and all of the interactions that take place in the listener. You will be prompted for a file in which to save the recording.

## APPENDIX: VISTA'S MENUS - The Options Menu - Repair Font (A.20.5)

### REPAIR FONT

If text has unexpected spaces in it, or is otherwise awkwardly formatted, this item will reinstall the font which may correct the problem.

## APPENDIX: VISTA'S MENUS - The Options Menu - Renew FileTypes (A.20.6)

### RENEW FILETYPES

ViSta is supposed to run when you double click files with extensions .VDF, .VAF, or .VIS. If this does not happen the problem may be fixed if you use this item to renew the file associations.

#### APPENDIX: VISTA'S MENUS - The Options Menu - Refresh System (A.20.7) REFRESH SYSTEM

Refreshes the system by by stopping all dynamic graphics activities and by removing unused, error-containing user environments.

It is recommended that you use this item whenever ViSta becomes sluggish, or whenever the Listener window's prompt has a number in front of it.

#### HOW CAN VISTA BECOME SLUGISH?

ViSta may also become sluggish when there are too many dynamic graphics windows. Such windows, even when they have been minimized, use up your systems computing power. When many windows are contending for time to do their activity, and when there isn't sufficient computing power, the entire system slows down.

#### WHAT ARE USER ENVIRONMENTS?

A user enviroment is the set of conditions and states that exist in ViSta as a result of a user's interaction with ViSta.

#### WHY MORE THAN 1 ENVIRONMENT?

Normally, ViSta just runs one user environment. In this case the Listener window's prompt has no number in front of it.

However, after an error has occured, ViSta automatically generates a new, error-free user environment in addition to the user environment containing the error. ViSta then automatically places the user in the new error-free environment.

This provides all users with an automatic way arround the problem that caused the error, and provides the advanced user the option of investigating the cause of the error by returning to the errorful environment.

The number of additional user environments ViSta is running is indicated by a number in front of the Listener Window's prompt. Thus 2> indicates two user environments have been generated in addition to the one being used.

While this provides an automatic way of dealing with errors, ViSta will stop running if it attempts to run an additional environment when there is insufficient memory.

Thus, it is recommended that you use this item whenever the Listener window's prompt has a number in front of it.

#### APPENDIX: VISTA'S MENUS - The Options Menu - Developers Mode (A.20.8) DEVELOPERS MODE

Toggles between application developer's mode and user's mode. In developer's mode an additional menu of developer's tools is installed in the desktop menubar, and the automatic compile and make system is activated.

## APPENDIX: VISTA'S MENUS - The Help Menu - The Help Menu (A.21.1) HELP FOR THE HELP MENU

The HELP menu provides you access to ViSta's extensive help system

WELCOME TO VISTA presents a Help Panel with information for the new user who needs to get started using ViSta.

HELP TOPICS presents help information organized into topics. The topics are ordered in a way designed to help your understanding, and can be read like chapters in a book. Each topic has subtopics corresponding to sections of a chapter.

MENU ITEM HELP presents help for each menu item on the menubar.

DEMONSTRATIONS shows a dialog box of demonstrations.

VISTA ONLINE opens up your web browser to the ViSta WebSite's online help.

ABOUT VISTA shows you an animated window presenting information about ViSta and its authors.

## APPENDIX: VISTA'S MENUS - The DeskTop Menu - The DeskTop Menu (A.22.1) HELP FOR THE DESKTOP menu

The DESKTOP menu lets you set specifics about the appearance and operation of the DeskTop window. The changes you make are saved and will be in effect when you use ViSta again. The items include:

DESKTOP WINDOW  
XLISPSTAT WINDOW  
SPREADPLOT WINDOW  
REPORT WINDOW

These items display the specified window

MAXIMIZE WORKMAP  
MAXIMIZE LISTENER  
HIDE/SHOW SELECTOR  
RESTORE LAYOUT

The MAXIMIZE items effect the window panes of the desktop window by maximizing the specified pane to occupy the entire height of the DeskTop window. HIDE/SHOW SELECTOR hides or shows the selector pane. RESTORE LAYOUT item restores maximized panes to their size prior to maximization.

FULL SCREEN

Maximizes the desktop window so that it occupies the entire screen.

EXPLODE DESKTOP  
IMPLODE DESKTOP

Explode DeskTop separates the panes of the desktop window into separate windows. IMPLODE rejoins them.

HIDE DESKTOP

This item hides (minimizes) the desktop. You can show it again by clicking on the minimized window icon.

DEFAULT DESKTOP  
MINI DESKTOP

These items set the desktop to a default large and small configuration, respectively.

**DESKTOP OPTIONS:**

Use this to change the relative proportions of the various window-panes of the desktop, and to remove or add title bars to the window-panes.

**TOOLBAR BUTTONS:**

You can change the function (and number) of the buttons on the toolbar at the top of the workmap.

**DESKTOP COLORS:**

Changes the colors of windows, of tool and button bars, and of icons and graph elements. You can set the maximum number of colors to correspond with your hardware's capabilities.

**APPENDIX: VISTA'S MENUS - The Window Menu - The Window Menu (A.23.1)  
HELP FOR THE WINDOW and DESKTOP menus.**

The WINDOW menu lets you display ViSta's windows. The items include:

DESKTOP WINDOW  
XLISPSTAT WINDOW  
SPREADPLOT WINDOW  
REPORT WINDOW

These items display the specified window

Additional items appear as additional windows are created.