

AllHelp.txt

ViSta: The Visual Statistics System
Created by: Forrest W. Young
On the Web: www.visualstats.org
Copyright: Copyright © 1991-2002 by Forrest W. Young
Version: 7.0.7967 (EWU.2002.08.11.7967)

Forrest Young: forrest@visualstats.org
Pedro Valero-Mora: Pedro.Valero-Mora@uv.es
Gabriel Molina: Gabriel.Molina@uv.es

NEWSGROUP vista@unc.edu

ALL HELP FILES

PART I: WELCOME

- 1.0 Welcome User!
- 1.1 Welcome!
- 1.2 Vista Preview
- 2.0 Welcome Developer!
- 2.1 Welcome Developer!
- 2.2 Developer Features
- 2.3 Application Development
- 2.4 System Development

PART II: GETTING STARTED

- 3.0 Getting Started
- 3.1 Getting Started
- 3.2 Installing Vista
- 4.0 Using The Desktop
- 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
- 4.8 The Listener
- 5.0 Using Excel
- 5.1 Excel And Vista
- 5.2 From Excel To Vista
- 5.3 From Vista To Excel
- 5.4 Notes About Using Excel
- 5.5 Changing Excel'S Vista Menu
- 6.0 Entering Data
- 6.1 Editing Data
- 6.2 Vista'S Datasheet
- 6.3 Excel Spreadsheets
- 6.4 Sas Datasets
- 6.5 Importing Data
- 6.6 Simulating Data
- 6.7 Writing Datacode
- 7.0 Managing Data

- 7.1 Managing Data
- 7.2 Variable And Data Types
- 7.3 Manipulating Variables
- 7.4 Using Viva
- 7.5 Making Variables
- 7.6 Making Data

PART III: UNDERSTANDING DATA

- 8.0 Graphing Data
 - 8.1 Graphing Your Data
 - 8.2 The Box, Diamond And Dot Plots
 - 8.3 The Histogram And Frequency Plots
 - 8.4 Mosaic Plot And Bar Chart
 - 8.5 Scatterplot
 - 8.6 Spin Plot
 - 8.7 Scatterplot Matrix
 - 8.8 Tour Plot
- 9.0 Analyzing Data
 - 9.1 The Analyze Menu
 - 9.2 Analysis Of Variance ...
 - 9.3 Regression Analysis ...
 - 9.4 Univariate Analysis ...
 - 9.5 Frequency Analysis ...
 - 9.6 Log Linear Analysis ...
 - 9.7 Multivariate Regression ...
 - 9.8 Principal Components ...
 - 9.9 Item Analysis ...
 - 9.10 Correspondence Analysis ...
 - 9.11 Homogeneity Analysis ...
 - 9.12 Metric Averaged Mds ...
 - 9.13 Cluster Analysis ...
 - 9.14 Impute Missing Data ...
- 10.0 Modeling Data
 - 10.1 The Model Menu
 - 10.2 About The Analysis
 - 10.3 Visualize Model
 - 10.4 Report Model ...
 - 10.5 Interpret Model
 - 10.6 Delete Model
 - 10.7 Create Data ...

PART IV: ENHANCING VISTA

- 11.0 Enhancing Vista
 - 11.1 Enhancing Vista
 - 11.2 Application Development
 - 11.3 System Development
 - 11.4 Writing Spreadplots
 - 11.5 Writing Plugins
 - 11.6 About Datatypes
 - 11.7 About Filetypes

PART V: VISTA'S MENUS

- 12.0 About The Menus
 - 12.1 The Menu Bar
 - 12.2 Pop-Up Menus
- 13.0 The File Menu

- 13.1 The File Menu
- 13.2 New Data ...
- 13.3 Open Data ...
- 13.4 Simulate Data ...
- 13.5 Import Data ...
- 13.6 Export Data ...
- 13.7 Save Data As ...
- 13.8 Save Model As ...
- 13.9 New Edit...
- 13.10 Open Edit ...
- 13.11 Load Edit ...
- 13.12 Print File ...
- 13.13 Print Active Pane...
- 13.14 Print Entire Window
- 13.15 Exit
- 14.0 The Edit Menu
- 14.1 The Edit Menu
- 14.2 Cut
- 14.3 Copy
- 14.4 Paste
- 14.5 Clear
- 14.6 Copy-Paste
- 15.0 The Data Menu
- 15.1 The Data Menu
- 15.2 About These Data
- 15.3 Visualize Data ...
- 15.4 Summarize Data ...
- 15.5 Crosstabulate Data
- 15.6 Browse Data
- 15.7 List Data
- 15.8 Create Stats Object
- 15.9 Create-Convert Data
- 15.10 Delete Data Object ...
- 15.11 Impute Missing Data ...
- 15.12 Edit Data
- 15.13 Edit Source
- 15.14 Edit Variables
- 15.15 Print Data
- 15.16 Print Source
- 15.17 Merge Variables
- 15.18 Merge Observations
- 15.19 Merge Matrices
- 16.0 The Transform Menu
- 16.1 The Transform Menu
- 16.2 Sort-Permute ...
- 16.3 Ranks
- 16.4 Normal Scores
- 16.5 Absolute Value
- 16.6 Rounding ...
- 16.7 Reciprocal
- 16.8 Exponential ...
- 16.9 Logarithm ...
- 16.10 Trigonometric ...
- 16.11 Correlations
- 16.12 Covariances
- 16.13 Distances
- 16.14 Transpose Data
- 16.15 Standardize Data ...
- 16.16 Orthogonalize Data

- 16.17 Folded Power
- 16.18 Box-Cox Power
- 16.19 Dummy Coding
- 16.20 Split By Categories
- 16.21 Join Category Variables
- 16.22 Program Editor
- 17.0 The Analyze Menu
 - 17.1 The Analyze Menu
 - 17.2 Analysis Of Variance ...
 - 17.3 Regression Analysis ...
 - 17.4 Univariate Analysis ...
 - 17.5 Frequency Analysis ...
 - 17.6 Log Linear Analysis ...
 - 17.7 Multivariate Regression ...
 - 17.8 Principal Components ...
 - 17.9 Item Analysis ...
 - 17.10 Correspondence Analysis ...
 - 17.11 Homogeneity Analysis ...
 - 17.12 Metric Averaged Mds ...
 - 17.13 Cluster Analysis ...
 - 17.14 Impute Missing Data ...
- 18.0 The Model Menu
 - 18.1 The Model Menu
 - 18.2 About The Analysis
 - 18.3 Visualize Model
 - 18.4 Report Model ...
 - 18.5 Interpret Model
 - 18.6 Delete Model
 - 18.7 Create Data ...
- 19.0 The Options Menu
 - 19.1 The Options Menu
 - 19.2 Preferences ...
 - 19.3 Run Excel ...
 - 19.4 Show Clock
 - 19.5 Clock Options
 - 19.6 Twiddle
 - 19.7 Record Listener
 - 19.8 Refresh System
 - 19.9 Developers Mode
- 20.0 The Help Menu
 - 20.1 The Help Menu
- 21.0 The Desktop Menu
 - 21.1 The Desktop Menu
- 22.0 The Window Menu
 - 22.1 The Window Menu

1.1: TOPIC: WELCOME User! - SUBTOPIC: WELCOME!

WELCOME to ViSta !

ViSta - The Visual Statistics System - is ready to help you "Have fun seeing what your data seem to say".

AllHelp.txt

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 - YOUR PERSONAL STATISTICIAN!

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 - YOUR STATS VIDEO GAME!

ViSta is your statistics 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.

HAVE FUN! LEARN LOTS!

Once again: Welcome To ViSta! Have fun with your data, and learn a lot!

Forrest W. Young

1.2: TOPIC: WELCOME User! - SUBTOPIC: ViSta Preview

AllHelp.txt

Welcome to ViSta 7!

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.

2.1: TOPIC: WELCOME Developer! - SUBTOPIC: WELCOME Developer!

WELCOME! ... Welcome to the ViSta Development Team!

and

THANKS! ... 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.

Forrest

2.2: TOPIC: WELCOME Developer! - SUBTOPIC: Developer Features

YOUR NEW DEVELOPER'S ENVIRONMENT IS READY

Your developer's tools have been loaded, and ViSta's environment has been changed

into a developer's

environment. The changes are summarized below.

There are two more menus on the menubar: Graphics and Develop. The Develop menu contains the tools for

developing new applications for ViSta. The graphics menu is a preview of things to come, but it has been

disabled and is not yet ready to use.

There is a "Toggle Devel Mode" item at the end of the Options menu which toggle you in and out of

developer's mode.

When you exit the make-vista system will automatically check for code changes, and attempt to compile any

file whose time stamp has changed. If the compile fails, you will have to fix the file and try again. If

it succeeds, then the make-vista system will form a new vista workspace. (This step is not always needed,

and is not always carried out.)

When you rerun ViSta, you will be once again introduced to ViSta, since the entire sytem has to be

reinitialized so that everything works as expected.

Note that a new icon has been added to Window's START menu. It allows you to make ViSta without having to

initially run ViSta.

2.3: TOPIC: WELCOME Developer! - SUBTOPIC: Application Development

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

AllHelp.txt

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 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 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. You are, of

course, free to distribute your app independently from 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

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 website (and on our mirror sites), and links will be made so that it

can be downloaded.

2.4: TOPIC: WELCOME Developer! - SUBTOPIC: System Development

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.

3.1: TOPIC: Getting Started - SUBTOPIC: Getting Started

GETTING STARTED:

AllHelp.txt

The 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.

ARE YOU A NEW USER?

If so, read all of the subtopics of the help topics on

- ViSta OnLine
- Getting Started
- Using The DeskTop
- The Data Menu

They will familiarize you with ViSta and its capabilities.

ARE YOU AN EXPERIENCED USER?

Go ahead! Jump right in!

DO YOU PLAN TO USE EXCEL AND VISTA TOGETHER?

Then the material about USING EXCEL is for you. You may also need to read the items mentioned above.

3.2: TOPIC: Getting Started - SUBTOPIC: Installing ViSta

INSTALLING VISTA - SINGLE USER and NETWORK INSTALL

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.

INSTALLING VISTA - NETWORK INSTALL

Follow the steps given above. At the least step, choose ALL OTHER SITUATIONS, you see dialog boxes. Do

this:

- 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.

SETTING VISTA INITIAL WINDOW SIZE/LOCATION FOR NT SERVERS

There are two alternate approaches to setting window location/size. The first approach is simplest.

FIRST APPROACH

For NT when ViSta is being setup to serve clients without special privileges, the steps to create default window location/size are:

- 1) logon to NT as administrator.
- 2) start Vista with command line option -useHKLM.
- 3) set Vista main window the way users will see it on startup.

SECOND APPROACH

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

Here is what ViSta does with HKLM and HCU on startup and shutdown:

Reading on startup:

CASE - Entry in HKLM	Entry in HCU	Behavior
yes	No	use entries in HKLM
yes	yes	use entries in HKLM

		AllHelp.txt
no	yes	use entries in HCU
no	no	provide default window size

Writing on shutdown:

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

4.1: TOPIC: Using the DeskTop - SUBTOPIC: The DeskTop

ABOUT VISTA'S DESKTOP

ViSta's DeskTop window has three window-panes. By default, you see all 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 are described briefly below. More information is available by reading appropriate

subtopics of the USING THE DESKTOP help topic.

WORKMAP

The upper-left window-pane is 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.

SELECTOR

At the upper-right is the SELECTOR, which itself consists of two small window-panes. 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 subsets of the data.

LISTENER

The Listener window-pane displays messages about the system, and lets you type commands in ViSta's

underlying languages.

4.2: TOPIC: Using the DeskTop - SUBTOPIC: The Menu Bar

ABOUT VISTA'S MENUBAR MENUS:

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

- File menus: FILE and EDIT
- Analysis menus: DATA, 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 Menus DATA - TRANSFORM - ANALYZE - MODEL

The four statistics menus are the main data analysis menus. They are also the main four steps in

understanding 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 four 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 about them.

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.

4.3: TOPIC: Using the DeskTop - SUBTOPIC: Pop-Up Menus

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.

4.4: TOPIC: Using the DeskTop - SUBTOPIC: The ToolBar

ABOUT VISTA'S WORKMAP 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.

4.5: TOPIC: Using the DeskTop - SUBTOPIC: The WorkMap

ABOUT VISTA'S WORKMAP

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.

4.6: TOPIC: Using the DeskTop - SUBTOPIC: The DataSheet

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

AllHelp.txt

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 $(1 +$

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.

4.7: TOPIC: Using the DeskTop - SUBTOPIC: The Selector

The SELECTOR displays 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.

4.8: TOPIC: Using the DeskTop - SUBTOPIC: The Listener

The Listener Window displays messages about ViSta's operation and provides a place for typing statements

in the underlying Lisp language, and in ViSta's ViVa language.

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.

AllHelp.txt

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

5.1: TOPIC: Using Excel - SUBTOPIC: Excel and ViSta

USING EXCEL AND VISTA TOGETHER

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 changing Excel's ViSta

menu.

5.2: TOPIC: Using Excel - SUBTOPIC: From Excel to ViSta

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

AllHelp.txt

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.

5.3: TOPIC: Using Excel - SUBTOPIC: From ViSta to Excel

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

AllHelp.txt

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.

5.4: TOPIC: Using Excel - SUBTOPIC: Notes about Using Excel

NOTES ABOUT USING EXCEL WITH VISTA

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

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:

AllHelp.txt

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.

5.5: TOPIC: Using Excel - SUBTOPIC: Changing Excel's ViSta Menu

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:

AllHelp.txt

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.

6.1: TOPIC: Entering Data - SUBTOPIC: Editing Data

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

unobtrusitvely 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.

6.2: TOPIC: Entering Data - SUBTOPIC: ViSta's DataSheet

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

AllHelp.txt

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 $(1 +$

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

AllHelp.txt

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.

6.3: TOPIC: Entering Data - SUBTOPIC: Excel SpreadSheets

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 MENUIf 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

AllHelp.txt

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

AllHelp.txt

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.

6.4: TOPIC: Entering Data - SUBTOPIC: SAS Datasets

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.

AllHelp.txt

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");

AllHelp.txt

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-etal:75) );
```

The output file, berkeley.lsp looks like this:

```
(DATA "berkeley"
  :TITLE "berkeley"
  :ABOUT "Berkeley admissions data, from Bickel-etal:75"
  :FREQ t
  :VARIABLES (QUOTE ( "Gender" "Dept" "Admit" "Freq"))
```

AllHelp.txt

```
: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
))
)
```

6.5: TOPIC: Entering Data - SUBTOPIC: Importing Data

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 it as a datasheet.

A dialog box will ask for the name of the data, whether the first line in the file contains variable

names, whether the second line contains variable types, whether the first column contains observation

labels, whether you want to see the data sheet, and whether you need to check missing data.

AllHelp.txt

When the first line of the file specifies variable names, there must be one name for each variable, each

one separated by white space. If the first variable name is LABELS, or if the "First Column is Labels"

dialog item is checked, then the first "variable" will be used as observation labels. Variable names are

unquoted strings. Optionally, they may be inside double quotes, permitting white space inside the name.

If the first line specifies variable names, then the second one can specify variable types, if desired.

This can greatly speed up importation for large data. Variable types must be one of the following double

quoted strings: "label" "labels" "category" "numeric". Case is ignored. If the second line does not

specify variable types, then a variable that has one or more non-numeric values is treated as a category

variable, otherwise variables are assumed to be numeric.

When the first line does not contain variable names it must contain the first data line. The line must

have one data value for each variable.

Choosing to not see the datasheet can greatly improve the speed for importing large data.

If there are no missing values, or if missing values are all coded by the symbol NIL, and none are coded

by the string "NIL", then you do not need to check missing data, a process which can be time-consuming.

Data values must be separated by white space. If variable names are not specified, then the number of data

values in the first record determines the number of variables in the data object. Succeeding records must

have the same number of data values.

Data values may be numeric or non-numeric. Non-numeric values may optionally be in double quotes. Double

quotes permit entry of spaces, and preserve character case (unquoted non-numeric values are converted to

upper case). Missing values must be coded as nil or "nil".

After you import data you will see a datasheet of your data. You may use this datasheet to browse through

your data and to edit your data.

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 you entered into the dialog box.

If you are using guidemaps, the GET YOUR DATA guidemap will be replaced by the DATA ANALYSIS guidemap.

6.6: TOPIC: Entering Data - SUBTOPIC: Simulating Data

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

AllHelp.txt

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.

6.7: TOPIC: Entering Data - SUBTOPIC: Writing DataCode

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 $(1 +$

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.

7.1: TOPIC: Managing Data - SUBTOPIC: Managing Data

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

AllHelp.txt

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.

AllHelp.txt

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.

7.2: TOPIC: Managing Data - SUBTOPIC: Variable and Data Types

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.

AllHelp.txt

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:

AllHelp.txt

- 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	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:

- 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.

7.3: TOPIC: Managing Data - SUBTOPIC: Manipulating Variables

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" ...)  
)
```

AllHelp.txt

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.

7.4: TOPIC: Managing Data - SUBTOPIC: Using ViVa

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.

AllHelp.txt

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. ViVa 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
```

AllHelp.txt

```
? 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)

```

7.5: TOPIC: Managing Data - SUBTOPIC: Making Variables

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.

AllHelp.txt

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: blff10, 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
```

AllHelp.txt

```
> (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
>
```

7.6: TOPIC: Managing Data - SUBTOPIC: Making Data

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

AllHelp.txt

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.

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 typeing:

(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).

8.1: TOPIC: Graphing Data - SUBTOPIC: Graphing Your Data

This topic presents you with the help information for the items of the Graphics Menu. Choose an item to

see its help.

8.2: TOPIC: Graphing Data - SUBTOPIC: The Box, Diamond and Dot Plots

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.

8.3: TOPIC: Graphing Data - SUBTOPIC: The Histogram and Frequency Plots

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

8.4: TOPIC: Graphing Data - SUBTOPIC: Mosaic Plot and Bar Chart

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.

8.5: TOPIC: Graphing Data - SUBTOPIC: Scatterplot

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.

8.6: TOPIC: Graphing Data - SUBTOPIC: Spin Plot

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.

8.7: TOPIC: Graphing Data - SUBTOPIC: Scatterplot Matrix

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.

8.8: TOPIC: Graphing Data - SUBTOPIC: Tour Plot

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.

AllHelp.txt

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

AllHelp.txt

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.

9.1: TOPIC: Analyzing Data - SUBTOPIC: The Analyze Menu

This topic presents you with the help information for the items of the Analyze Menu. Choose an item to see

its help.

9.2: TOPIC: Analyzing Data - SUBTOPIC: Analysis of Variance ...

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.

9.3: TOPIC: Analyzing Data - SUBTOPIC: Regression Analysis ...

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.

9.4: TOPIC: Analyzing Data - SUBTOPIC: Univariate Analysis ...

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.

9.5: TOPIC: Analyzing Data - SUBTOPIC: Frequency Analysis ...

Help is not yet available.

9.6: TOPIC: Analyzing Data - SUBTOPIC: Log linear analysis ...

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

9.7: TOPIC: Analyzing Data - SUBTOPIC: Multivariate Regression ...

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).

9.8: TOPIC: Analyzing Data - SUBTOPIC: Principal Components ...

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

AllHelp.txt

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.

9.9: TOPIC: Analyzing Data - SUBTOPIC: Item Analysis ...

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.

9.10: TOPIC: Analyzing Data - SUBTOPIC: Correspondence Analysis ...

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.

9.11: TOPIC: Analyzing Data - SUBTOPIC: Homogeneity Analysis ...

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 structure 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.

9.12: TOPIC: Analyzing Data - SUBTOPIC: Metric Averaged MDS ...

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

AllHelp.txt

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.

9.13: TOPIC: Analyzing Data - SUBTOPIC: Cluster Analysis ...

No Help Available

9.14: TOPIC: Analyzing Data - SUBTOPIC: Impute Missing Data ...

AllHelp.txt

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.

10.1: TOPIC: Modeling Data - SUBTOPIC: The Model Menu

This topic presents you with the help information for the items of the Model Menu. Choose an item to see

its help.

10.2: TOPIC: Modeling Data - SUBTOPIC: About The Analysis

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.

AllHelp.txt

The information must be inside double quote marks. If the information itself contains double quote marks,

they must be preceded by a back-slash.

10.3: TOPIC: Modeling Data - SUBTOPIC: Visualize Model

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.

AllHelp.txt

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).

10.4: TOPIC: Modeling Data - SUBTOPIC: Report Model ...

REPORT MODEL produces a listing of numeric information about your model.

The details of the report depend on which model you have chosen.

10.5: TOPIC: Modeling Data - SUBTOPIC: Interpret Model

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.

10.6: TOPIC: Modeling Data - SUBTOPIC: Delete Model

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.

10.7: TOPIC: Modeling Data - SUBTOPIC: Create Data ...

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.

11.1: TOPIC: Enhancing ViSta - SUBTOPIC: Enhancing ViSta

ENHANCING ViSta:

ViSta is an OPEN software system. That means that 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.

11.2: TOPIC: Enhancing ViSta - SUBTOPIC: Application Development

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 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 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. You are, of

course, free to distribute your app independently from visualstats.org.

AllHelp.txt

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

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 website (and on our mirror sites), and links will be made so that it

can be downloaded.

11.3: TOPIC: Enhancing ViSta - SUBTOPIC: System Development

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.

11.4: TOPIC: Enhancing ViSta - SUBTOPIC: Writing Spreadplots

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

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 ascetically

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:

AllHelp.txt

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 preceding 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

AllHelp.txt

:UPDATE-SPREADPLOT (i j &rest args)

to the spreadplot. 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."

11.5: TOPIC: Enhancing ViSta - SUBTOPIC: Writing Plugins

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
```

AllHelp.txt

```
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 Homogeneity analysis and Averaged Metric 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-constructor-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.

AllHelp.txt

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.

It's 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-constructor-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-constructor 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
```

```

                                AllHelp.txt
                                (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))
      (lcol (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)
      )))

```

11.6: TOPIC: Enhancing ViSta - SUBTOPIC: About DataTypes

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 defintions 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:

- a) TABLE datatype is no longer used.
- b) ORDINAL variables are treated as NUMERIC.
- c) 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.

- d) In addition, datatype may be temporarily set to "EnAbld" "DisAbld" "ReEnAbld".

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.

AllHelp.txt

For a help message about the datatype aspects of the current dataobject, type
(current-datatype)

11.7: TOPIC: Enhancing ViSta - SUBTOPIC: About FileTypes

About File Types (aka: mimetypes)

The ViSta system includes the executables vista.exe, lspedit.exe and lispboss.exe.

When a file associated with lispboss.exe is double-clicked, LispBoss determines whether the file should be

processed by vista.exe or lspedit.exe.

Since Lispboss is the primary processor, all of the filetypes discussed here must be associated with

Lispboss. Thus, during installation, your computer's operating environment is modified so that the

filetypes are all associated with LISPBOSS (except .WKS files, which are associated with ViSta.exe). This

is done by the filetype.bat file, which resides in the ViSta home directory. On my NT2000 machine, the

filetype.bat file contains:

```
assoc .lsp=XLispStat
assoc .vdf=ViStaDataFile
assoc .vaf=ViStaAppletFile
assoc .vis=ViStaProgramFile
assoc .fsl=XLispCompiledCode
assoc .wks=XLispWorkSpace
ftype XLispStat="D:\CVS\ViSta7\LispBoss\lispboss.exe" "%*1"
ftype ViStaDataFile="D:\CVS\ViSta7\LispBoss\lispboss.exe" "%*1"
ftype ViStaAppletFile="D:\CVS\ViSta7\LispBoss\lispboss.exe" "%*1"
ftype ViStaProgramFile="D:\CVS\ViSta7\LispBoss\lispboss.exe" "%*1"
ftype XLispCompiledCode="D:\CVS\ViSta7\LispBoss\lispboss.exe" "%*1"
ftype XLispWorkSpace="D:\CVS\ViSta7\ViSta.exe" "%*1"
```

LispBoss decides whether a file should be Loaded by ViSta.exe or edited by LspEdit.exe. It makes this

decision by using the extension of the file along with other information. The file extensions, the

additional information, and resulting LispBoss actions are given in the remainder of

this help file.

.LSP - Lisp File.

This file type must contain Lisp code appropriate for editing or evaluating. Whether the file is loaded by

ViSta or edited by LspEdit depends on the value of the environmental variable *edit-lisp-files* and on

whether the file is a datafile. In particular, there are two different .LSP filetypes.

.LSP Datafiles

This file type is always loaded by ViSta, regardless of *edit-lisp-files*. Currently, a LSP file is

defined to be a datafile if it is in a directory whose path contains a directory named DATA, MYDATA or

MY-DATA (capitalization ignored). A more sophisticated way to define a LSP ViSta Data File than on the

basis of its directory name is needed. Identical to a .VDF datafile.

.LSP Progfiles

Progfiles are LSP files that are not datafiles. These files are loaded by ViSta or edited by LspEdit

depending on the value of *edit-lisp-files*

.VAF - ViSta Applet File.

.VIS - ViSta Applet File.

These two file types must contain Lisp programs appropriate for evaluating by ViSta. It is assumed that

ViSta resides on a client computer. The file can reside on the local client computer or can be served to

the local client computer by a remote server.

.VDF - ViSta Data File.

This file type uses ViSta to load and run ViSta datafiles. This file type must contain ViSta DataCode

programs appropriate for evaluating by ViSta. It is assumed that ViSta resides on a client computer. The

file can reside on the local client computer or can be served to the local client computer by a remote

server. Identical to a .LSP datafile.

.FSL - XLispStat L-Code file.

AllHelp.txt

This file type uses ViSta to load compiled bytecode files. The compiled bytecode (L-code) can result from

compiling any of the file types above.

.WKS - XLispStat Workspace file.

This file type uses ViSta to load an XLispStat workspace.

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.

12.1: TOPIC: About The Menus - SUBTOPIC: The Menu Bar

ABOUT VISTA'S MENUBAR MENUS:

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

- File menus: FILE and EDIT
- Analysis menus: DATA, 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 Menus
DATA - TRANSFORM - ANALYZE - MODEL

The four statistics menus are the main data analysis menus. They are also the main four steps in

understanding 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 four 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 about them.

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.

12.2: TOPIC: About The Menus - SUBTOPIC: Pop-Up Menus

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.

13.1: TOPIC: The File Menu - SUBTOPIC: The File Menu

This topic presents you with the help information for the items of the File Menu. Choose an item to see

its help.

13.2: TOPIC: The File Menu - SUBTOPIC: New Data ...

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 specifies 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.

13.3: TOPIC: The File Menu - SUBTOPIC: Open Data ...

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.

13.4: TOPIC: The File Menu - SUBTOPIC: Simulate Data ...

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.

13.5: TOPIC: The File Menu - SUBTOPIC: Import Data ...

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 it as a datasheet.

A dialog box will ask for the name of the data, whether the first line in the file contains variable

names, whether the second line contains variable types, whether the first column contains observation

labels, whether you want to see the data sheet, and whether you need to check missing data.

When the first line of the file specifies variable names, there must be one name for each variable, each

one separated by white space. If the first variable name is LABELS, or if the "First Column is Labels"

dialog item is checked, then the first "variable" will be used as observation labels. Variable names are

unquoted strings. Optionally, they may be inside double quotes, permitting white space inside the name.

If the first line specifies variable names, then the second one can specify variable types, if desired.

This can greatly speed up importation for large data. Variable types must be one of the following double

AllHelp.txt

quoted strings: "label" "labels" "category" "numeric". Case is ignored. If the second line does not

specify variable types, then a variable that has one or more non-numeric values is treated as a category

variable, otherwise variables are assumed to be numeric.

When the first line does not contain variable names it must contain the first data line. The line must

have one data value for each variable.

Choosing to not see the datasheet can greatly improve the speed for importing large data.

If there are no missing values, or if missing values are all coded by the symbol NIL, and none are coded

by the string "NIL", then you do not need to check missing data, a process which can be time-consuming.

Data values must be separated by white space. If variable names are not specified, then the number of data

values in the first record determines the number of variables in the data object. Succeeding records must

have the same number of data values.

Data values may be numeric or non-numeric. Non-numeric values may optionally be in double quotes. Double

quotes permit entry of spaces, and preserve character case (unquoted non-numeric values are converted to

upper case). Missing values must be coded as nil or "nil".

After you import data you will see a datasheet of your data. You may use this datasheet to browse through

your data and to edit your data.

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 you entered into the dialog box.

If you are using guidemaps, the GET YOUR DATA guidemap will be replaced by the DATA ANALYSIS guidemap.

13.6: TOPIC: The File Menu - SUBTOPIC: Export Data ...

EXPORT DATA exports your ViSta data as a text file containing your data in a form that can be imported

into other programs (or can be reimported into ViSta).

The standard SAVE dialog box will appear to identify where you wish to save your data. A second dialog box

will appear and ask whether you wish to export variable names and observation labels.

If you choose to export variable names, the first line of the file will contain variable names. There will

be one name for each active variable, each name separated by white space. If you also choose to export

observation labels, then the first variable name is LABELS.

If you do not choose to export variable names, the first line will contain the first observation, the

second will contain the second observation, etc. The line will have one data value for each active

variable. The values will be separated by white space. If you choose to export observation labels, the

first value on each line will be the observation label.

13.7: TOPIC: The File Menu - SUBTOPIC: Save Data As ...

SAVE DATA saves your ViSta data as a file on your computer so that it can be used again after you finish

using ViSta.

If you are using guidemaps, then at this point you should save the active dataset before proceeding. You

do that by using the !! side of the SAVE DATA button.

13.8: TOPIC: The File Menu - SUBTOPIC: Save Model As ...

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.

13.9: TOPIC: The File Menu - SUBTOPIC: New Edit...

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.

13.10: TOPIC: The File Menu - SUBTOPIC: Open Edit ...

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.

13.11: TOPIC: The File Menu - SUBTOPIC: Load Edit ...

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.

13.12: TOPIC: The File Menu - SUBTOPIC: Print File ...

PRINT FILE

The PRINT FILE menu item opens a dialog box for you to select a file, which is then printed.

13.13: TOPIC: The File Menu - SUBTOPIC: Print Active Pane...

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.

13.14: TOPIC: The File Menu - SUBTOPIC: Print Entire Window

PRINT ENTIRE WINDOW

The PRINT ENTIRE WINDOW menu item prints an image of the active window.
A window is activated by clicking on it.

13.15: TOPIC: The File Menu - SUBTOPIC: Exit

EXIT

The EXIT menu item terminates your session with ViSta. Please make sure that you have saved your work.

14.1: TOPIC: The Edit Menu - SUBTOPIC: The Edit Menu

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.

14.2: TOPIC: The Edit Menu - SUBTOPIC: Cut

CUT

AllHelp.txt

Cuts selected text from the Listener window's prompt line.

14.3: TOPIC: The Edit Menu - SUBTOPIC: Copy

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.

14.4: TOPIC: The Edit Menu - SUBTOPIC: Paste

PASTE

Pastes text into the Listener window's prompt line.

14.5: TOPIC: The Edit Menu - SUBTOPIC: Clear

CLEAR

AllHelp.txt

Clears selected text from the Listener window's prompt line.

14.6: TOPIC: The Edit Menu - SUBTOPIC: Copy-Paste

COPY-PASTE

Copies selected Listener text to the Listener's prompt line.

15.1: TOPIC: The Data Menu - SUBTOPIC: The Data Menu

This topic presents you with the help information for the items of the Data Menu. Choose an item to see

its help.

15.2: TOPIC: The Data Menu - SUBTOPIC: About These Data

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

AllHelp.txt

contains double quote marks,
they must be preceded by a back-slash.

15.3: TOPIC: The Data Menu - SUBTOPIC: Visualize Data ...

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).

15.4: TOPIC: The Data Menu - SUBTOPIC: Summarize Data ...

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.

15.5: TOPIC: The Data Menu - SUBTOPIC: CrossTabulate Data

Sorry, no help is available for this topic.

15.6: TOPIC: The Data Menu - SUBTOPIC: Browse Data

AllHelp.txt

BROWSE DATA displays a datasheet of your data, so that you can look at your data. Double-clicking on a

data icon does the same thing as BROWSE DATA. OPEN DATA loads in a new data file and then does a BROWSE

DATA to show the newly opened data.

When the datasheet is shown in this fashion, it is not editable. It may be made editable by using the EDIT

DATA menu item.

The left column of cells contains observation labels.
The top row of cells contains variable names.
The second row of cells contains variable types.
The remaining cells of the datasheet contain the data.

You can control the width of columns and the number of decimal places shown by using the DataSheet menu.

You may analyze your data while an uneditable datasheet is open (you cannot analyze data while an editable

datasheet is open).

More extensive help is available for the datasheet when it is open by making it the active window and

using the HELP menu's WINDOW HELP item.

15.7: TOPIC: The Data Menu - SUBTOPIC: List Data

LIST DATA

The List Data menu item produces a window containing a listing of your active data.

15.8: TOPIC: The Data Menu - SUBTOPIC: Create Stats Object

The CREATE STATS OBJECT menu item creates a new data object which contains the summary statistics for the current data object.

15.9: TOPIC: The Data Menu - SUBTOPIC: Create-Convert Data

CREATE-CONVERT DATA creates new data from the current data. In appropriate circumstances it also provides

you with the option to convert between the various kinds of Frequency Data.

If data are created from existing data, CREATE DATA creates a new dataset which can have a SUBSET of the

data in the current dataset. You determine which subset of the data will be placed in the new dataset by

selecting variables and observations with the selector.

Note that CATEGORY, CLASSIFICATION, FREQUENCY CLASSIFICATION and FREQUENCY data can be converted back and

forth with the CREATE-CONVERT DATA menu item. It will automatically ask you what type of conversion, if

any, you want to do. Note that FREQUENCY TABLE data (the kind used by CORRESP) can only be created when the

classification is two-way.

See the help for the NEW DATA menu item for more about data types.

15.10: TOPIC: The Data Menu - SUBTOPIC: Delete Data Object ...

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:

AllHelp.txt

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 "emmediately above" the current data object. An object is "emmediately 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.

15.11: TOPIC: The Data Menu - SUBTOPIC: Impute Missing Data ...

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.

15.12: TOPIC: The Data Menu - SUBTOPIC: Edit Data

EDIT DATA

AllHelp.txt

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 +

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.

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 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

```

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
```

AllHelp.txt

```
? 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. ViVa 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)
```

AllHelp.txt

```
? 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)
```

AllHelp.txt

```
#<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)
```

15.15: TOPIC: The Data Menu - SUBTOPIC: Print Data

PRINT DATA

This item prints a listing of your data on your printer.

15.16: TOPIC: The Data Menu - SUBTOPIC: Print Source

PRINT DATA

This item prints the source code underlying your dataobject on your printer.

15.17: TOPIC: The Data Menu - SUBTOPIC: Merge Variables

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.

15.18: TOPIC: The Data Menu - SUBTOPIC: Merge Observations

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.

15.19: TOPIC: The Data Menu - SUBTOPIC: Merge Matrices

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.

16.1: TOPIC: The Transform Menu - SUBTOPIC: The Transform Menu

This topic presents you with the help information for the items of the Transform Menu. Choose an item to

see its help.

16.2: TOPIC: The Transform Menu - SUBTOPIC: Sort-Permute ...

The Sort-Permute transformation creates a new dataset in which the observations of a selected variable in

AllHelp.txt

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.

16.3: TOPIC: The Transform Menu - SUBTOPIC: Ranks

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.

16.4: TOPIC: The Transform Menu - SUBTOPIC: Normal Scores

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.

16.5: TOPIC: The Transform Menu - SUBTOPIC: Absolute Value

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.

16.6: TOPIC: The Transform Menu - SUBTOPIC: Rounding ...

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.

16.7: TOPIC: The Transform Menu - SUBTOPIC: Reciprocal

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.

16.8: TOPIC: The Transform Menu - SUBTOPIC: Exponential ...

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.

16.9: TOPIC: The Transform Menu - SUBTOPIC: Logarithm ...

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.

16.10: TOPIC: The Transform Menu - SUBTOPIC: Trigonometric ...

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.

16.11: TOPIC: The Transform Menu - SUBTOPIC: Correlations

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.

16.12: TOPIC: The Transform Menu - SUBTOPIC: Covariances

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.

16.13: TOPIC: The Transform Menu - SUBTOPIC: Distances

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.

16.14: TOPIC: The Transform Menu - SUBTOPIC: Transpose Data

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.

16.15: TOPIC: The Transform Menu - SUBTOPIC: Standardize Data ...

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.

16.16: TOPIC: The Transform Menu - SUBTOPIC: Orthogonalize Data

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.

16.17: TOPIC: The Transform Menu - SUBTOPIC: Folded Power

The Folded Power transformation creates a new dataset where the variables have been transformed according

to the following transformation:

$$\frac{(2Y)^p - (2-2Y)^p}{2^p} \quad \text{for } p \neq 0$$
$$\frac{\log(Y) - \log(1-Y)}{2} \quad \text{for } p = 0$$

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.

16.18: TOPIC: The Transform Menu - SUBTOPIC: Box-Cox Power

The Box Cox Power transformation creates a new dataset where the variables have been transformed according

to the following transformation:

$(Y^p - 1)/p$ for p not equal to 0

$\log(Y)$ for p 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

16.19: TOPIC: The Transform Menu - SUBTOPIC: Dummy coding

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.

16.20: TOPIC: The Transform Menu - SUBTOPIC: Split By Categories

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.

16.21: TOPIC: The Transform Menu - SUBTOPIC: Join Category Variables

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.

16.22: TOPIC: The Transform Menu - SUBTOPIC: Program Editor

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.

17.1: TOPIC: The Analyze Menu - SUBTOPIC: The Analyze Menu

This topic presents you with the help information for the items of the Analyze Menu. Choose an item to see

its help.

17.2: TOPIC: The Analyze Menu - SUBTOPIC: Analysis of Variance ...

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.

17.3: TOPIC: The Analyze Menu - SUBTOPIC: Regression Analysis ...

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.

17.4: TOPIC: The Analyze Menu - SUBTOPIC: Univariate Analysis ...

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.

17.5: TOPIC: The Analyze Menu - SUBTOPIC: Frequency Analysis ...

Help is not yet available.

17.6: TOPIC: The Analyze Menu - SUBTOPIC: Log linear analysis ...

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

17.7: TOPIC: The Analyze Menu - SUBTOPIC: Multivariate Regression ...

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).

17.8: TOPIC: The Analyze Menu - SUBTOPIC: Principal Components ...

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.

17.9: TOPIC: The Analyze Menu - SUBTOPIC: Item Analysis ...

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.

17.10: TOPIC: The Analyze Menu - SUBTOPIC: Correspondence Analysis ...

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.

17.11: TOPIC: The Analyze Menu - SUBTOPIC: Homogeneity Analysis ...

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 structure 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.

AllHelp.txt

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.

17.12: TOPIC: The Analyze Menu - SUBTOPIC: Metric Averaged MDS ...

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.

17.13: TOPIC: The Analyze Menu - SUBTOPIC: Cluster Analysis ...

No Help Available

17.14: TOPIC: The Analyze Menu - SUBTOPIC: Impute Missing Data ...

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.

18.1: TOPIC: The Model Menu - SUBTOPIC: The Model Menu

AllHelp.txt

This topic presents you with the help information for the items of the Model Menu. Choose an item to see

its help.

18.2: TOPIC: The Model Menu - SUBTOPIC: About The Analysis

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.

18.3: TOPIC: The Model Menu - SUBTOPIC: Visualize Model

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,

AllHelp.txt

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).

18.4: TOPIC: The Model Menu - SUBTOPIC: Report Model ...

REPORT MODEL produces a listing of numeric information about your model.

The details of the report depend on which model you have chosen.

18.5: TOPIC: The Model Menu - SUBTOPIC: Interpret Model

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.

18.6: TOPIC: The Model Menu - SUBTOPIC: Delete Model

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.

18.7: TOPIC: The Model Menu - SUBTOPIC: Create Data ...

CREATE DATA creates new data from the current model. For some models, many different kinds of data can be

AllHelp.txt

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.

19.1: TOPIC: The Options Menu - SUBTOPIC: The Options Menu

This topic presents you with the help information for the items of the Options Menu.
Choose an item to see

its help.

19.2: TOPIC: The Options Menu - SUBTOPIC: Preferences ...

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:

SHOW EXIT DIALOG - Causes a dialog to show which asks if you really want to exit?

SPLASH SCREEN OPTIONS - The opening "ViSta" splash screen can be hidden, static or
dynamic.

LOAD DATA SHOWS DATASHEET - Click this if you want to automatically see a dataheet
of your data. after you

load in data.

SHOW SPINPLOTS SPINNING - Click this if you want to see spin plots spinning when they
are first shown.

AUTOSHOW MODEL REPORTS - This lets you automatically see the model's report after a
model is constructed.

AUTOSHOW MODEL SPREADPLOTS - This lets you automatically see the model's spreadplot
after a model is

constructed.

AllHelp.txt

CLICK-CLOSE HELP WINDOWS - Close a help window by clicking anywhere on the window.

LONG MENUS adds data and model object names to the data and model menus. This can be useful when you hide

the workmap.

LOAD FILES WITH RUNNING VISTA - When you double click a data or applet file, load it into the most

recently run instance of ViSta. Otherwise, a new instance will always be used.

19.3: TOPIC: The Options Menu - SUBTOPIC: Run Excel ...

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.

19.4: TOPIC: The Options Menu - SUBTOPIC: Show Clock

SHOW CLOCK

Allows you to turn the morphing vista logo clock on and off. When the clock appears ViSta disappears, and

can be made to reappear when the clock window is clicked.

19.5: TOPIC: The Options Menu - SUBTOPIC: Clock Options

CLOCK OPTIONS:

Sets options for the morphing ViSta logo clock.

19.6: TOPIC: The Options Menu - SUBTOPIC: Twiddle

TWIDDLE

Watch ViSta twiddle its thumbs!

19.7: TOPIC: The Options Menu - SUBTOPIC: Record Listener

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.

19.8: TOPIC: The Options Menu - SUBTOPIC: Refresh System

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 environment 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 occurred, 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 around 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.

19.9: TOPIC: The Options Menu - SUBTOPIC: Developers Mode

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.

20.1: TOPIC: The Help Menu - SUBTOPIC: The Help Menu

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.

DOCUMENTATION show a dialog box of files containing documents that have been written about ViSta.

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.

COPYRIGHTS shows a window of copyrights related to ViSta.

21.1: TOPIC: The DeskTop Menu - SUBTOPIC: The DeskTop Menu

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.

22.1: TOPIC: The Window Menu - SUBTOPIC: The Window Menu

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.