

VISRAL L.P.

Python Solutions and Operating Environment

Visral® User Manual

Volume 1

3 Sept 2016

www.visral.com

P.O. Box 646

Rockwall, Texas 75087

Preliminary

© 2005 - 2016 Visral L.P. All rights reserved

VISRAL is a registered trademark of Visral L.P.

Document Revisions

Date	Version Number	Document Changes
05/02/2014	0.1	Initial Draft
07/05/2015	0.2	Beta release
08/30/2015	0.21	Minor update
09/06/2015	0.22	Inclusion of Python 3 features and controls
10/13/2015	0.23	Minor update
1/21/2016	0.31	Updated to cover OE and Sire versions 3.0.1.2
1/24/2016	0.32	Minor update
3/2/2016	0.33	Updated to cover OE/SIRE combination 3.0.2.5
4/18/2016	0.34	Updated to cover multiport diagrams 3.0.2.14
6/3/2016	0.35	Update to cover new operational methodology of SIO builds.
9/3/2016	0.36	Sire features included in OE.

Notes:

1. This guide may describe some features that have not been enabled.
2. The representative images may in some cases appear slightly different from those of application itself.
3. References to right click or left click means respectively, pushing right mouse button or pushing left mouse button.

Icons and images by [Aha-Soft](#), [Yusuke Kamiyamane](#), [LED24.DE](#), [PC.DE](#), and [Fatcow](#)
Other images are the property of Visral L.P. or in the Public Domain.

Contents

INTRODUCTION 5

WORKSPACE 7

Other.....9

VISRAL OE BASICS 10

 PANELS AND OPERATORS.....10

 INTERACTIVE DOCUMENTS 11

 DATASET SPREADSHEETS.....12

 PYTHON PAD EDITORS.....13

 ACTIVE DIAGRAMS 14

 IMPORTING FILES.....15

Recent Files15

 SELECTING WORKING VENUES 16

VENUES, PANELS, AND OPERATORS..... 17

 VENUES.....17

 PANELS18

 OPERATORS18

Munge / Calculate Panel.....19

Tracking Toolbar20

Input Elements23

Argument source.....26

SPREADSHEETS/DATASETS 27

The Difference.....27

Data Sets.....27

Moving dataset content view28

Editing Dataset Cell Content28

Capturing dataset data.....31

Selected data as a source for Operators31

Manipulating the Spreadsheet/Dataset Columns.....32

Munge / Calculate Spreadsheets and Datasets33

 SERIES.....34

 SPREADSHEET CELLS, FORMULAS, AND EXECUTION 35

DOCUMENTS..... 37

Report, Guide, and Trace Editing38

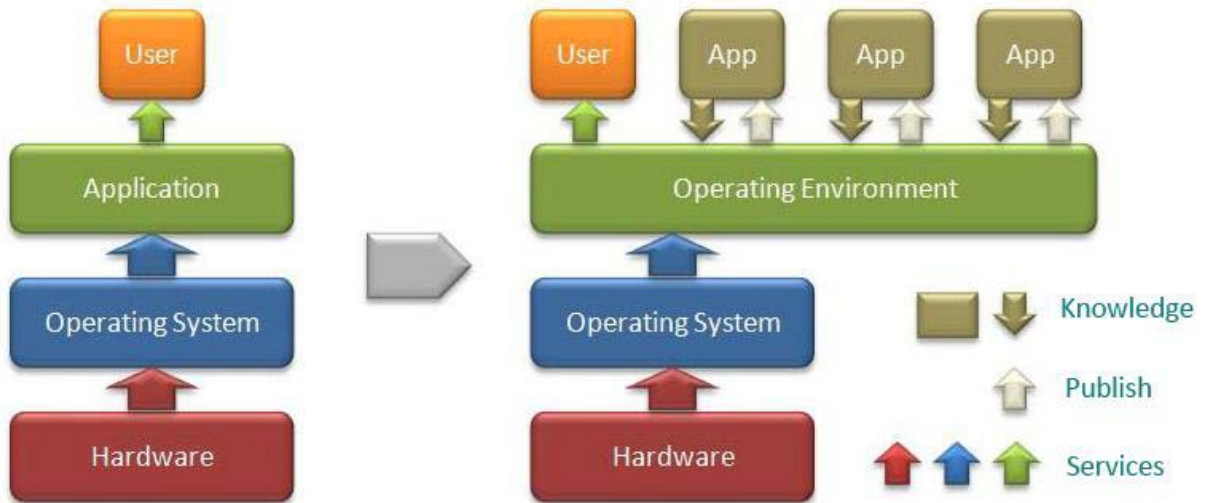
Editing embedded items41

<i>Guide embedded response</i>	42
<i>Passage Management</i>	44
<i>Encryption</i>	45
<i>Python Control of Visral Assets</i>	46
CREATING SIO FILES	50
PAD EDITORS	53
PYTHON CODE	58
SQL CODE	60
FORMULA EDITOR	61
<i>Execution History</i>	61
<i>Python Import Aliases</i>	62
<i>Python Components</i>	63
OPTIONAL: USING OTHER VERSIONS OF PYTHON.....	64
VISRAL DIAGRAMS – VENUES/PANELS/OPERATORS	66
CREATING APPLICATIONS	66
<i>Now for Inputs</i>	70
VENUE CONSTRUCTION DIAGRAMS.....	72
<i>Panel Elements</i>	74
<i>Placeholders</i>	79
VISRAL DIAGRAMS - UNIVERSAL	81
<i>Description</i>	81
<i>Loading, saving, and selecting files</i>	82
<i>Moving about the Diagram</i>	83
<i>Selecting an Element Group</i>	83
<i>Inserting Elements</i>	84
<i>Junctions</i>	85
<i>Editing Element Formulas</i>	86
<i>Wiring Two Port Elements</i>	89
AUTOMATION (PREVIEW)	92
INTRODUCTION TO AUTOMATONS	92
<i>Automaton Instantiation</i>	93
APPENDIX	96
<i>Product Specifications</i>	96

Introduction

Notes:

- ✓ This manual covers both the Visral OE features.



Hardware and Software Requirements

Windows: 7, 8, or 10
2 GB RAM (the more the better)
1024 x 768 or higher
Up to 1 GB free disk
2 button mouse w/ wheel

Extensive testing was done on Windows 7, but final checkout is performed on 8 and 10. (However, OE has run under: Vista, MAC Bootcamp, and VirtualBox.)

- ✓ The embedded Python includes all of the modules required to run the various solutions contained in the provided Panels.

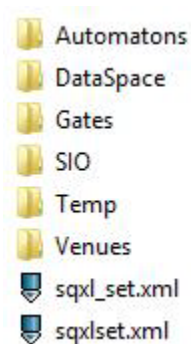
The Visral Package

By default, Visral installs its application and bridges in “Program Files (x86)” under the directory name “Visral 3.0”. Included are several subdirectories where the more permanent content is kept.

Visral creates the directory “Visral 3.0” in the user’s Document directory containing several subdirectories for updatable semi-permanent data and working files. Program and manual references to Integrants are referring to files at these locations.

The XML files “sqlset.xml/sql_set.xml” contain property settings, syntax information, and icon image descriptions.

As with Python, This version of Visral is a memory resident application that does not offload working data structures to disk.



Workspace

Covers:

- ✓ Layout
- ✓ Adjusting Window Panes
- ✓ Toolbar's Multiple Faces
- ✓ Recent Files
- ✓ System Managers

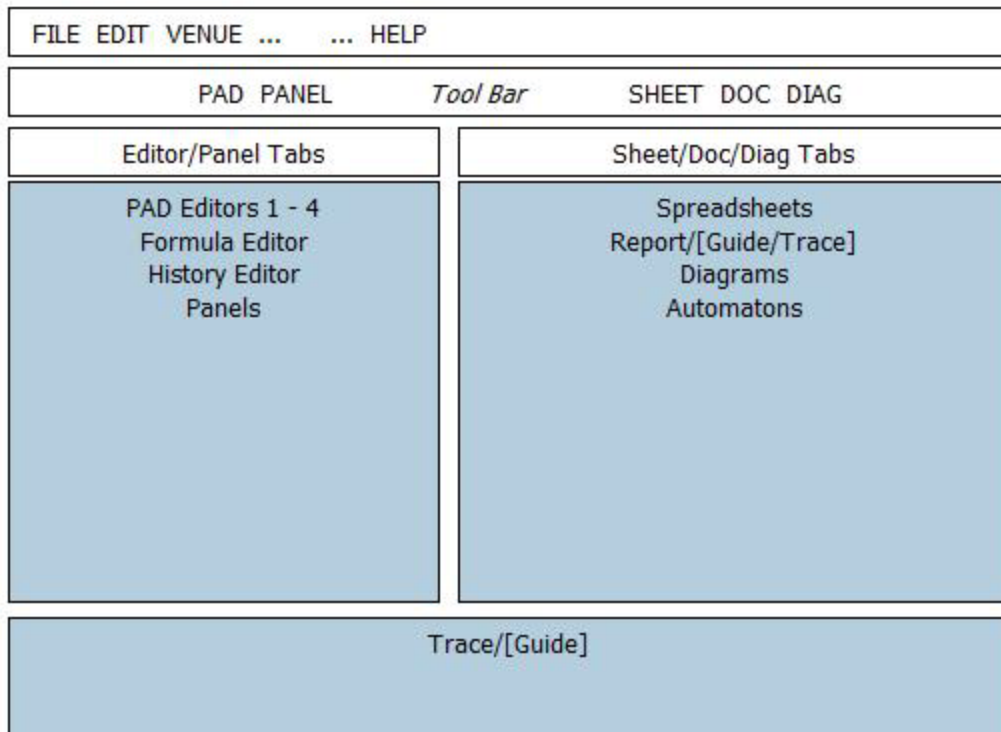
Layout

Visral has three main window panes. The one with focus will have a white background, where as the background of the other two will be light grey. To compensate for the dearth of vertical space that followed HD video to PC screens, the layout is designed around a single morphing toolbar and the elimination of ornamental and little used widow frames.

The Trace window by default is displayed in the bottom window but can also be displayed in the right one, allowing the lower pane to be collapsed.

The legend below highlights the three panes and lists the various resources each can displayed and be interacted with.

The main menu names in capitals (FILE, EDIT, VENUE, and HELP) are a standard part of the Visral system. The titles between VENUE and HELP are automatically created from the detection of installed and selected Venues.



There is a single main toolbar that changes with the particular window that has focus.



Grayed icons indicate the button is disabled.











The main toolbar is divided into 5 sections of icon buttons.

	Description
1	The first ten icons in the toolbar apply individually to each window that has focus. The ones that are not applicable will be grayed.
2	The second two icons allow switching the left window pane between Pad editors and Panels.
3	This group has two buttons. The first brings up the Control and Communications morphing toolbar and the second cycles section 4 buttons through vista selection, window size adjustments, and back to the buttons associated with the window that has focus.
4	The next four icons are different depending on which window has focus and the nature of the window's content.
5	These three icons allow switching the right window pane between spreadsheets and datasets, documents, and diagrams .


Section 4 Examples (Icons change depending on the process/window that has focus.)

Windows	Instances	Morphing toolbar	
Pads	4		
Formula	1		
Panels	Multiple		
Sheets	Multiple		
Report	1		
Guide	1		
Diagrams	Multiple		
Resize	4		

Section 1 (Icons may be grayed depending on the process/window that has focus.)

Button	Description
	Input File – This button will bring up the load file menu. Depending on which window has focus, it will be Python, CSV, RTF, XML or other file types.
	Save File – This button will bring up the save file menu. Depending on which window has focus, it will be Python, CSV, RTF, XML or other file type. (Saving a file does not automatically close it.)
	Close Window - This button will close the window in focus without saving its contents.
	Print Window Contents - Pressing the print button will bring up the printer menu for printing contents from the window that has focus.
	Cut - This button will cause the selected content of the window with focus be copied to the clipboard and removed.
	Copy - This button will cause the selected content of the window with focus be copied to the clipboard without being removed.
	Paste - This button will cause the clipboard contents to be inserted at the location of the cursor, or overlaid if contents had been selected.
	Find - Brings up the search menu for windows that have that facility.
	Undo – undoes the last operation of window that has focus.
	Redo – redoes the last undone operation of window that has focus.

Adjusting Window Panes - Vistas

Window pane sizes can be adjusted through the centrally located toolbar **Vista** button . Initial clicking of the button will change the Morphing toolbar to display the resizing buttons. Clicking these buttons will cause a percentage change in the panes in accordance with the button images for that particular vista.



Subsequent clicking will step through the four possible Vistas.

The sizes of the panes is remembered for each of the 4 Vista.

Other

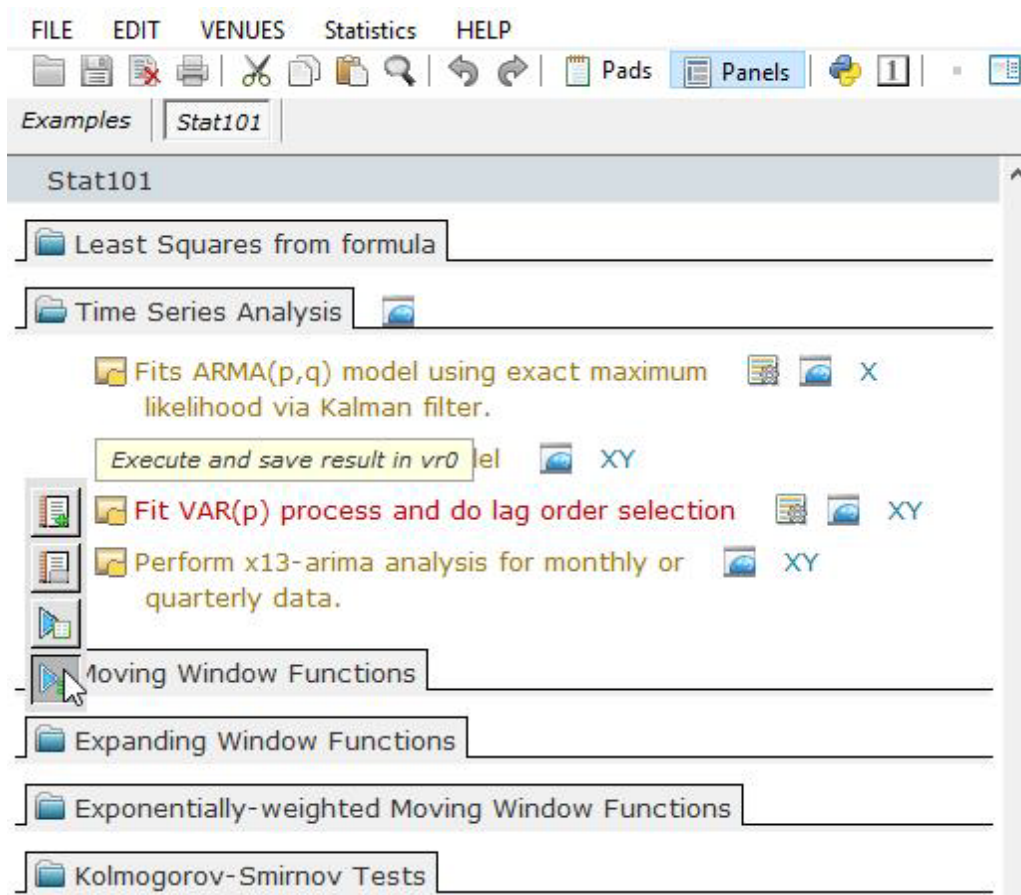
Closing popup menus: Clicking on the background of the relevant windowpane will cause a popup menu to close. Other menu types will have a close button.

Closing a tab: A Panel, Dataset sheet, or Diagram tab and its contents that have focus can be closed by either clicking the **Close Window** icon in the toolbar or by right clicking on the window itself or selecting the Close entry. Both these action will cause a conformation popup to appear requiring an OK (or cancel).

Visral OE Basics

Visral OE can simply be used by itself with the provided Venues (solution sets) or with those that others may produce. It may be used for its Python features or because of its spreadsheet capacities. It has the ability to save and retrieve work in progress, including those of Python, Datasets, and Documents.

Panels and Operators




Panels are enhanced replacements for Microsoft's menus and generally contain a number of operators. The left window pane will show the Panel view by selecting the PANEL button in the toolbar. Clicking on a sub-menu under any of the main menu entries between VENUE and HELP will bring up the relevant Panel.



Operators are the specific entries within a Panel that causes an execution or activity of some type. Their text is gold color and they have a gold box symbol for their icon. The icons that include the image of a small folder can be expanded to show relevant settable parameters by clicking on them.

Interactive Documents



Welcome to Visral OE/SIRE

And its Operating Environment. This interactive tutorial uses synthesized voice to help explain some of its features. Window panes can be adjusted by clicking the button  which will bring up the resize buttons.

The Panel to the left has some examples and references , which can be executed by clicking on the Operator text or the  symbol next to it.

Press the following buttons for demonstrations of some of OE's features.

Actuate

Demonstrates executing the Lag-1 Autocorrelation Operator. It may be necessary to page down the panel to locate it.

An encapsulated Textual Passage with an OK button is hidden here (or not).

Actuate

This button will bring up a Panel of sample Operators.

Operator

This button will bring up a Panel with a single Operator which when clicked will perform the same function that the following button will perform directly when clicked.

The textual and graphical results of operations are directed to the Trace (default) and Report editors where they can be supplemented with the author's own text, spreadsheet printouts, and the pasting of photos and other images.

Beyond that Visral allows the embedding of datasets, Venues, Operators, and sections of code into the Report. The Guide, Report, and Trace editors have search ability and, paragraph and character formatting, as well as accept the pasting of images.

Dataset Spreadsheets

The screenshot shows the Visral interface with a toolbar at the top containing icons for file operations and a 'Diag' button. Below the toolbar, the 'SERIES' dropdown is set to 'sp500' and the 'mpg' dataset is selected. The spreadsheet window displays a table with columns labeled 'Clear', 'A', 'G/x', 'H/x', 'I/z', 'J', 'K', and 'L/y'. The first row (row 0) is highlighted in red. The second row (row 1) contains the text 'Index (Model Type Index)'. The third row (row 2) is the header for the data table, with columns: 'Index', 'Eng [', '# Cy', 'Transmission', 'City I', and 'Hwy'. The data rows (rows 22-33) are highlighted in green. A mouse cursor is pointing at the 'Index' column header.

Clear	A	G/x	H/x	I/z	J	K	L/y
0							
1		Index (Model Type Index)					
0		Index	Eng [# Cy	Transmission	City I	Hwy
22	21	438	3	6	Auto(AM-S7)	17	24
23	22	60	8	16	Auto(AM-S7)	8	15
24	23	59	6.2	8	Auto(S8)	16	29
25	24	67	6.2	8	Auto(S8)	13	21
26	25	60	6.2	8	Manual(M7)	17	29
27	26	68	6.2	8	Manual(M7)	15	22
28	27	185	8.4	10	Manual(M6)	12	21
29	28	143	4.5	8	Auto(AM7)	13	17
30	29	142	4.5	8	Auto(AM7)	13	17
31	30	146	4.5	8	Auto(AM7)	13	17
32	31	147	4.5	8	Auto(AM7)	13	17
33	32	149	4.5	8	Auto(AM7)	13	17

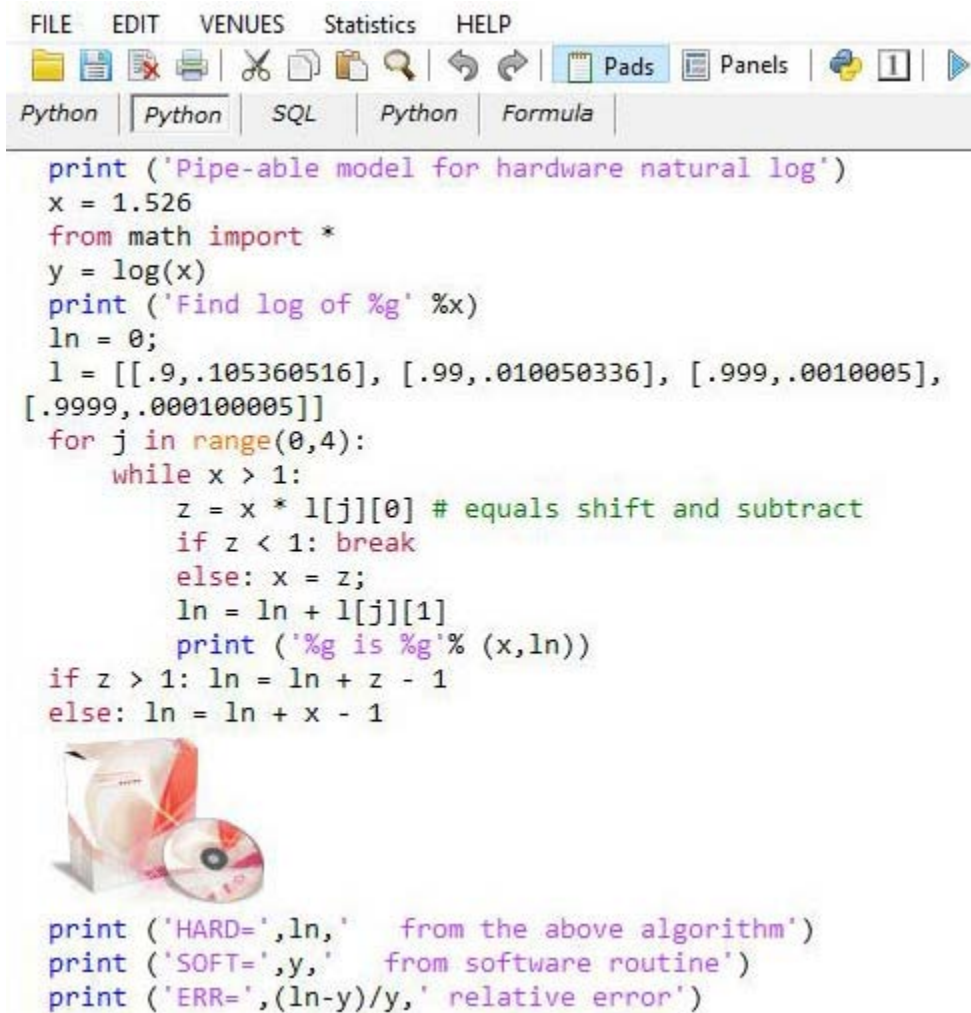
solutions.

The Sheets window is divided into two sections. Dataset content is loaded into passive cells located in the lower portion with the black line number. The spreadsheets with the active cells that can contain Python instructions are located in the top section with the red line numbers.

Memory permitting, datasets as large as 4095 columns by 1 million rows can be hosted.


The spreadsheet can assemble results from any combination of rows and columns for processing or creation of new DataFrames. Where conventional spreadsheets have a single calculate button, multiple Visral Operators offer the opportunity for complex Python

Python PAD Editors



```

print ('Pipe-able model for hardware natural log')
x = 1.526
from math import *
y = log(x)
print ('Find log of %g' %x)
ln = 0;
l = [[.9,.105360516], [.99,.010050336], [.999,.0010005],
[.9999,.000100005]]
for j in range(0,4):
    while x > 1:
        z = x * l[j][0] # equals shift and subtract
        if z < 1: break
        else: x = z;
        ln = ln + l[j][1]
        print ('%g is %g'% (x,ln))
if z > 1: ln = ln + z - 1
else: ln = ln + x - 1



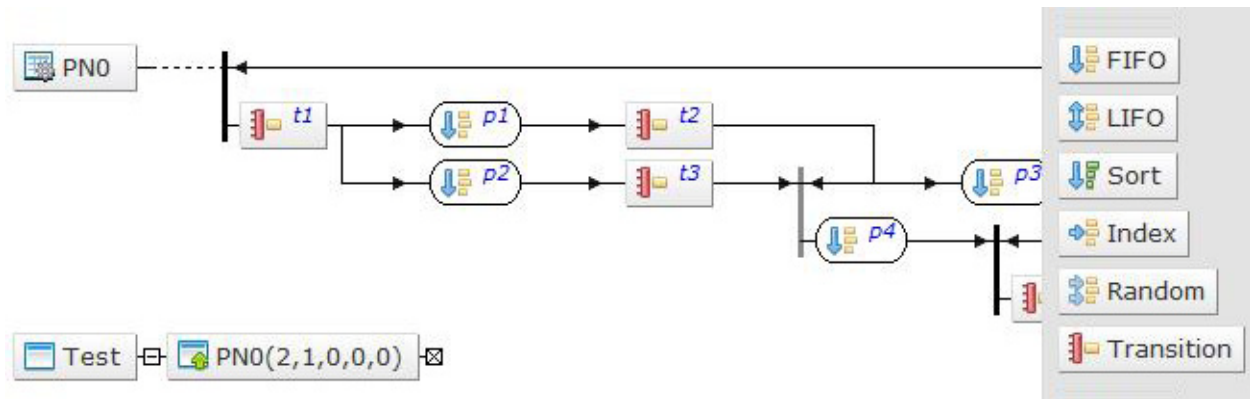
print ('HARD=',ln,' from the above algorithm')
print ('SOFT=',y,' from software routine')
print ('ERR=',(ln-y)/y,' relative error')

```

The PAD editors can be configured to process or communicate with different language systems or networks. This includes setting syntax coloring, indentation, auto completion, and execution. Contents can be saved as text or in RTF format. In preparation for forthcoming features, the PAD editors also accept the pasting of images without interference.

Python runs natively on five of the Visral editors; the four Pad editors and the Formula editor. The results from the four PADs are directed to either the Trace or Report document editors. The behavior is as would be expected executing the last line of code following a return and sending the result to the Trace editor when its entry has been completed. Visral allows copying, pasting, and executing selected ranges of code.

Active Diagrams



Not to be confused with the plots and charts outputted from Python, Visral Diagrams are interactive graphics, for conveying information to applications. The nature of their visual presentation depends upon the particular type of diagram being displayed and interacted with. The same is true as to whether their elements are 2-port or multiport, and their processing, which fall into two categories, formulation and compilation.

Formulation is where the diagram is interpreted and converted to a DataFrame or array for computation by Python or another language. In compilation, diagrams create active models or automaton, which are subsequently engaged by other operations or applications. This brings significantly more power to models as it allows the integration of complex and non-linear elements.

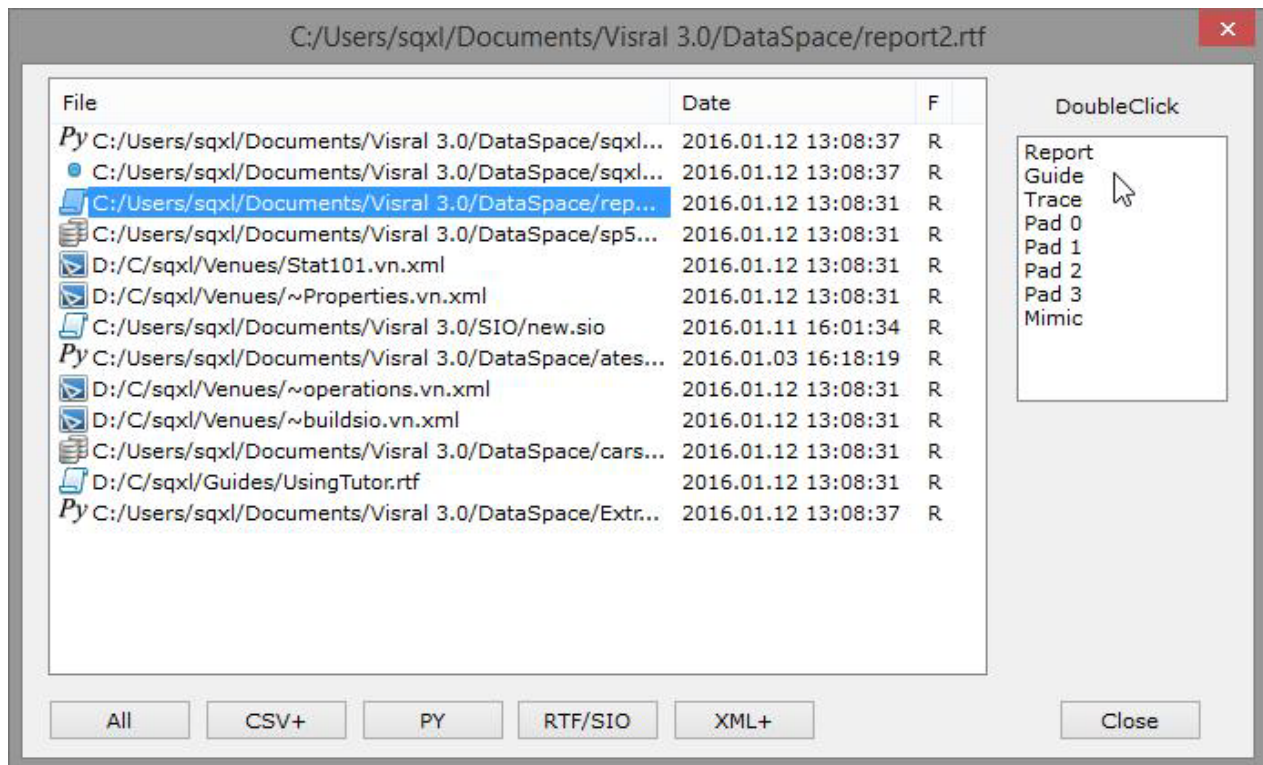
Importing files

Depending on which window has focus, the main menu FILE entry will provide various options over and above that of the main toolbar File Input button. Visral also supports drag and drop of files, but they can only be dropped onto the left window pane (PADs, Formula, or Panel). (**Do not drop them in the Report, Guide, or Trace as it will embed them in the document.**) SIO files can configure and load many features, XML files will go to the Diagram, Python to the Pad that has focus, CSV files to the dataset sheet, and RTF files to the Report. Venues, Datasets, and other feature are added to Visral through SIO files.

Double clicking on a SIO file in a file explorer will cause Visral to load and operate it. (If Visral was not running, it will start it up.)

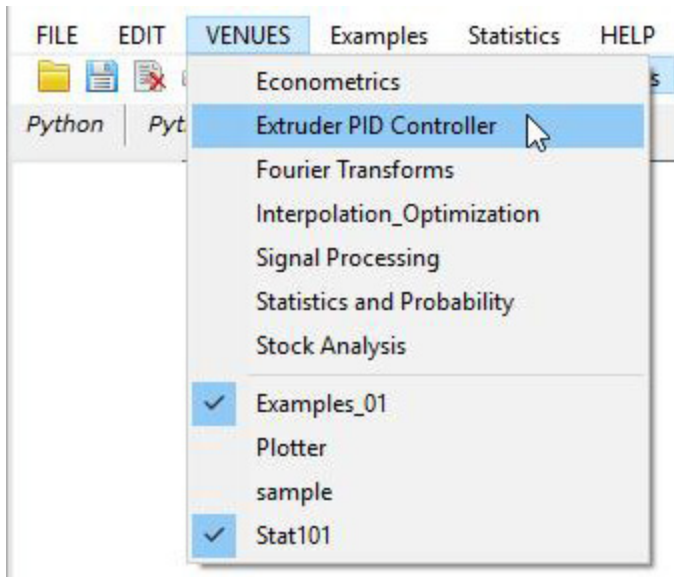
Recent Files

The recent files menu can be reached from the main menu FILE entry and tracks file activity. When a particular file is selected the list box to the right will show the possible destinations for it. Double clicking an entry in the box will send the file to that location.



- ✓ Double clicking on the File or Date headings will reorder the contents of the listing.
- ✓ The five buttons CSV+, PY, RTF/SIO, and XML+ will display just the files with those extensions. The + indicates other extensions within the same category. Obviously, the All button will display all files.

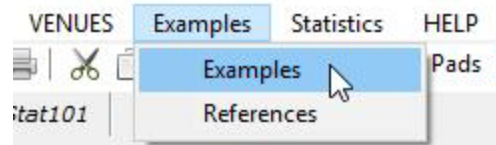
Selecting Working Venues



A Venue is a solution set composed of one or more Panels, each with one or more Operators. They are intended to bring together a group of solutions covering a specific topic or problem area.

All installed non-system Venues are located under VENUE of the main menu. Those with checks indicate loaded Venues. To load or unload a Venue simply click the appropriate entry to toggle its condition. Venues are created from construction diagrams as described in the Visral Diagrams.

Loading a Venue will cause the insertion of entries into the main menu between the VENUE and HELP titles. Clicking any of those entries will expose a dropdown listing the available Panels. The Venue names are derived from their XML file names. The names of the inserted entries in the main menu come from the MenuBar elements within the XML file. The Panel names come from the Panel elements.



Venues, Panels, and Operators

Covers:

- ✓ Venues
- ✓ Operators
- ✓ Tracking Toolbar
- ✓ Input Elements
- ✓ Argument Source

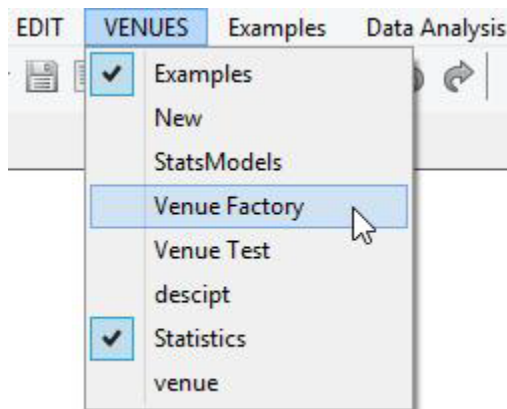
Venues

A Venue is a solution set composed of one or more Panels, each with one or more Operators. They are intended to bring together a group of solutions covering a specific topic or problem area. Venues are automatically loaded from XML files located in one of two directories:

- C:\Users\...\Documents\Visral 3.x\Venues
- C:\Program Files (x86)\Visral 3.x\Venues

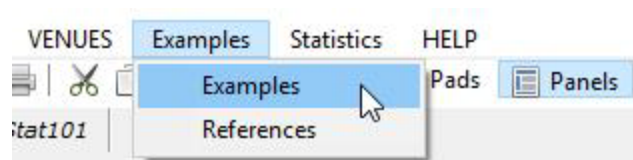
New Venues are installed from RTF file distributions whose creation is described in the Document section of this manual. This permits one or more of them to be accompanied by datasets, examples, and explanations; all within a single file.

All installed non-system Venues are located under VENUE of the main menu. Those with checks indicate loaded Venues. To load or unload a Venue simply click the appropriate entry to toggle its condition. (Venues are created from construction diagrams as described in the Visral Diagrams section of this manual.)



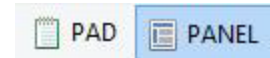
Loading a Venue will cause the insertion of entries into the main menu between the VENUE and HELP titles. Clicking any of those entries will expose a dropdown listing the available Panels. (The Venue names come from the name of their filenames. The names of the inserted entries in the main menu come from the elements within the file itself and not from the name of the file.)

There can be multiple folders within each displayed Panel.



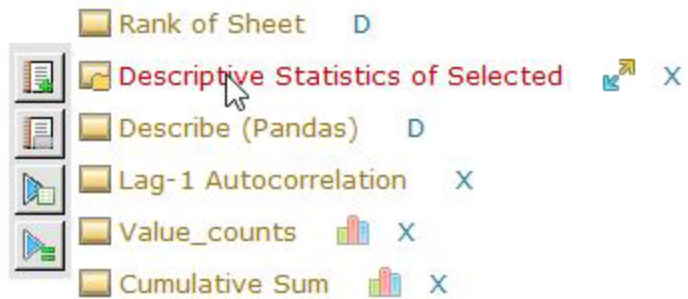
Panels

Panels are enhanced replacements for Microsoft's menus and generally contain a number of operators. The left window pane will show the Panel view by selecting the **PANEL** button in the toolbar. Clicking on a sub-menu under any of the main menu entries between VENUE and HELP will bring up the relevant Panel.



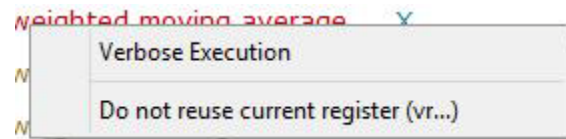
Operators

Operators are the specific entries within a Panel that causes an execution or activity of some type. Their text is gold color and they have a gold box symbol for their icon.



The image to the right is of a portion of the Stat101 Panel. The Operator that the cursor is over has turned red and the top button of the tracking toolbar aligns with it. A left click on the text or any of the menu buttons will cause the processing to take place. The icons that include the image of a small folder can be expanded to show any settable parameters by clicking on them.

Right click on an Operator will reveal a menu with two entries. Selecting **Verbose Execution** will execute and trace the actual code as it is fed to the bridge. This can be helpful in developing new solutions or understanding a specific Operator's requirements and actions.



Auxiliary Operators and Conditions

The following icons may be found to the right of some Operators.

	This provides access to information by sending the default browser to a previously assigned web page. (Does not cause the Operator to execute.)
	Clicking (toggle) this will instruct the Operator to produce a plotted output when executed.
	Clicking this will cause an example to be executed illustrating the functions of the Operator. The example's code is stored under the #PAYLOAD entry of the Operator's element.
	The function of the toggle icon varies with individual Operators. (Does not cause the Operator to execute.)
	This icon is only revealed when the cursor is over the line the element is on and reveals a popup explanation of the element when passed over.

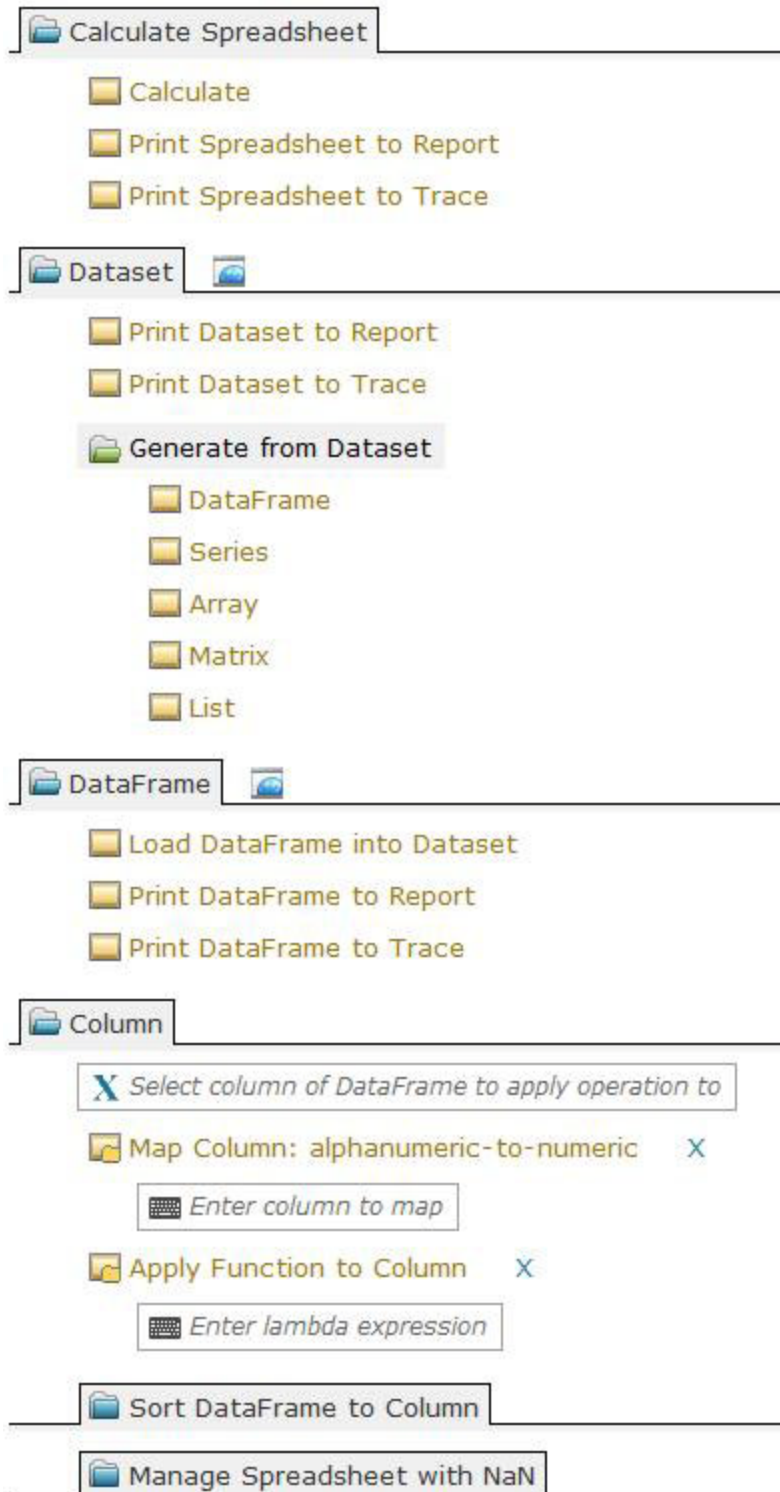
Munge / Calculate Panel

Beyond files, dataset sheets can be loaded from Python DataFrames, matrixes, and arrays, but they are copies of the data. If they are edited and it is desired to have the original source reflect the changes, the DataFrame sources will need to be updated.

New DataFrames, matrixes, arrays, and series can be generated from dataset sheets, either partially by selecting rows and columns or fully by clearing all selections.

The Munge part of the Panel refers to operators for generating DataFrames, Series, arrays, matrixes, and Lists. The Munge Panel can be found in the Morphing menu and in the main menu's SYSTEM dropdown.

The Calculate Operator simply executes any code that might reside within the cells. The real power of Visral is the user's ability to create an array of Operators that implement sophisticated solutions.



Tracking Toolbar

There is a tracking toolbar that appears to the left as the cursor moves over each Operator within a Panel.

At the same time the title turns from gold to red, where upon clicking it will cause the process to be executed with the results going to the Trace.

Clicking the top button in the tracking toolbar will cause the results to go to the Report.

The cursor must be moved horizontally to the menu before choosing a button further down the column.





See upcoming tracking toolbar table for more details. (The toolbar buttons may vary in type and count, depending on Operator and the specific Panel.)

The folder tabs expand to reveal Operators and collapse to hide them with left clicks.

The screenshot displays a vertical toolbar on the left with four icons: a document, a document with a cursor, a chart, and a chart with a cursor. To the right, a list of operators is shown, each with a folder icon, a name, and a status icon (X or XY). A yellow box highlights the text "Send code as a Button to Report". Below the operator list, a series of folder tabs are shown, each with a folder icon and a label: "Least Squares from formula", "Time Series Analysis", "Moving Window Functions", "Expanding Window Functions", "Exponentially-weighted Moving Window Functions", and "Kolmogorov-Smirnov Tests".

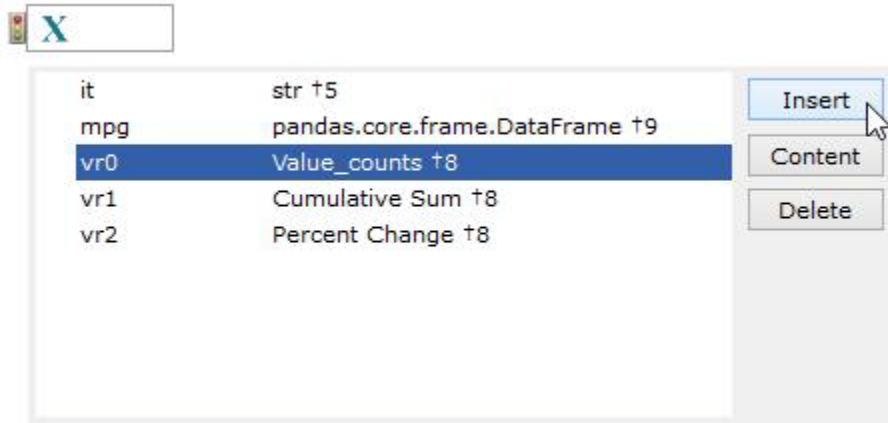
Operator Name	Status
Cumulative Sum	X
Percent Change	X
distplot	X
Crosstab	XY
Correlation	XY
Correlation Coefficient	XY
<i>Send code as a Button to Report</i>	
Autocovariance for 1D	X
Autocorrelation function for 1d arrays.	X
Partial autocorrelation estimated	X
Partial autocorrelation estimated with non-recursive yule_walker	X
Calculate partial autocorrelations	X
Crosscovariance for 1D	XY
Cross-correlation function for 1d	XY
Returns the periodogram for the natural frequency of X	X
Augmented Dickey-Fuller unit root test	X
Levinson-Durbin recursion for autoregressive processes	X

The following table details the actions of the menu buttons that track alongside the Operators. The number and selection of entries may change depending on application.

Button	Description																																																												
	<p>Left clicking on the Report icon with the plus symbol will execute the currently selected operation and send results to the Report editor.</p> <p>Unlike the Trace, which receives all Python notifications, Visral blocks those sent to the Report that are not actually part of a result.</p>																																																												
	<p>Left clicking on the Report icon with the button symbol will embed the currently selected operation in the Report editor as a button. When the button is clicked in the Report or Guide an Operate entry with the code will be created in a temporary Panel. (See document section for details.)</p>																																																												
	<p>This action is only works for Series results. It displays a scrollable overlay in the Panel window and records the results in the SERIES sheet. They are entered into the sheet in a push down fashion and include both column and index. (Actually, a slide right fashion.)</p> <p>The difference here is that the complete result is available for viewing, where the normal printout from Python will ususally just display the beginning and end of larger listings.</p> <div data-bbox="857 785 1365 1230" style="border: 1px solid gray; padding: 5px;"> <table border="1"> <thead> <tr> <th colspan="2"></th> <th>SERIES</th> <th>cars32</th> <th>ti</th> </tr> <tr> <th colspan="2"></th> <th>Clear</th> <th>A</th> <th>B</th> </tr> </thead> <tbody> <tr> <td>190</td> <td>4</td> <td>Layer D</td> <td><input type="checkbox"/></td> <td>VRvc</td> </tr> <tr> <td>200</td> <td>3</td> <td>1</td> <td>200</td> <td>3</td> </tr> <tr> <td>197</td> <td>3</td> <td>2</td> <td>197</td> <td>3</td> </tr> <tr> <td>191</td> <td>2</td> <td>3</td> <td>191</td> <td>2</td> </tr> <tr> <td>188</td> <td>2</td> <td>4</td> <td>188</td> <td>2</td> </tr> <tr> <td>194</td> <td>2</td> <td>5</td> <td>194</td> <td>2</td> </tr> <tr> <td>196</td> <td>2</td> <td>6</td> <td>196</td> <td>2</td> </tr> <tr> <td>212</td> <td>2</td> <td>7</td> <td>212</td> <td>2</td> </tr> <tr> <td>177</td> <td>2</td> <td>8</td> <td>177</td> <td>2</td> </tr> <tr> <td>193</td> <td>1</td> <td>9</td> <td>193</td> <td>1</td> </tr> </tbody> </table> </div>			SERIES	cars32	ti			Clear	A	B	190	4	Layer D	<input type="checkbox"/>	VRvc	200	3	1	200	3	197	3	2	197	3	191	2	3	191	2	188	2	4	188	2	194	2	5	194	2	196	2	6	196	2	212	2	7	212	2	177	2	8	177	2	193	1	9	193	1
		SERIES	cars32	ti																																																									
		Clear	A	B																																																									
190	4	Layer D	<input type="checkbox"/>	VRvc																																																									
200	3	1	200	3																																																									
197	3	2	197	3																																																									
191	2	3	191	2																																																									
188	2	4	188	2																																																									
194	2	5	194	2																																																									
196	2	6	196	2																																																									
212	2	7	212	2																																																									
177	2	8	177	2																																																									
193	1	9	193	1																																																									
	<p>This button will cause the result of the Operator to be stored in a register for later use and not sent to either the Trace or Report. The registers names start with lower case 'vr' followed by a sequentially increasing number. (Registers are Python variables managed by Visral.)</p> <p>All Operators, where possible, will store their results in a register along with sending them to the Trace or Report. The difference is the assigned register number will not be automatically incremented. The Do not reuse current register entry of the right click menu can be used to save the results from any Operator by allowing it to increment for the next use.</p> <div data-bbox="818 1528 1377 1661" style="border: 1px solid gray; padding: 5px;"> <p>weighted moving average Y</p> <p>Verbose Execution</p> <p>Do not reuse current register (vr...)</p> </div>																																																												

The following menu will appear with a right click on any Input Element. Single or multiple items may be selected for inserting in to the element. For certain variable types, the value can be chosen to be inserted as opposed to the name, by selecting the Content button.

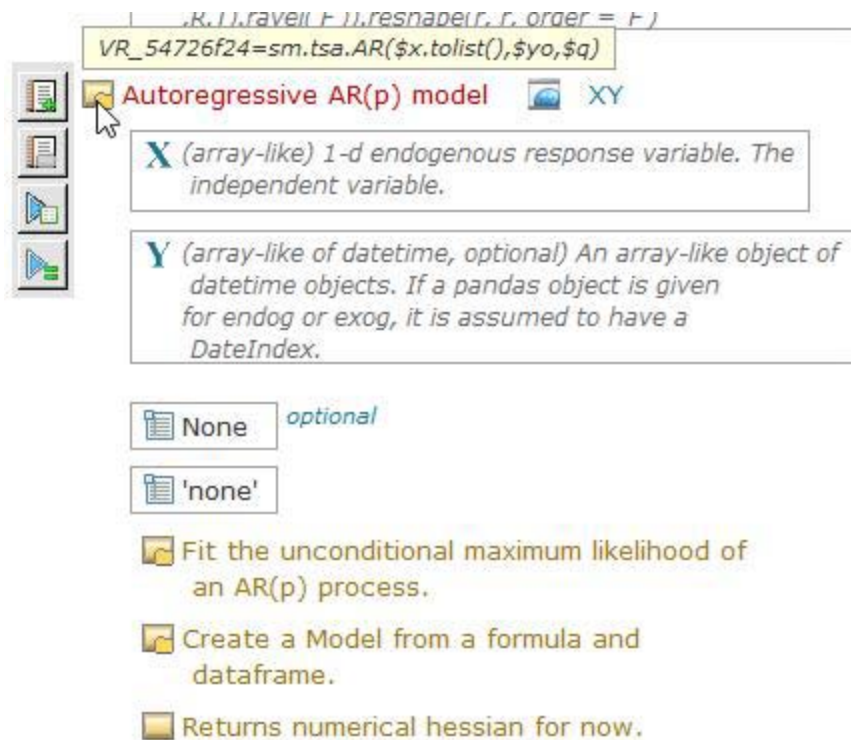
The vr0, vr1, and vr2 were generated from a Panel and as such include the title of the Operator. Otherwise, the type will be displayed. The dagger followed by a number is a type designation used internally by Visral.



The Delete button allows removing the entry and content from Python. Of course, if there are other items pointing to the same content, it will not be removed.



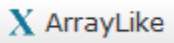
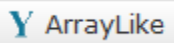
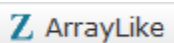
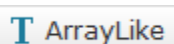
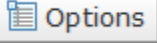

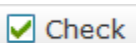
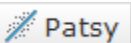
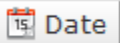
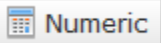
If the icon associated with the operator is a folder, it means there are items that can be accessed. If the folder is in the closed attitude, clicking will display the relevant arguments and any dependent methods. In the example below the X, Y and two selection inputs provide a means to enter arguments to the parent Operator.

The child Operators in this example represent the parent's method programs. The parent Operator must be executed first to provide the results the child programs need.



Input Elements

The following is a partial list of elements used to provide arguments to an Operator. In practice they may contain default values.

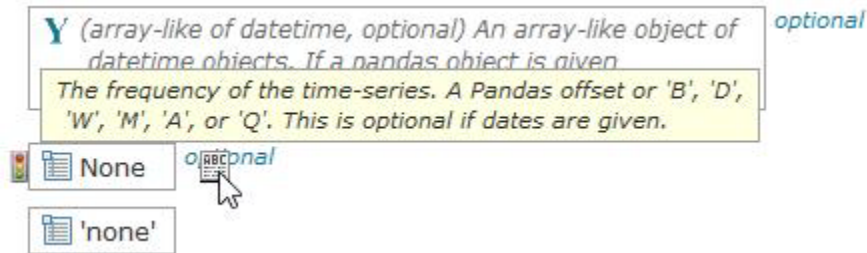
Element	Description
 Expression	This element accepts typed text by first left clicking the element.
 Tuple	This element accepts typed text by first left clicking the element.
   	These elements accept typed text by first left clicking the element. A right click will bring up a menu of all available DataFrames. Double clicking on a DataFrame entry will expose all of its columns. A DataFrame or any combination of columns can be selected by clicking the insert menu's button. Menu example below.
 Options	This element accepts typed text by first left clicking the element. Left clicking will bring up a menu of options to choose from. Double clicking a menu entry item will insert it into the element.
 	Clicking a check box element toggles it back and forth between checked and unchecked. If check boxes are children of a Radio element, it will allow only one check box at a time to be set.
 Patsy	Similar to the Data In element except indicates the input is expected to be a Patsy formula expression. It accepts column names to form expressions, such as: "Z ~ X + Y". See http://patsy.readthedocs.org/en/latest/formulas.html for more details.
 Date	Understands various date formats and converts them to a standard for processing.
 Numeric	Expects a numerical value to be entered.

See Venue construction diagrams for more details on available elements.

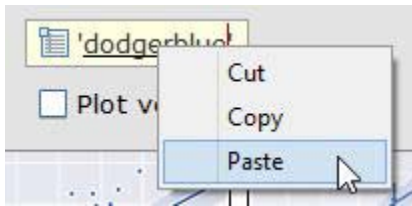
Options

Moving the cursor over the icon of an Operator will reveal a popup with the program contents as shown in the example to the right. Passing the cursor over the news icon to the right will reveal a popup explanation of the element. The news icon is only revealed when the cursor is over the line the element is on.

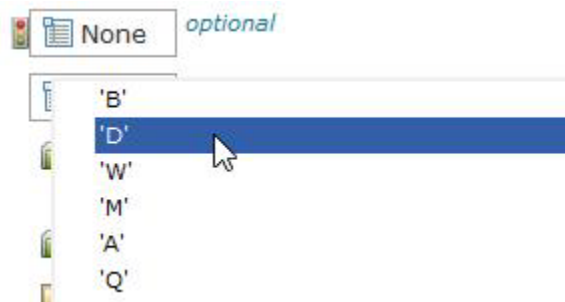
The mini traffic light to the left indicates which element has the focus. Operator's indication of focus is the red text.



Left clicking on an options input will cause it to go into edit mode, allowing any text expression to be entered. A right click while in edit mode will display the cut-copy-paste menu.

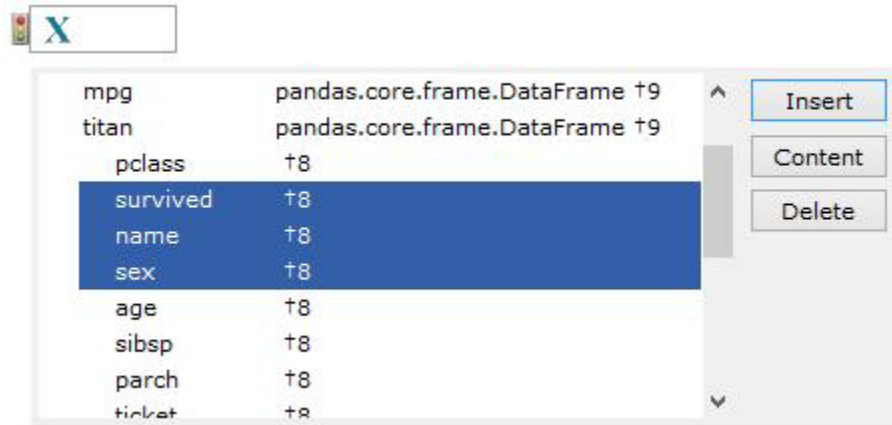


Right clicking on an options input will display a menu of entries to choose from. Left clicking on an options input will put it into edit mode where a specific value may be entered.

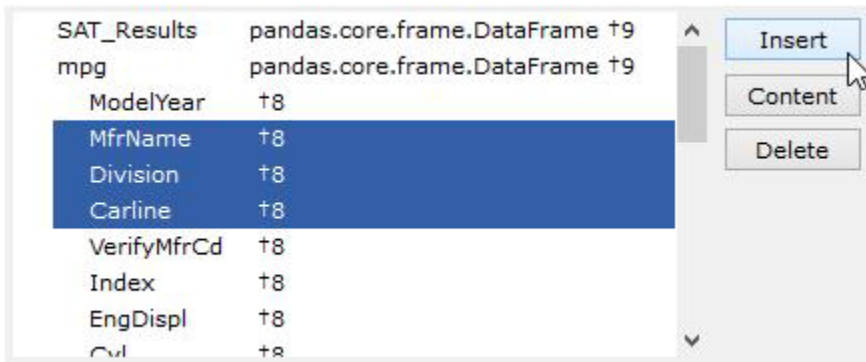


ArrayLike

Array-like elements expect the name of arrays, DataFrames, or columns. Left clicking on an input element will cause it to go into edit mode, allowing any text expression to be entered. A right click while in edit mode will display the insert-cut-copy-paste menu.



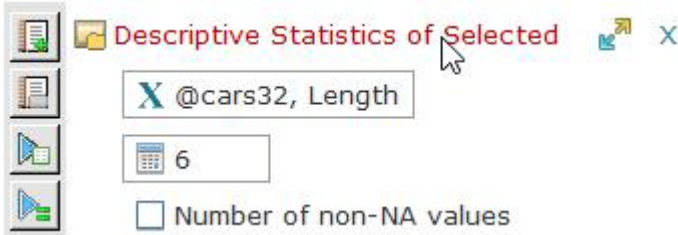
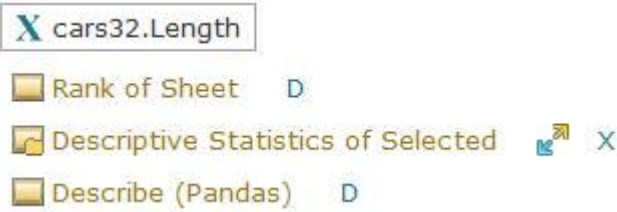
Double clicking on a DataFrame entry will expand below it to show the available columns. A second double click will collapse the view. (An entry with a dagger followed by 9 indicates a DataFrame and therefore it can be expanded and collapsed.)



In edit mode the element will accept either:
 Standard pandas expressions: DataFrame['column',...] and DataFrame.column or Visral's form whose EBNF of is: "@<DataFrame>{"," <column>}*" .

Argument source

Information used by operations can be entered in several locations. The following table explains the rules used to decide which will be the source.

<p>Highest priority 1</p>	<p>Child element of the parent operation overrides all other possible sources of parameter values.</p> 																									
<p>2</p>	<p>Preceding sibling to the operation. There can be multiple sources for the same parameter. The issue is resolved with the closest to operation having the highest priority.</p> 																									
<p>3</p>	<p>Dataset sheet selections for column parameters.</p> <table border="1" data-bbox="435 1100 1000 1283"> <thead> <tr> <th></th> <th>C/x</th> <th>D/y</th> <th>E</th> <th>F/x</th> </tr> </thead> <tbody> <tr> <td></td> <td>CITY</td> <td>HWY</td> <td>WEIGHT</td> <td>CYLINDERS</td> </tr> <tr> <td></td> <td>19</td> <td>30</td> <td>3545</td> <td>6</td> </tr> <tr> <td></td> <td>23</td> <td>31</td> <td>2795</td> <td>4</td> </tr> <tr> <td></td> <td>23</td> <td>32</td> <td>2600</td> <td>4</td> </tr> </tbody> </table>		C/x	D/y	E	F/x		CITY	HWY	WEIGHT	CYLINDERS		19	30	3545	6		23	31	2795	4		23	32	2600	4
	C/x	D/y	E	F/x																						
	CITY	HWY	WEIGHT	CYLINDERS																						
	19	30	3545	6																						
	23	31	2795	4																						
	23	32	2600	4																						
<p>Lowest priority 4</p>	<p>The arguments used at the time an Operator was embedded in a document are returned as immediate values so the same operation can be rerun. Any new value entered has higher priority to allow testing different hypothesis.</p> <p>Preset: <code>\$x @cars32,Length</code></p>																									

Notes:

- Following siblings to an Operator have no claim and do not contribute values to the operation.
- Priorities are based on individual parameters. In other words, the X columns may be chosen from the dataset sheet and a Y column could be indicated by a child location.

Spreadsheets/Datasets

Covers:

- ✓ Loading, viewing, and editing dataset content
- ✓ SERIES sheet
- ✓ Spreadsheet Formulas and Active Cells

From its inception the computerized spreadsheet has used simple cell formulas to perform calculations. While a convenient concept, by its nature it removes one of the most common and powerful tools in the programmer's bags of tricks; recursion.

Although the Visral spreadsheet allows distributing formulas among cells, it is the replacing of the calculate button with an unlimited number of Operators with the ability to contain as complex code as needed, including subroutines and recursion.

Visral's spreadsheets are intended to work in conjunction with datasets; figuratively as an overlay. For example, in the case of an income statement, each year's worth of actual monthly revenue and expenses would be considered a dataset. The total monthly and yearly revenue, expense, and pre-tax and net income would flow to the Python based output cells for display.

The Difference

Static in design, the common spreadsheet's primary dynamics is changing values or pushing the calculate button. On the other hand, Visral's is meant for the interactive funneling of diverse compositions of data to a variety of powerful Operators with real-time data collection and updating of results and predictions.

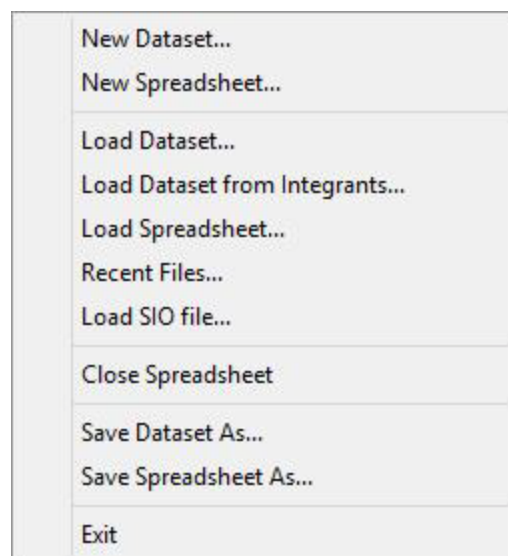
The Sheets window is divided into two sections. Dataset content is loaded into passive cells located in the lower portion with the black line number. The spreadsheets with the active cells that can contain Python instructions are located in the top section with the red line numbers.

Data Sets

The dataset sheet is brought into view by first selecting the **Sheets** button on the toolbar and then choosing the particular the appropriate tab referencing a previously loaded dataset.

Dataset files can be accessed via the folder and floppy buttons, and from the File dropdown menu. When a dataset is loaded, a DataFrame copy is also created. These are two distinct copies of the same data. The spreadsheet can be edited randomly cell by cell but the corresponding DataFrame will need to be updated via the **Munge / Calculate Panel** to reflect those changes.

The **New Spreadsheet...** and **Save Spreadsheet As...** entries are not functional in OE.



File Extensions	Description
*.csv	Comma Separated Values (Visral default for datasets)
*.xls	Excel (Uses Python and Pandas and xlrd to load files)
*.sxml	Visral Spreadsheet (unstructured data/instruction)
.dta.sav*.sas*.json*.hdf	Requires Python and Pandas module

Moving dataset content view

The page up and page down keys will cause the view of the dataset contents to scroll up and down.

Using the scroll wheel on the mouse will cause the view of the dataset contents to be scrolled up or down if the cursor is within the body of the sheet or on the row numbers.

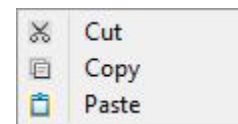
The scroll wheel will cause the view of the contents to be scrolled left or right if the cursor is on the row of column letters. (The scroll wheel will change the width of individual columns when on the row of column names.)

The small red square is a replacement for the two traditional scroll bars and allows easy two dimensional movements. Pressing and holding the left button on the small floating red box while dragging it will cause the window to move about the dataset contents. When clicked the size of the box changes to represent the relative size of the view to the total contents.

543	543	male	36	0	0
544	544	male	34	1	0
545	545	female	30	3	0
546	546	female	28	0	0
547	547	male	23	0	0
548	548	male	0.83	1	1
549	549	male	3	1	1
550	550	female	24	2	3
551	551	female	50	0	0

Editing Dataset Cell Content

Double clicking on any cell will cause it to enter edit mode. Once in edit mode other cells can be edited by a single left click or using the up, down, left, and right cursor keys. A right click on a cell in edit mode will bring up the Cut, Copy, and Paste menu. A right click on any other cell or a carriage return will cause edit mode to be exited.



Selecting Columns

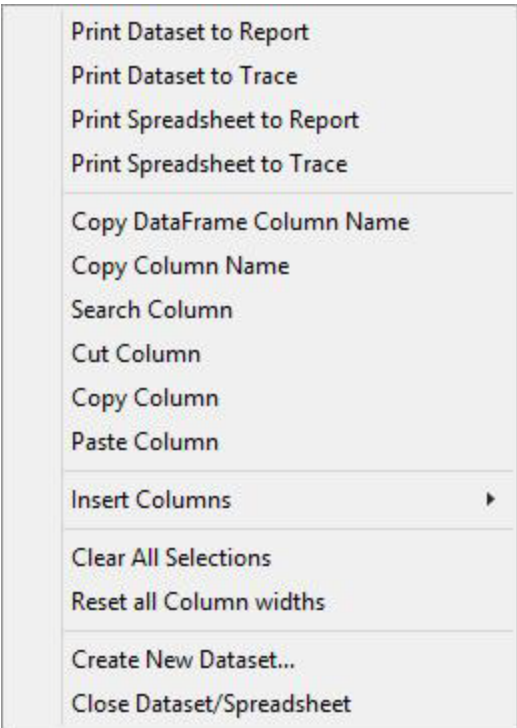
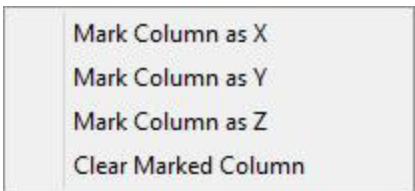
Double clicking on the column head of the top row of sequential letters will cycle the column through three different column references; X, Y, and Z, and then no assignment. The color of the columns will change along with each reference. There are two other methods by which to select column assignments; through the Column Menu and through the Column Management Menu. The last is particularly useful when there are a large number of columns. (Up to 4,000)

To select more than one column from the dataset simply hold down the left button and drag the cursor across adjacent columns.

Column Menu

A Right click on top row of sequential letters will bring up the column operation menu.

Undo is not enabled for column operations. This is to avoid over consumption of memory due to column lengths potentially being up to a million rows. If not sure use cut; paste will provide one undo.



A Right click on column title row will bring up a more comprehensive column operation menu.

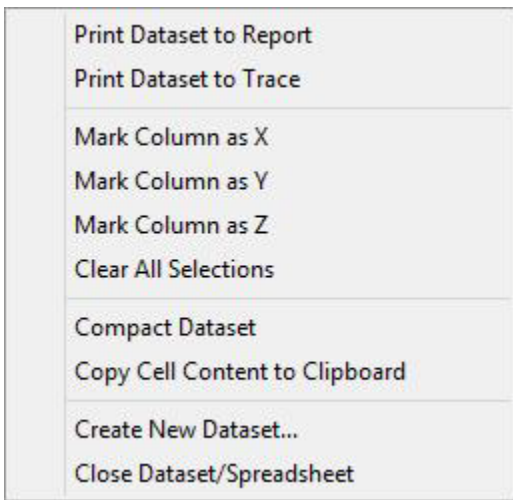
The Mark Column as X Y or Z is equivalent to double clicking on the column heads.

Insert Empty Column does just that and is useful for making room to manually adding a column of data. The new column is inserted to the left of the column this menu was activated on.

Insert Column with Ones places a number 1 in each cell of a new column.

Cell Menu

A right click on any cell when not in edit mode will bring up the cell operation menu. The **Copy Spreadsheet to Report** and **Copy Spreadsheet to Trace** will produce a table of the selected entries. If there are no selections then the entire spreadsheet will be used.

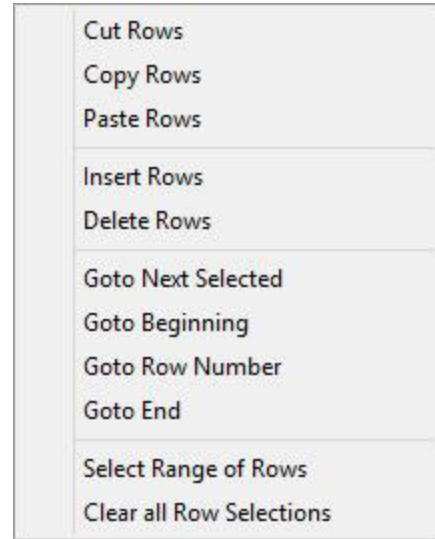


Row Menu

A right click on leftmost column of sequential numbers will bring up the row operation menu. Holding down the left button and dragging it down the column of row numbers will select them. If the row was previously selected when the button pushed, then rows will become unselected. The left upper corner of the sheet window has a box with **Clear** in it. Left clicking this will unselect all previous selections.

The row menu various operations of which Cut, Copy, and Delete require rows be selected first. Undo and redo work on all row modifications; that includes Cut, Copy, Paste, Insert, and Delete.

The Goto's and Select Range are particularly useful when working very large datasets.



Capturing dataset data

Unlike conventional spreadsheets, Visral's can assemble results from any combination of rows and columns. This is true whether printing a table or generating a DataFrame, array or list. The image below illustrates creating a table from a combination of columns and rows.

Clear	A	D	E/x	F	G	H/x	I/x	J/x	
Layer D		ENTRSIZE	FIRM	ESTB	EMPL_N	EMPLFL_R	EMPLFL_N	PAYR_N	PAYR
43179	4317	02	117	117	188		G	6487	G
43180	4318	03	36	36	229		H	6753	G
43181	4318	04	17	17	243		H	9083	H
43182	4318	05	170	170	660		G	22323	G
43183	4318	06	17	17	518		G	19715	G
43184	4318	07	3	3	0	E	D	0	D
43185	4318	08	190	190	1596		G	64255	H
43186	4318	01	190	190	1596		G	64255	H
43187	4318	02	117	117	188		G	6487	G
43188	4318	03	36	36	229		H	6753	G
43189	4318	04	17	17	243		H	9083	H
43190	4319	05	170	170	660		G	22323	G
43191	4319	06	17	17	518		G	19715	G

	FIRM	EMPLFL_R	EMPLFL_N	PAYR_N
43182	170		G	22323
43183	17		G	19715
43184	3	E	D	0
43185	190		G	64255
43188	36		H	6753
43189	17		H	9083

Holding down the left mouse button on a cell and dragging to another cell will select that range when the button is released. This illustrates the process, following which a DataFrame, array or list could be generated, or a table produced.

Birnbaum, Mr. Jakob	male	25	
Bishop, Mr. Dickinson H	male	25	
Bishop, Mrs. Dickinson H	female	19	
Bissette, Miss. Amelia	female	35	
Bjornstrom-Steffansson, M	male	28	
Blackwell, Mr. Stephen W	male	45	
Blank, Mr. Henry	male	40	

Birnbaum, Mr. Jakob	male	25	0
Bishop, Mr. Dickinson H	male	25	1
Bishop, Mrs. Dickinson H	female	19	1
Bissette, Miss. Amelia	female	35	0
Bjornstrom-Steffansson, M	male	28	0
Blackwell, Mr. Stephen W	male	45	0
Blank, Mr. Henry	male	40	0


When copying to the Report or Trace, the relative width of the columns will be the same as the displayed dataset. Although if running on Windows 8 or 10, they can subsequently be adjusted, it is more convenient to do on the dataset beforehand. This can be accomplished by placing the cursor on the column name and using the scroll wheel.

Selections are saved individually for each dataset open. Therefore, selections will not be lost when switching between them.

Selected data as a source for Operators

If operations are executed where the Sheet's dataset is the source of the data (selected), then it comes from the temporary DataFrame vr_temp.

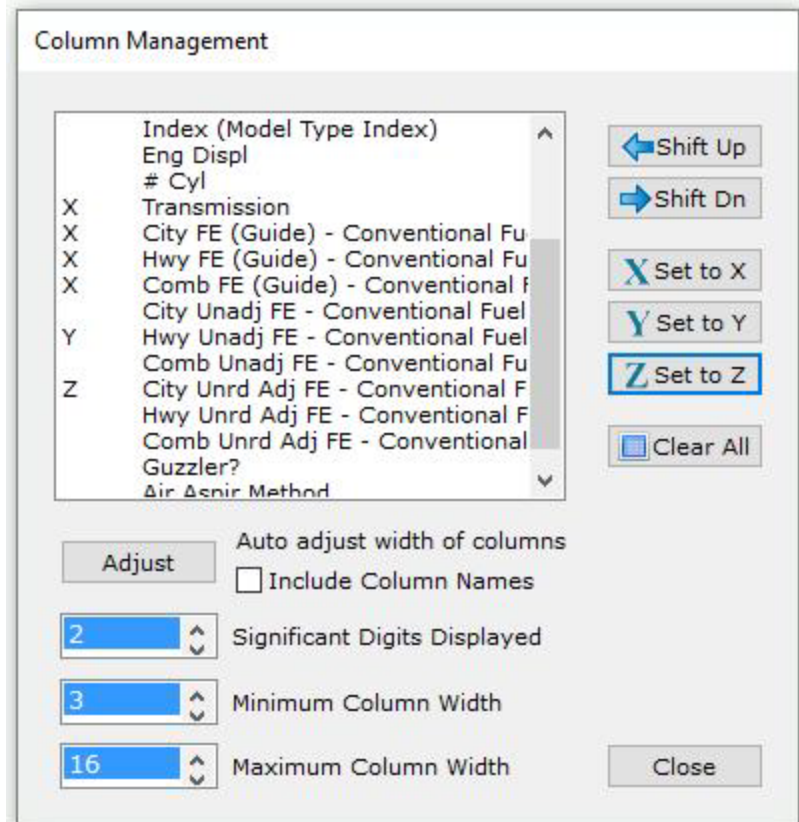
Manipulating the Spreadsheet/Dataset Columns

 Bring up the column management menu. The **Shift Up** and **Shift Dn** buttons allow rearranging the column of the current spreadsheet/dataset. The selected entries will move one place to the left or right each time the buttons are clicked.


The **Set to X**, **Set to Y**, and **Set to Z** buttons can assign any collection (selected) of columns to be used as sources for Panel operations. The **Clear All** clears all selections, both rows and columns of the current spreadsheet/dataset. This provides a more convenient means for selecting columns when there are a large number of them.

Row and columns assignments are not lost when moving between displayed datasets.

Include Column Names and **Minimum Column Width** provide automatic column name width compensation for datasets and printouts. They are not functional in OE.



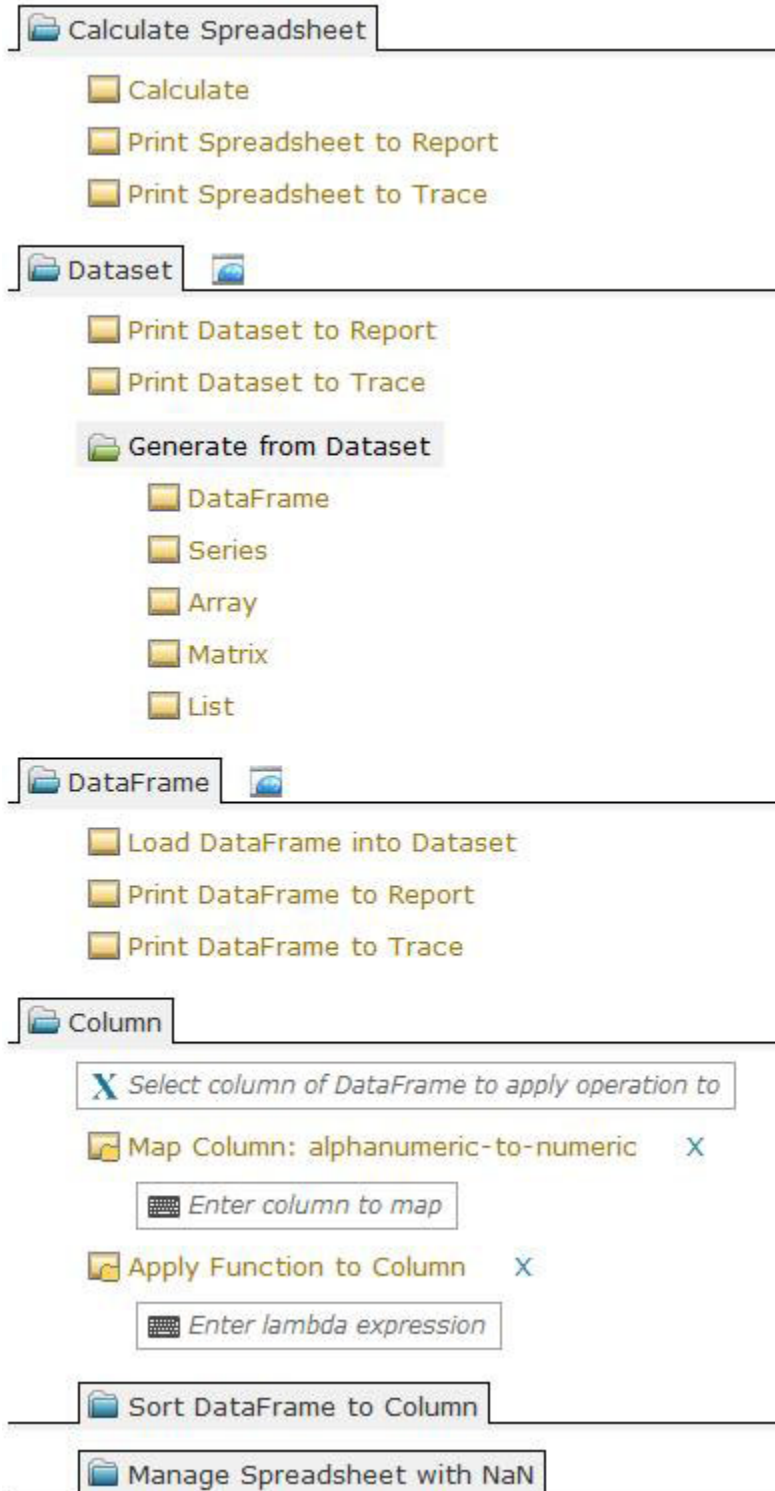
Munge / Calculate Spreadsheets and Datasets

 Beyond files, dataset sheets can be loaded from Python DataFrames, matrixes, and arrays, but they are copies of the data. If they are edited and it is desired to have the original source reflect the changes, the DataFrame sources will need to be updated.

New DataFrames, matrixes, arrays, and series can be generated from dataset sheets, either partially by selecting rows and columns or fully by clearing all selections.

The Munge part of the Panel refers to operators for generating DataFrames, Series, arrays, matrixes, and Lists. The Munge Panel can be found in the Morphing menu and in the main menu's SYSTEM dropdown.


The Calculate Operator simply executes any code that might reside within the cells. The real power of Visral is the user's ability to create an array of Operators that implement sophisticated solutions.

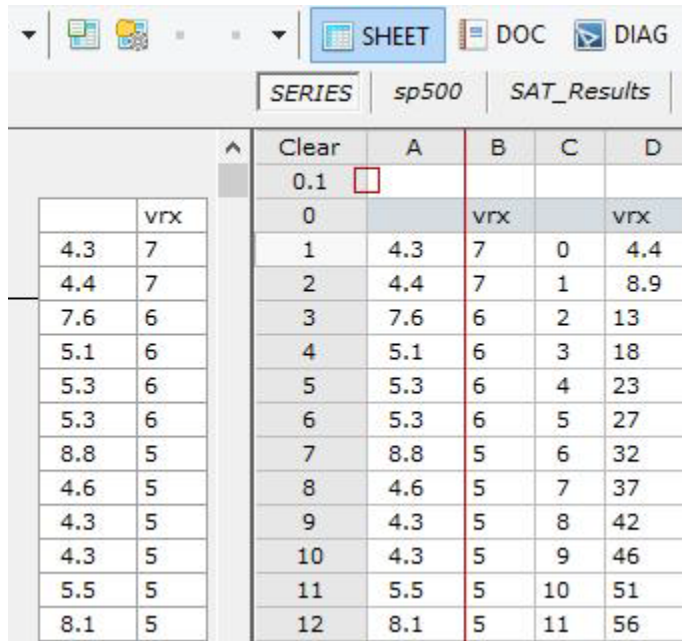


The screenshot displays a vertical menu structure with the following sections and options:

- Calculate Spreadsheet**
 - Calculate
 - Print Spreadsheet to Report
 - Print Spreadsheet to Trace
- Dataset**
 - Print Dataset to Report
 - Print Dataset to Trace
 - Generate from Dataset**
 - DataFrame
 - Series
 - Array
 - Matrix
 - List
- DataFrame**
 - Load DataFrame into Dataset
 - Print DataFrame to Report
 - Print DataFrame to Trace
- Column**
 - Select column of DataFrame to apply operation to
 - Map Column: alphanumeric-to-numeric
 - Enter column to map
 - Apply Function to Column
 - Enter lambda expression
- Sort DataFrame to Column
- Manage Spreadsheet with NaN

SERIES

The SERIES is a special spreadsheet, which collects all Python pandas Series printouts when using the  Tracking toolbar button. It acts in a push down fashion with the newest entering into columns A and B with the rest shifted to the right. Both columns of the Series are included. This allows examining or extracting subsets from Series results that can be up to a million rows in length.



	Clear	A	B	C	D
	0.1				
	0		vrX		vrX
	1	4.3	7	0	4.4
	2	4.4	7	1	8.9
	3	7.6	6	2	13
	4	5.1	6	3	18
	5	5.3	6	4	23
	6	5.3	6	5	27
	7	8.8	5	6	32
	8	4.6	5	7	37
	9	4.3	5	8	42
	10	4.3	5	9	46
	11	5.5	5	10	51
	12	8.1	5	11	56

The same operations performed on all other spreadsheets can be used in the SERIES spreadsheet.

To extract a pandas Series from the spreadsheet select a column or a portion of one and click the **Generate Series from Spreadsheet** entry of the Munge Panel. A popup will follow allowing a name to be assigned to the new Series.

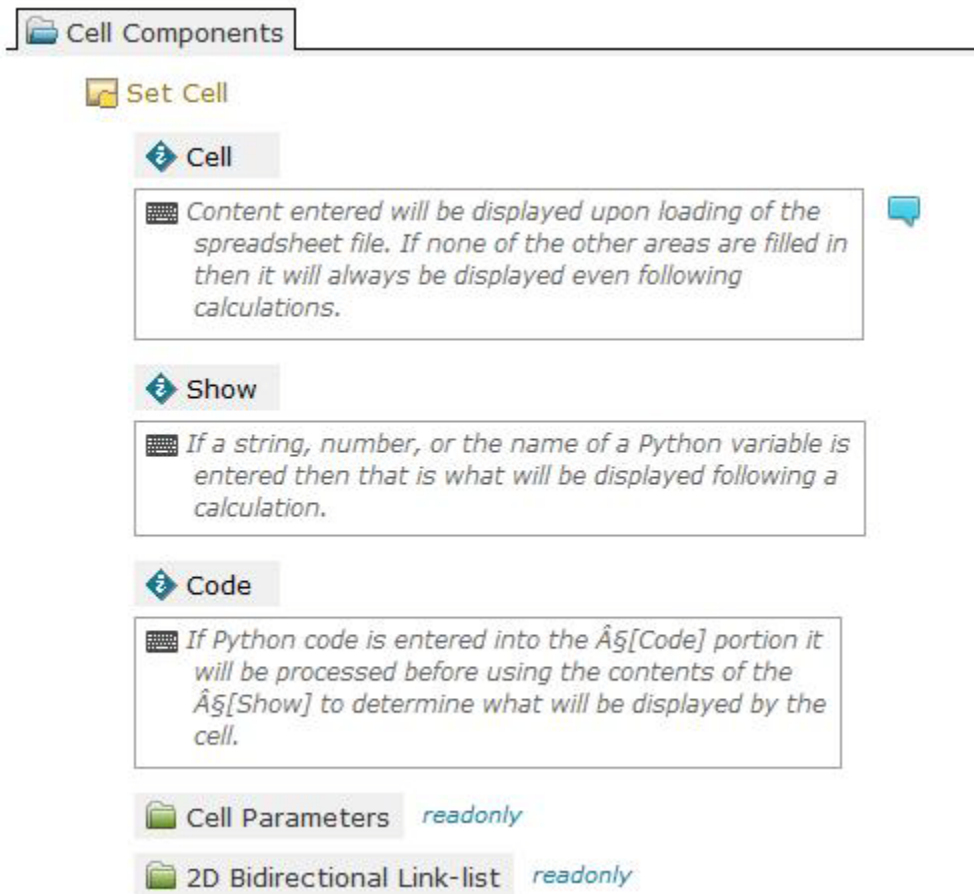
Spreadsheet Cells, Formulas, and Execution



The Sheets window is divided into two sections. Dataset content is loaded into passive cells located in the lower portion with the black line number. The spreadsheet's active cells are located in the top section with the red line numbers. These cells can contain Python instructions.

If the spreadsheet section is not visible, place the cursor on the left most column and use the scroll wheel of the mouse to scroll down. That will bring the spreadsheet section into view.

Double clicking on an empty cell will cause the *EDITOR* Panel to display three titles separating it into three areas as illustrated below.



Content entered below the **Cell** title will be displayed upon loading of the spreadsheet file. If none of the other areas are filled in then it will always be displayed even following calculations.

If a string, number, or the name of a Python variable is entered into the **Show** portion then that is what will be displayed following a calculation.

If Python code is entered into the **Code** portion it will be processed before using the contents of the **Show** to determine what will be displayed by the cell.

After making any desired alterations, click the **Set Cell** Operator in the Panel

Use `vr.gen(calculate)` at the end of any Operator code to cause the spreadsheet to be processed when the Operator is selected.

Here is a simple example. When first loaded the cell displays the number 12. (It could have as well been a string.) The cell content also reflects this.

After the calculate function is processed it can be seen that the code produced and displayed a new result.

	Before	After <code>vr.gen(calculate)</code>
Spreadsheet Cell	<input type="text" value="12"/>	<input type="text" value="5"/>
<i>EDITOR</i>		

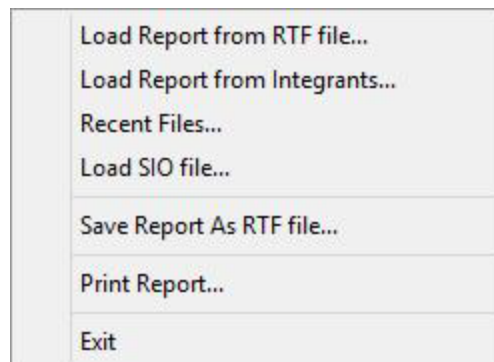
Documents

Covers:

- ✓ Report and Trace Editing
- ✓ Embedding
- ✓ Moving and copying embedded items
- ✓ Report and Guide response

The Report, Guide, and Trace editors are selected by first selecting the **Docs** button on the toolbar and then choosing from one of the three tabs. Only a single file can be opened for each of the RTF based Report, Trace, and Guide editors at one time.

This is the file menu when the Report has focus. The Guide and Trace have similar file menus.



The textual and graphical results of Python operations are directed to the Trace (default) and Report editors where they can be supplemented with the author's own text, spreadsheet printouts, and the pasting of photos and other images. Beyond that Visral allows the embedding of datasets, Venues, Operators, and sections of code into the Report.

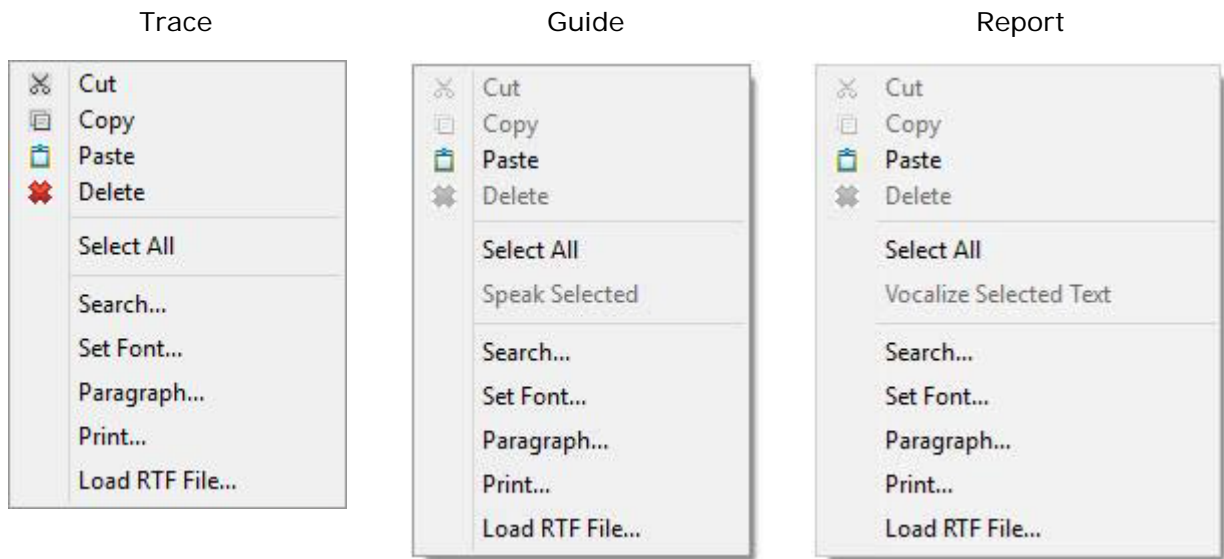
Report, Guide, and Trace Editing



Both the Report and Trace editors have search ability and, paragraph and character formatting, as well as accept the pasting of images.

A right click on the background brings up the menus as illustrated below. They provide access for searching, setting font, formatting, printing, and the loading of a document. The Report menu also offers the ability in embed active features into the documents. There is duplicate access to some of these features available under EDIT of the main menu.



Note: In Windows 8 and later, columns and rows of tables in RTF documents can be resized.



This table details the actions of the Morphing menu for the Report.

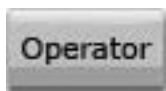
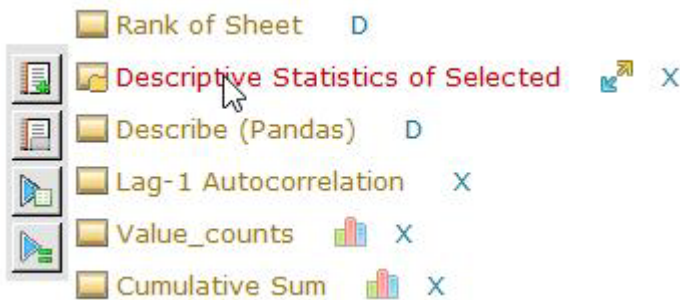
Button	Description	
	Left clicking on the Report icon with the plus symbol will execute the selected code from the previously PAD to have focus and send results to the Report editor.	
	Left clicking on the Report icon with the button symbol will embed the selected code from the previously PAD to have focus, into the Report editor as a button. When the button is clicked in the Report of Guide a Panel entry with the code will be created. (See document section for details.)	
	Left clicking on the Report icon with the gear symbol will embed the selected code from the previously PAD to have focus, into the Report editor as a button. When the button is clicked in the Report of Guide the code will be executed. (See document section for details.)	
	Left clicking on this will bring up the passage management menu. See Visral User Manual, Volume 2 for details of advanced features.	

This table details the actions of the Morphing menu for the Guide.

Button	Description
	Enables and disable machine voice reading of button descriptions when they are clicked. (This feature in not available in OE.)
	Directs the results of execution by Code buttons to the Trace or Report documents.

Embedding Operator

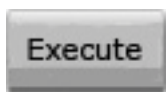
Left clicking on the Embed Operator entry will embed the currently red highlighted Operator the showing Panel, into the Report document. The Operator will be inserted into the Report document at the location of the original right click. It will appear as a button with the word **Operator** on it.



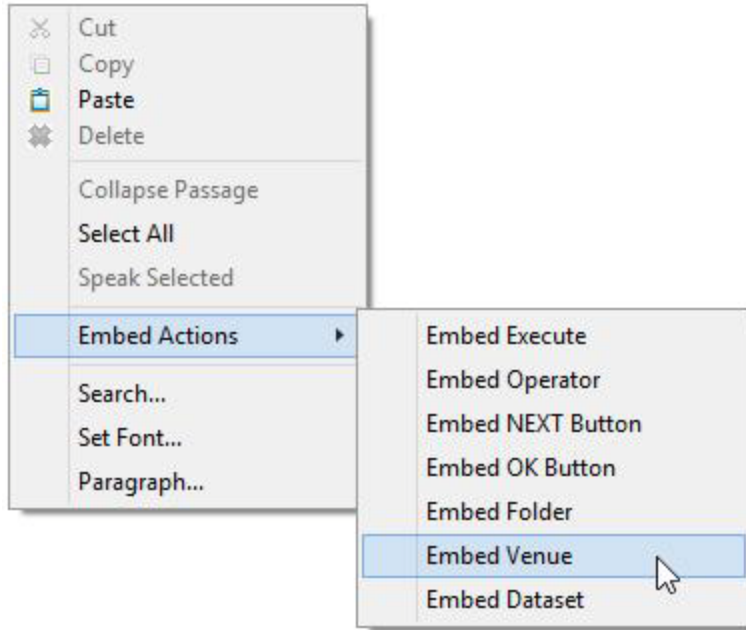
Once embedded, clicking on the Operator button will cause a new Panel to be displayed with the original operator, which if clicked will cause the operation to be executed.

Embedding Code

Left clicking on the Embed Execute entry will embed the selected code from the previously PAD to have focus, into the Report document. The code will be inserted into the Report document at the location of the original right click. It will appear as a button with the word **Execute** on it.



Once embedded, clicking on the Code button will cause original code to be executed with the results going to the Trace. If the button is clicked from the Guide, the results can be direct to either the Trace or Report.



Embedding Venue



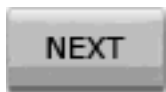
Right clicking will provide access to a list the embeddable components. Venues can be embedded into RTF documents and recalled and installed in other Visral applications. Clicking the Venue button will show a file directory set to the default location and the name of the file that was embedded, both of which can be altered.

Embedding Dataset



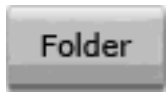
Datasets can be embedded into RTF documents allowing them to be distributed to other Visral installations. Clicking the Venue button will bring up a menu for selecting a DataFrame name and loading the dataset.

Embedding NEXT Button



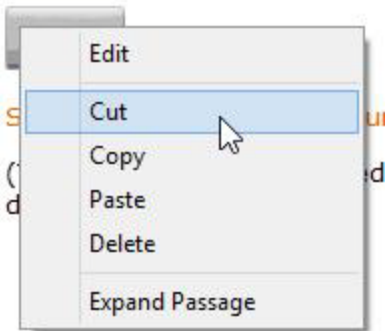
The NEXT Button can be embedded into RTF documents allowing users to step through a sequence of operations. Clicking the Next button will bring up a menu.

Embedding Folder



The Embed Folder entry button will embed the folder with all of its contents in the Report at the location of the cursor and display the Folder button.

Moving and copying embedded items

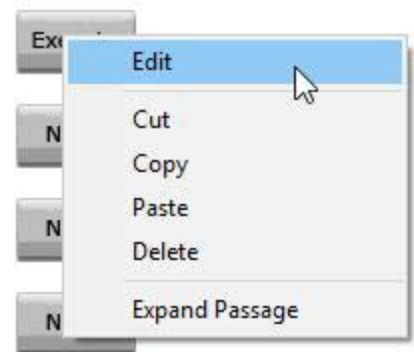


A button must not be dragged to a different position or moved by using the normal select, cut and paste. This is to assure that all of the embedded material associated with the image is caught and moved together. To move or copy right click on the button. Now the cut, copy, and paste will work properly. This will select all relevant material associated with the button and its embedded function. Now a cut or copied button can be pasted.

To access the menu to do this, place the cursor on the button and click the right mouse button. Coping and deleting a button also requires this procedure to avoid leaving dangling embedded components around.

Editing embedded items

Embedded components can be edited in the Report by right clicking on a button and selecting the first entry of the context menu. That will cause the contents of button to be edited in the *EDITOR* Panel. The following is an example of how the editable content might appear.



Passage Components

Set Passage

Description

Execute

Program

```

size = 10000
x = sp.arange(size)
y = sp.int_(sp.round_(sp.stats.vonmises.rvs(5
,size=size)*31))
h = plt.hist(y, bins=range(32), color='w')
disttypes = ['gamma', 'beta', 'norm']
for disttype in disttypes:
    dist = getattr(sp.stats, disttype)
    param = dist.fit(y)
    pdf_fitted = dist.pdf(x, *param[:-2],
loc=param[-2], scale=param[-1]) * size
    plt.plot(pdf_fitted, label=disttype)
    plt.xlim(0,31)
plt.legend(loc='upper right')
plt.show()
    
```

Passage Components

Set Passage

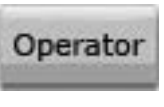
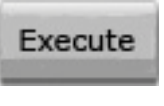
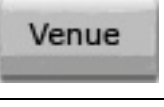

Description


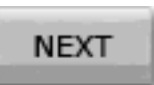
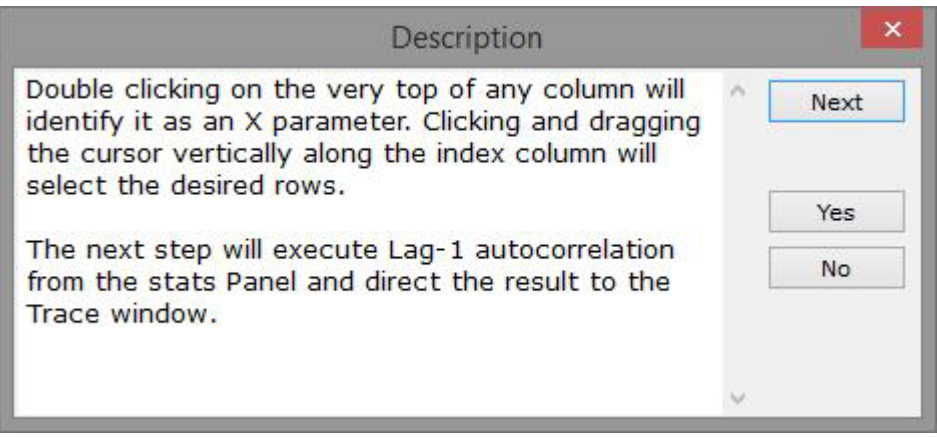
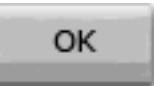
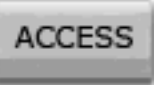
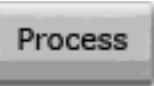
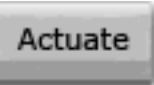
The text represents what will be displayed in the popup Description window and optionally machine spoken within the Guide.

Program

This location contains program code that will be executed after the button is pressed.

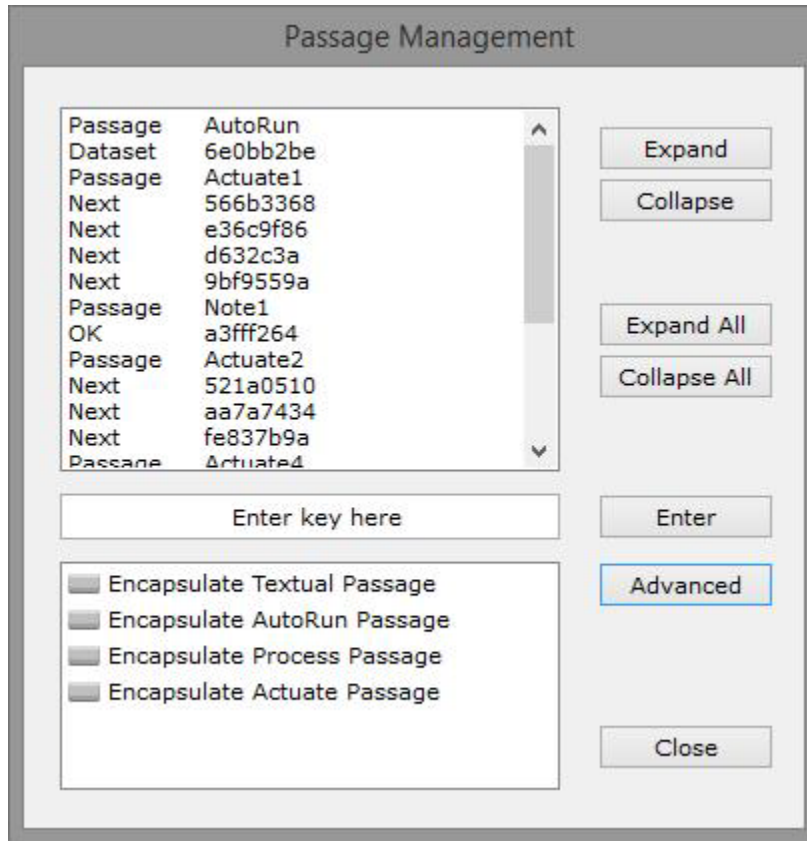
Guide embedded response

	<p>Once embedded, clicking on the Operate button will cause a new Panel to be displayed with the original Operator, which if clicked will cause the operation to be executed.</p>
	<p>Once embedded, clicking on the Code button will cause original code to be executed with the results going to the Trace. If the button is clicked from the Guide, the results can be direct to either the Trace or Report.</p>
	<p>Clicking the Venue button will show a file directory set to the default location and the name of the file that was embedded, both of which can be altered. The newly load Venue will not be available until Visral is restarted.</p>
	<p>Clicking the Data button will display a menu for specifying the name of the DataFrame to be replaced or created. It will be loaded into a spreadsheet at the same time. The operation can be aborted with the cancel button.</p> <div data-bbox="787 1249 1421 1837" style="border: 1px solid gray; padding: 10px;"> <p style="text-align: center;">Load Document DataSet</p> <p>df0 (Unassigned name) cars32 (Button ID) cars20 cars32 chmovie titan</p> <p style="text-align: right;">OK Cancel</p> </div>

	<p>Clicking the Folder button will show a previously saved folder and all of its Operators in a Panel.</p>
	
 	<p>Unlike the NEXT button, these cause whatever code is assigned to them, to execute immediately when the button is pushed.</p>
	<p>Items combined into this passage are executed in sequence when the button is clicked. It will pause when user inputs are required.</p>
	<p>Items combined into this passage are executed in sequence when the button is clicked. Unlike the Process passage, it uses default values where available and does not pause for user responses except for the Next commands.</p>

The document editors permit font selection and coloring, while the paragraphs can be formatted and justified. Microsoft Office word processors accept RTF files and can be used for further editing, as well as generating HTML code and Adobe Acrobat files.

Passage Management

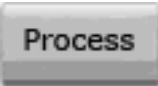



The Passage Management menu allows the combining of multiple buttons and text into a single embedded entity called a passage.

A selection of texts and buttons are combined together using the encapsulate commands above, and replaced by a single button from the following table.

Double clicking on an entry in the encapsulate list will apply the appropriate wrap the selected text and buttons.

Encapsulate Table

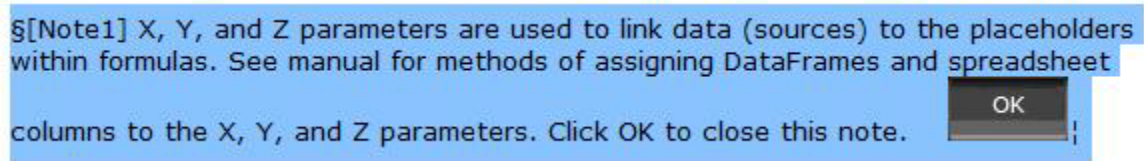
Under Program Control	Encapsulate Textual Passage ; allows program control of the display of sections of text and any included buttons. vr.set.textual(<text ID>,<action>) <action>={1:show, 0:hide}
	Encapsulate AutoRun Passage ; items combined into this passage are automatically executed in sequence when the document is loaded into the Guide only. Like the Actuate passage, it uses default values where necessary; i.e. does not require user responses.
	Encapsulate Process Passage ; items combined into this passage are executed in sequence when the button is clicked. It will pause when user inputs are required.
	Encapsulate Actuate Passage ; items combined into this passage are executed in sequence when the button is clicked. Unlike the Process passage, it uses default values where available and does not pause for user responses except for the Next commands.

Textual passages do not produce a button replacement. Their id code is used with the `vr.set.textual(...)` instructions to hide, display, or otherwise apply their content. Textual passages can include buttons, permitting conditional dialog based on the results of computations.

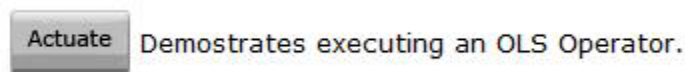
The following are two examples of how sections of text and buttons appear after encapsulation. (Except for the blue background which is here simply to highlight the section.)

Encapsulation merely involves surrounding the selected text, beginning with a section character followed by a unique identifier in square brackets and ending with a broken vertical bar character.

The first seven characters of the identifiers for AutoRun, Actuate, and Process are required to indicate passage type, but if more than one of either exists they will require more characters concatenated with them to provide uniqueness. As an example, the 2 added to the Actuate entry below.



The passages are transformed when the Collapse button in the Passage Management menu is selected. In that case the Textual passage, it disappears and can only be displayed by running a `vr.set` command. The Actuate passage appears as below.



Encryption

Both private-public and symmetrical encryption facilities are available to secure passages and code components. The Producer refer to is the one who creates encrypt content, Recipient to those receiving that encrypted content, and Personal refers to encrypted content received from others.

Python Control of Visral Assets

The following are commands recognized by Python with visral.pyd installed, which happens automatically during the startup of Visral.

visral.pyd	Output	Description
vr.show()	Yes/no	Show Python images in Report or Trace.
vr.gen(...)	Yes/no	General operations. This instruction triggers a Visral command and prevents any further Python instructions from executing until Visral has completed the task.
vr.set(...)	no	These instructions trigger Visral operations that are not Python dependent.
vr.ask(...)	yes	Retrieve Visral parameters or settings.
vr.job(...)	yes	Issues job requests to bridges, including such functions as encrypt/decrypt passages, code, and commands.
vr.run	yes	This instruction triggers a command and stalls Python instructions from executing until the command has been completed. For use with Universal Bridge applications.
vr.bitmap	yes	For internal use only.
vr.inset		These instructions are used internally, do not pass through Python, and although they maintain order amongst themselves, are executed before any related Python code.

Notes:

1. Visral converts plt.show() and fig.show() to vr.show(plt) and vr.show(fig) respectively. Pandas plotting functions in particular need to be followed by the plt.show() in order to output a Python image to the Trace or Report.
2. The vr.set instruction triggers a one-way command from Python to Visral. This means Python will continue executing instructions even if Visral is still working its task. However, Visral commands are scheduled through a FIFO so as they are always performed in order.
3. The commands of vr.set, vr.gen, and vr.run can be expressed in either of two ways; vr.set(command,...) or vr.set.command(...), vr.gen(command,...) or vr.gen.command(...), and vr.run(command,...) or vr.run.command(...). They both execute in the same fashion.
4. Individual arguments need only be wrapped in single or double quotes if they contain commas or parentheses. Only double quotes may be used internally within an argument. In no case must the argument end in a backslash, with or without quotes. If it is necessary to end any Python string with a backslash, try ending it as follows: "...\\ "[:-1] (That's an extra space between the backslash and the ending quote.)

vr.ask	Description
(vrplot)	A True response indicates the auxiliary plot button was clicked.
(vrlist)	A True response indicates the auxiliary list button was clicked.
(vrflip)	A True response indicates the auxiliary flip button was set.
(vryes)	A True response indicates the Yes button was clicked.
(vrno)	A True response indicates the No button was clicked.

vr.inset	Description
(transfer,'expr',0)	A file transfer a file defined in 'expr' expr = <src>; <dest>; <task>
(buildSIO,"",0)	Build an SIO file from the preceding transfers.
(saveSIO,'expr',0)	Save the SIO file at the location specified in 'expr'

vr.job	Description
()	Encrypt/Decrypt passages, code, commands and requests.

vr.gen.	Description
calculate()	Process spreadsheet
comment(Comment)	Display comment beside red arrow pointer.
compile()	Compile Rule Set
csv(expr)	
dataset(array)	Generate Array from Spreadsheet. Brings up a menu of existing and new Arrays from which to choose from.
dataset(dataframe)	Generate DataFrame from Spreadsheet. Brings up a menu of existing and new DataFrames from which to choose from.
dataset(list)	Generate List from Spreadsheet. Brings up a menu of existing and new Lists from which to choose from.
dataset(matrix)	Generate Matrix from Spreadsheet. Brings up a menu of existing and new Matrixes from which to choose from.
dataset(series)	Generate Series from Spreadsheet. Brings up a menu of existing and new Series from which to choose from.
dataset(report)	Print all or selection of Dataset to Report at current cursor location.
dataset(trace)	Print all or selection of Dataset to Trace at current cursor location.
df(expr)	Generate DataFrame named expr from top Spreadsheet. Selected rows and columns apply.
diag()	
doc(N)	Select document N = {0:return to last document, 1:go to Report, 2:go to Guide, 3:go to Trace}
emulate()	Emulate using selected test rules
fill()	
formulate()	
hide()	
load(expr,name)	Fill a new spreadsheet from an arraylike quantity expr. If name is present it becomes the name of the spreadsheet, else _df is appended to expr; i.e. expr_df.
mungebar()	Show the munge tracking toolbar.
notoolbar()	Do not show the tracking toolbar.

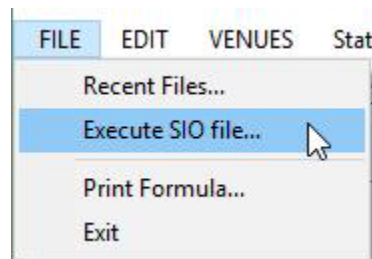
vr.gen.	Description
open(expr)	Returns the file name plus path in the Python string object vr_path and the file name alone in the Python string object vr_file. expr = default path ; extension ; menu title Example: vr.gen.open(C:/Python/Libs/;pyc;Open)
operator(\$/key text,1)	Insert text into element that has matching key.
pad(num)	Show PAD num
panel(expr,N)	Select Panel, Operator, and action N = {1:run code, 2:collapse, 3:expand, 4:show code, 20:show arrow}. expr = <panel name> <partial operator title>
passage()	
print()	
sheet(expr,N)	Select spreadsheet, columns N = {9:clear, 1:X, 2:Y, 3:Z} and rows.
spreadsheet()	Load Spreadsheet from DataFrame. Brings up a menu of existing DataFrames from which to choose from.
recursion()	Detect Recursion
table(expr)	Print table from Spradsheet expr.
textual(Note1,1)	Show "Note 1". {1:show, 0:hide}
typepad(text,N)	Type text into PAD N.
toebnf()	List Diagram Rules in EBNF
vocalize(text)	Machine voice will recite text.

Creating SIO Files

Executing an SIO File

SIO files provide a means to distribute solutions. Venues, documents, datasets, Python programs, and diagrams can all be saved and loaded individually. Venues, folders, operators, datasets, executable code sections, and passages of interactive exchanges can be embedded within documents to be used later or distributed.

SIO files can combine all of these into a single compressed entity, as well as any other files a user chooses to include. Double clicking on an entry in the file explorer with an SIO extension will cause Visral to expand, load, and process it. An SIO file can also be dragged to the left window pane or loaded through the main menu FILE dropdown.

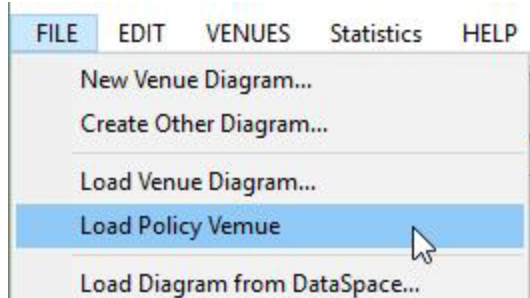


Creating an SIO build schema

The first step in creating an SIO file is to configure an Operator to generate it. This is to make it easy to repeatedly generate fresh copies of a particular SIO file. Begin the process by selecting the diagram button from the toolbar.

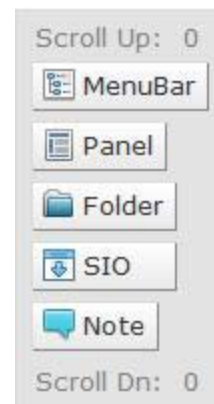


Then from the main menu FILE drop-down, select the **Load Policy Venue** entry.



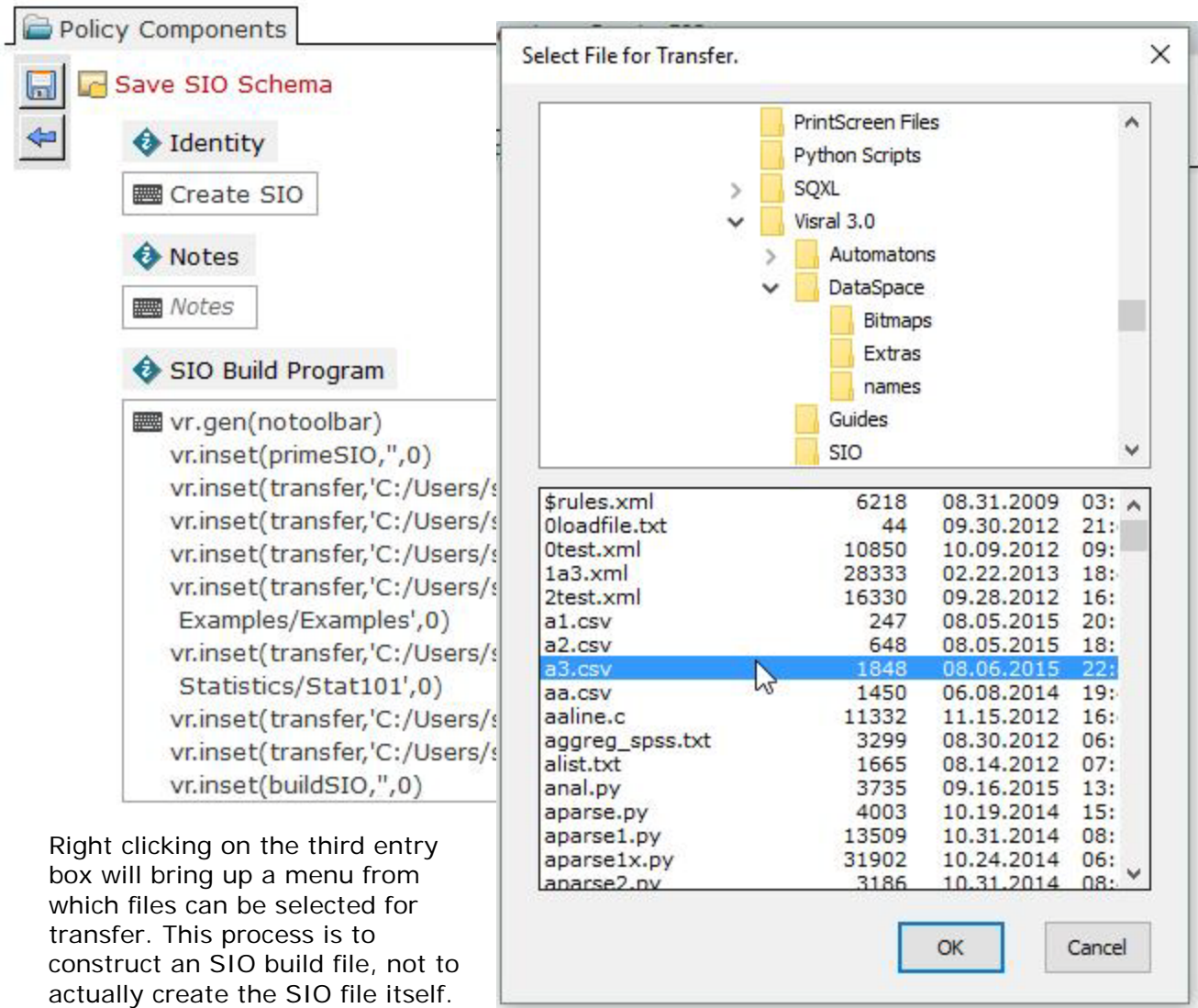
This in turn will display the construction diagram used to build SIO Schemas and the Policy element menu.

An SIO element can be inserted in the standard way as



described in the Visral Diagrams section of the manual. Clicking on the icon of an SIO element will cause an EDITOR Panel to appear where the functioning of an SIO file can be defined.





Right clicking on the third entry box will bring up a menu from which files can be selected for transfer. This process is to construct an SIO build file, not to actually create the SIO file itself.

Files to be transferred are specified with `vr.inset(transfer, '<src>; <dest>; <task>', 0)` instruction, which is generated and inserted into the listing automatically.

The tasks are added to the end of the string, following the last semicolon. The default for Python source files is PADO but can be change by hand. The default for Venues is keyword venue followed be a description of the main menu title followed by the name of the Panel that should be displayed. The following are aex examples of venue task descriptions:

- venue Examples/Examples
- venue Statistics/Stat101

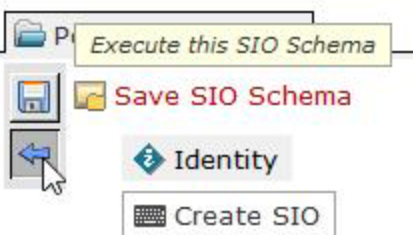
If the task is left or made blank, the file will be transferred, but not loaded into Visral.



Clicking on the Operator will cause any changes made in the EDITOR to be saved to the original element.



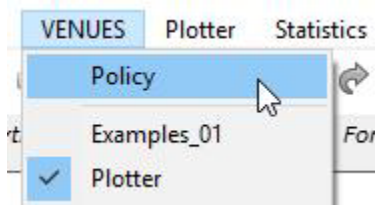
Clicking on the floppy icon will cause the SIO file described by the schema to be built. A file explorer menu will pop up to allow a destination file to be selected or created.



Clicking on the blue left facing arrow icon will cause the SIO file described by the schema to be built to an immediate file and then executed, installing it in the current program.

Building an SIO with an existing schema

To build an existing SIO file, begin by selecting the Policy entry from the VENUE drop-down and then the **Build SIO File** entry from The Policy drop-down.



That will display a Panel with all available SIO build schemas. Clicking on the desired Operator will cause the relevant file to be built.

If the following line is entered, then the result will automatically be saved to the indicated location.

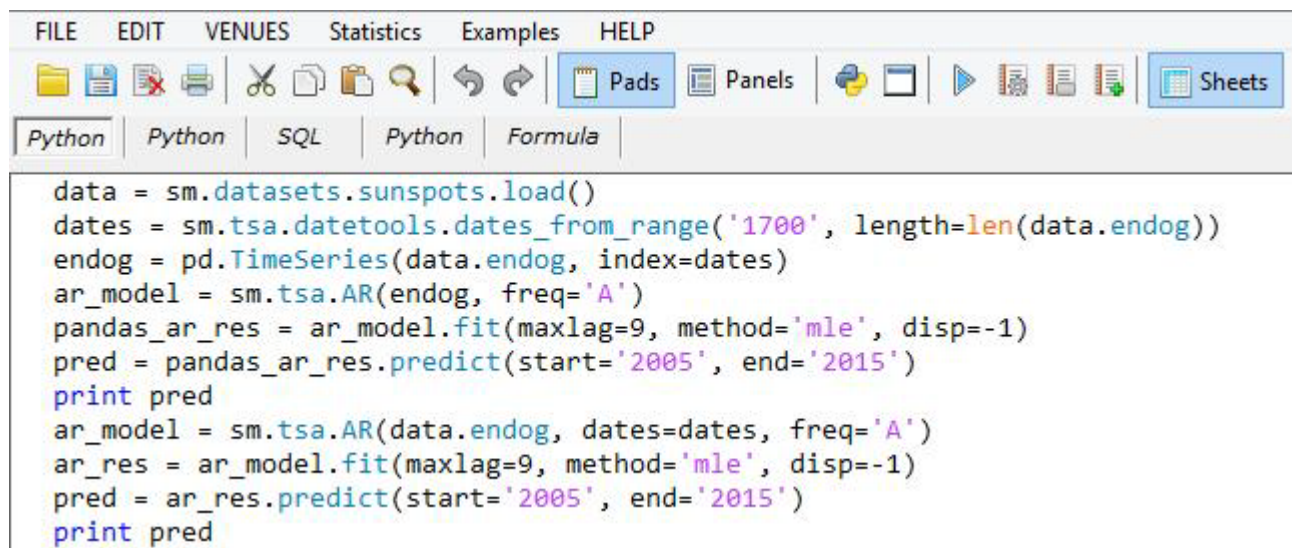
```
vr.inset(saveSIO, 'C:/Users/sqx1/Documents/Visral 3.0/SIO/new2.sio;Temp',0)
```

PAD Editors

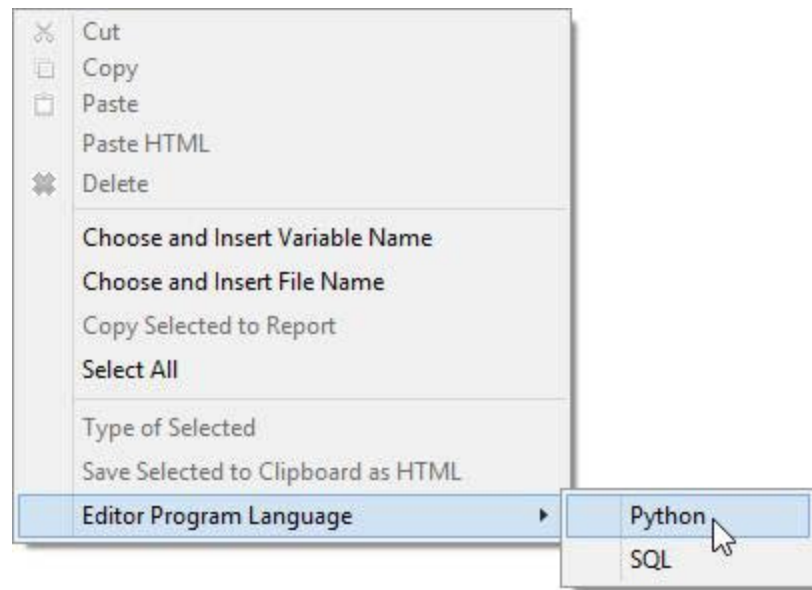
Covers:

- ✓ Basic PAD Features
- ✓ Executing Python Code
- ✓ Executing SQL Code

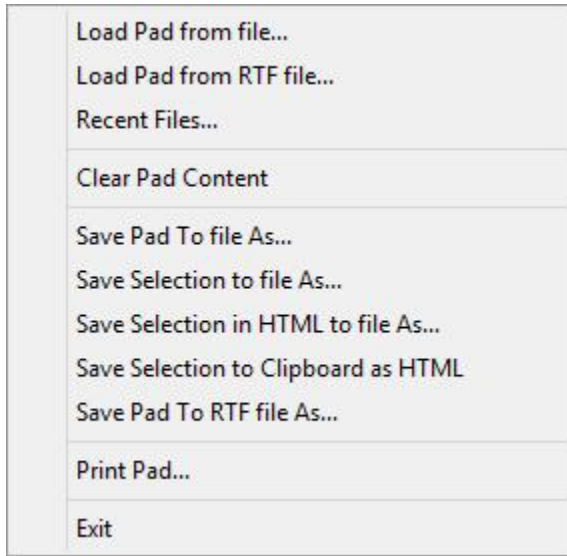
The PAD editors are selected by first selecting the **Pads** button on the toolbar and then choosing the particular editor of interest from the five tabs. Except for the Formula tab, the name on each tab indicates the language it is configured for.







The editors can be configured to process or communicate with different language systems or networks. This includes setting syntax coloring, indentation, auto completion, and execution. This is accomplished through the context menu. The image to the right illustrates configuring the current PAD editor to be Python.



FILE menu for PAD editors.

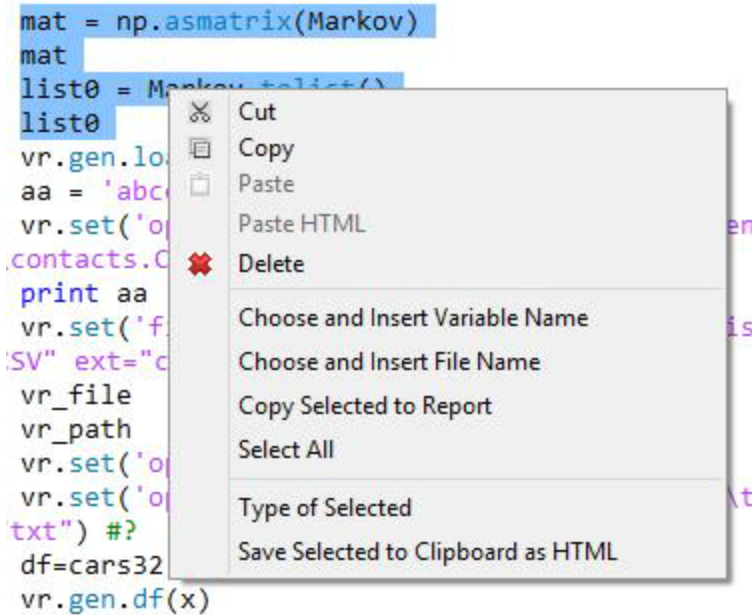


Selected code in the editor can be instructed to execute by clicking on one of the following buttons. (The editors must have focus for these particular buttons to appear.) The program language assigned to the edit dictates which bridge and its associated engine or network service will receive the selected code.

Button	Morphing Menu Description
	Left clicking on the Report icon with the plus symbol will execute the currently selected code and send results to the Report editor.
	Left clicking on the Report icon with the button symbol will embed the selected code in the Report editor as a button. When the button is clicked in the Guide a Panel entry with the code will be created. (See document section for details.)
	Left clicking on the Report icon with the gear symbol will embed the selected code in the Report editor as a button. When the button is clicked in the Report of Guide the code will be executed. (See document section for details.)
	Left clicking on the blue triangle icon will execute the currently selected code and send the results to the Trace editor.

Left click in the PAD editors will produce the following popup.





- The **Paste HTML** allows pasting all of the markup information along with the text copied from a page displayed on a browser.
- The **Copy Selected to Report** is a shortcut, replacing copy here and paste in report.
- The **Type of Selected** returns the type designation of selected word or method sequence.
- **Save Selected to Clipboard as HTML** is provided to allow colored copies of code to be used in the creation of web pages.

Python code can be copied from web pages and PDF files and pasted into the PAD editors. If Visral detects the three right angle brackets ">>>", the "In [num]", or the other artifacts of various Python interfaces, it will attempt to remove them before pasting. However, it cannot correct for such things as missing leading spaces within routines or conditional statements, or a wrapped instruction that shows up as multiple lines in the original text.

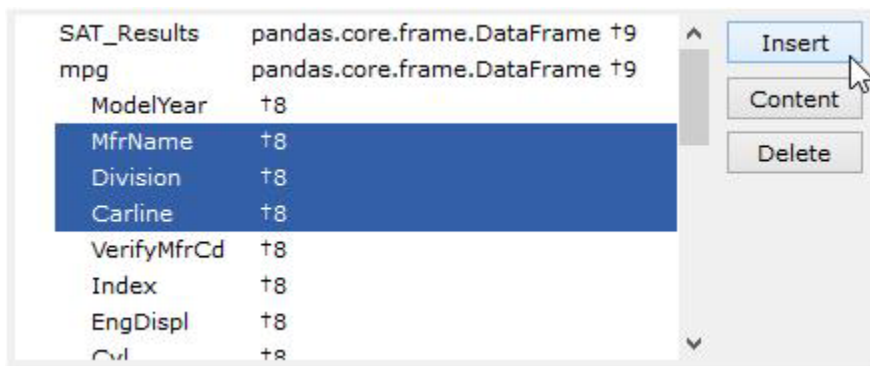
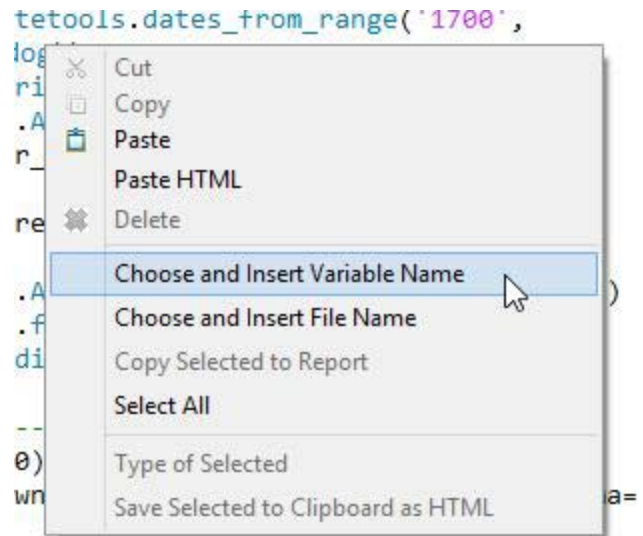
Wrapped instructions can be recognized in the PAD editors by the fact that subsequent lines will indent to the left (≪ that way), as illustrated below.

```
plt.text(0.5 * (a + b), 1, r"$\int_a^b f(x)\mathrm{d}x$",
horizontalalignment='center', fontsize=20)
```

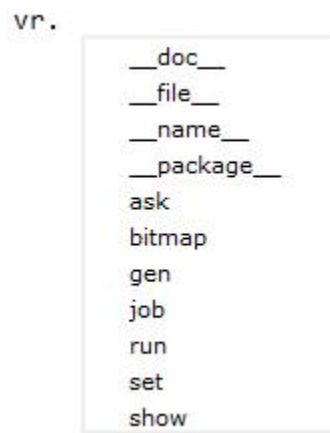
Single or multiple variable names may be selected for inserting into editor text by right clicking and selecting the **Choose and Insert Variable Name** entry of the context menu.

For certain variable types, the value can be chosen to be inserted as opposed to the name. Use the Content button to insert a value. (The dagger followed by a number is a type designation used internally by Visral.)

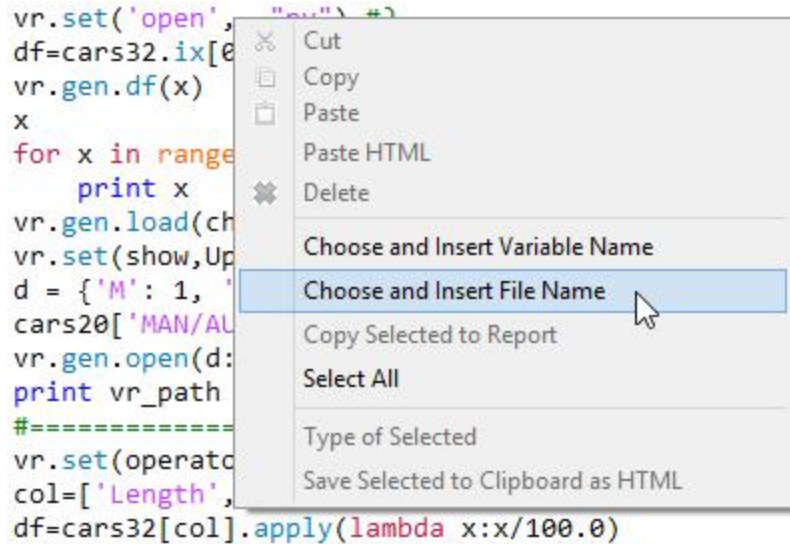
The point of insertion is where the cursor was when the right click occurred to bring up the context menu.



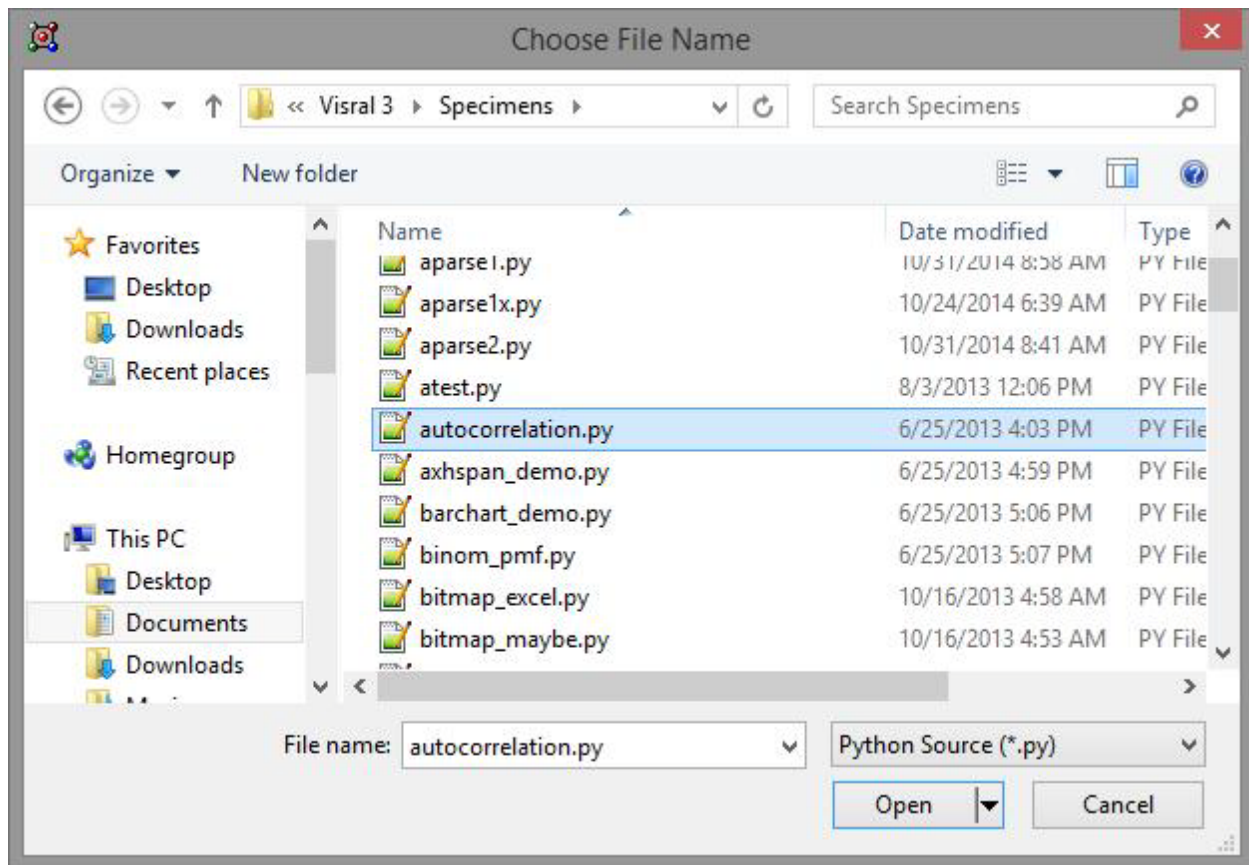
Of course, simply typing will display potential name choices as illustrated below. It can be seen that typing the additional period changes the choices which include variable names to strictly choices of methods.



There are also numerous convenience features. For example, file locations can be entered into static Python code through a windows files menu. This is accomplished by the **Choose and Insert File Name** in the right click context menu.



That will lead to the following menu appearing from which an entry can be chosen. The result is the full path name enclosed in single quotes will be entered into the editor at the point of the first left click.



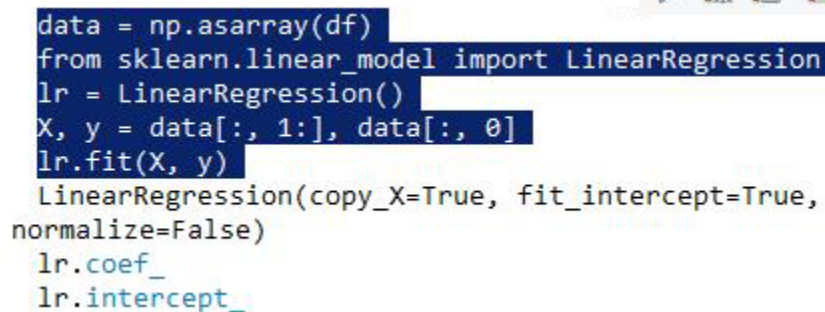
Python Code

Python runs natively on five of the Visral editors; the four Pad editors and the Formula editor. The results from the four PADs are directed to either the Trace or Report document editors; the Formula just to the Trace. The built-in Python functions "input" and "raw_input" are not applicable for this reason. Otherwise, the behavior is as would be expected from its normal interpreter self, executing the last line of code and sending the result to the Trace editor when its entry has been completed. (However, if a line of code is typed into the middle of existing code, it will not be executed. In other words, the following line must be blank or nonexistent for the preceding line to be executed.)

Unlike command line interpreters, Visral allows copying, pasting, and executing selected ranges of code. As viewed below, the selected five lines of code can be executed with their results being sent to either the Trace or Report. Left clicking on the blue (left) triangle icon on the toolbar will send the results to the Trace editor, the Report icon with the plus symbol will send it to the Report editor. (The function of the other two icons is described in the document section.)

All editors are Rich Text Format based. The four Pads and Formula are specially adapted for displaying Python source code with syntax coloring and auto indentation.

Indentation works in both directions. A carriage return at the end of a line will return the cursor back to the previous indent on the next line. Subsequent returns will step back indents without inserting another line until the left side of the page is reached.



```
data = np.asarray(df)
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
X, y = data[:, 1:], data[:, 0]
lr.fit(X, y)
LinearRegression(copy_X=True, fit_intercept=True,
normalize=False)
lr.coef_
lr.intercept_
```

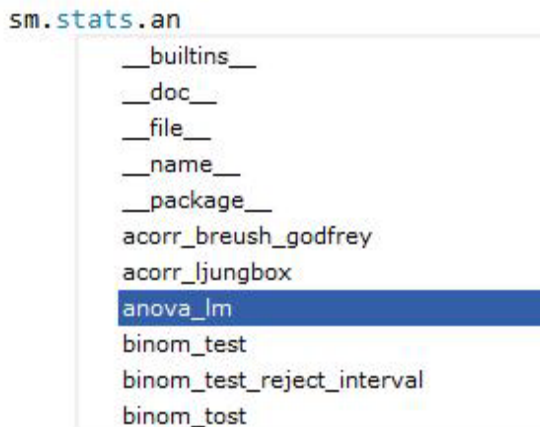
Python keywords

['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'not', 'or', 'pass', 'print', 'raise', 'return', 'try', 'while', 'with', 'yield']

Python Built-in Functions

abs, all, any, basestring, bin, bool, bytearray, callable, chr, classmethod, cmp, compile, complex, delattr, dict, dir, divmod, enumerate, eval, execfile, file, filter, float, format, frozenset, getattr, globals, hasattr, hash, help, hex, id, input, int, isinstance, issubclass, iter, len, list, locals, long, map, max, memoryview, min, next, object, oct, open, ord, pow, print, property, range, raw_input, reduce, reload, repr, reversed, round, set, setattr, slice, sorted, staticmethod, str, sum, super, tuple, type, unichr, unicode, vars, xrange, zip, __import__, apply, buffer, coerce, intern

As illustrated to the right, the editors also provide command preview and auto completion. A *tab* will complete the current word leaving the popup; at which point a *period* show the next set of options. A *return* will complete the word and remove the popup. Double clicking an expression in the popup will cause it to replace the last uncompleted word. A *down cursor key* will cause the focus to enter the popup. A left or right mouse click outside of the popup will cause it to close.



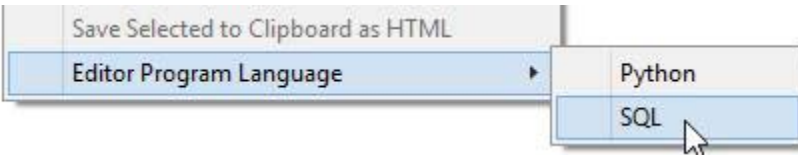
Visral communicates with Python through its standard interpreter interface by sending it code sequences. Python, or rather Python code, communicates with Visral by means of the following functions.

visral.pyd	Output	Description
vr.show()	Yes/no	Show Python images in Report or Trace. Visral converts plt.show() and fig.show() to vr.show(plt) and vr.show(fig) respectively. Pandas plotting functions need to be followed by the plt.show() in order to output a Python image to the Trace or Report.
vr.gen(...)	Yes/no	These instructions trigger Visral commands and prevent any further Python instructions from executing until they have completed the task. For example: <pre>vr.gen.open(C:\Users\sqx1\Documents\;py) print vr_path</pre> will open a files menu and wait for the user to select a file. When the menu closes, Python will proceed and execute the following print statement. (vr_path and vr_file contain the results from the open menu.)
vr.set(...)	no	These instructions trigger Visral operations that are not Python dependent.
vr.ask(...)	yes	Retrieve Visral parameters or settings.
vr.job(...)	yes	For use with bridges to issue Encrypt/Decrypt passages, code, commands and requests.
vr.run	yes	For use with Universal Bridge applications.
vr.bitmap	yes	For internal use only.

Detail on the above instructions can be found in the document section of this manual.

SQL Code

Visral utilizes the Python's embedded sqlite3 module to execute SQL commands. If the editor language is set to SQL in the context menu under the entry **Editor Program Language** then SQL text can be typed in directly and it will be both highlighted and executed correctly. (Visral performs the necessary wrapping to permit execution by Python.)



As illustrated below, Python code can also be included. They will be highlighted as usual but in italics and can be executable along with the SQL statements. (As with Python, sqlite3 keywords should not be used for variable names.)

```

connect
drop table if exists tbl
create table tbl (one varchar(10), two smallint);
insert into tbl values('SQXL',8);
insert into tbl values('Visral', 16);
commit
select * from tbl;
close
df=pd.read_table('http://www.cpc.ncep.noaa.gov/products/precip/CWLink/daily_ao_
index/monthly.ao.index.b50.current.ascii', '\s+',header=0,
names=['Year', 'Month', 'Value'])
vr.Load(df)
t = np.linspace(1,27,27).reshape(3,3,3)
pan = pd.Panel(t)
print pan
print pan.ix[0, :, :]

```

Editors set to SQL, load from and save to files with the '.sql' extension. This allows pure SQL statements to be saved and loaded. Python statements within those files appear as they are, but in italics for discrimination.

SQL code cannot be embedded into the Report as either an Operator or Execute button. However, they can be copied and pasted and the results of an execution can be directed to the Report as well as the Trace.

SQLite3 keywords:

connect, commit, close, select, rollback, create, insert, update, delete, drop, alter, table, into, from, values, where, order, set, count, if, exists, group, by, add, on, inner, join, column, having

SQLite References

<http://www.sqlite.org/>

<https://docs.python.org/2/library/sqlite3.html>


Formula Editor



Execution History

The history of executed code can be shown in the Formula editor by selecting History icon from the Morphing menu. It will display up to the last 30,000 characters sent to Python by the bridge from the code executed from both the PAD editors and Panel Operation. It does not record errors or results. However, it does display what was actually sent as can be seen below. This is the code processed by the SQL editor above. It illustrates the wrapping Visral performed on the SQL statements to have them processed by Python.

```
vr_conn = sqlite3.connect('test.db')
vr_cur = vr_conn.cursor()
vr_cur.execute('drop table if exists tbl')
vr_cur.execute('create table tbl (one varchar(10), two smallint)')
vr_cur.execute('insert into tbl values("SQXL",8)')
vr_cur.execute('insert into tbl values("Visral", 16)')
vr_conn.commit()
vr_cur.execute('select * from tbl')
for row in vr_cur: print (row)
vr_cur.close()
df=pd.read_table("http://www.cpc.ncep.noaa.gov/products/precip/CWlink
/daily_ao_index/monthly.ao.index.b50.current.ascii", "\s+", header=0,
names=["Year", "Month", "Value"])
vr.load('df')
t = np.linspace(1,27,27).reshape(3,3,3)
pan = pd.Panel(t)
print pan
print pan.ix[0, :, :]
```

Code in the Execution History listing can be re-executed by selecting that portion of interest and clicking the  icon from the Morphing menu.

Notes:

- ✓ The display of the Execution History does not automatically update and needs to be refreshed by clicking the History button causing the latest copy to be fetched from the currently active bridge.
- ✓ The code displayed is from the currently default Python. To see executed code of other Python installations the default must be changed.
- ✓ The history of any code executed from encrypted sources is not recorded.

Python Import Aliases

The following is a list of aliases assigned to imported modules in the Visral's Python startup files located in the *Program Files (x86)/Visral 3.0/PrimePy* directory.


Alias	Module Component
np	numpy
sp	scipy
mpl	matplotlib
mpimg	matplotlib.image
plt	matplotlib.pyplot
cm	matplotlib.cm
pl	matplotlib.pylab
dt	datetime
pd	pandas
web	pandas.io.data
sm	Statsmodels.api
smf	statsmodels.formula.api
sns	seaborn
vr	visral (assigned internally, not in the pyprime file)

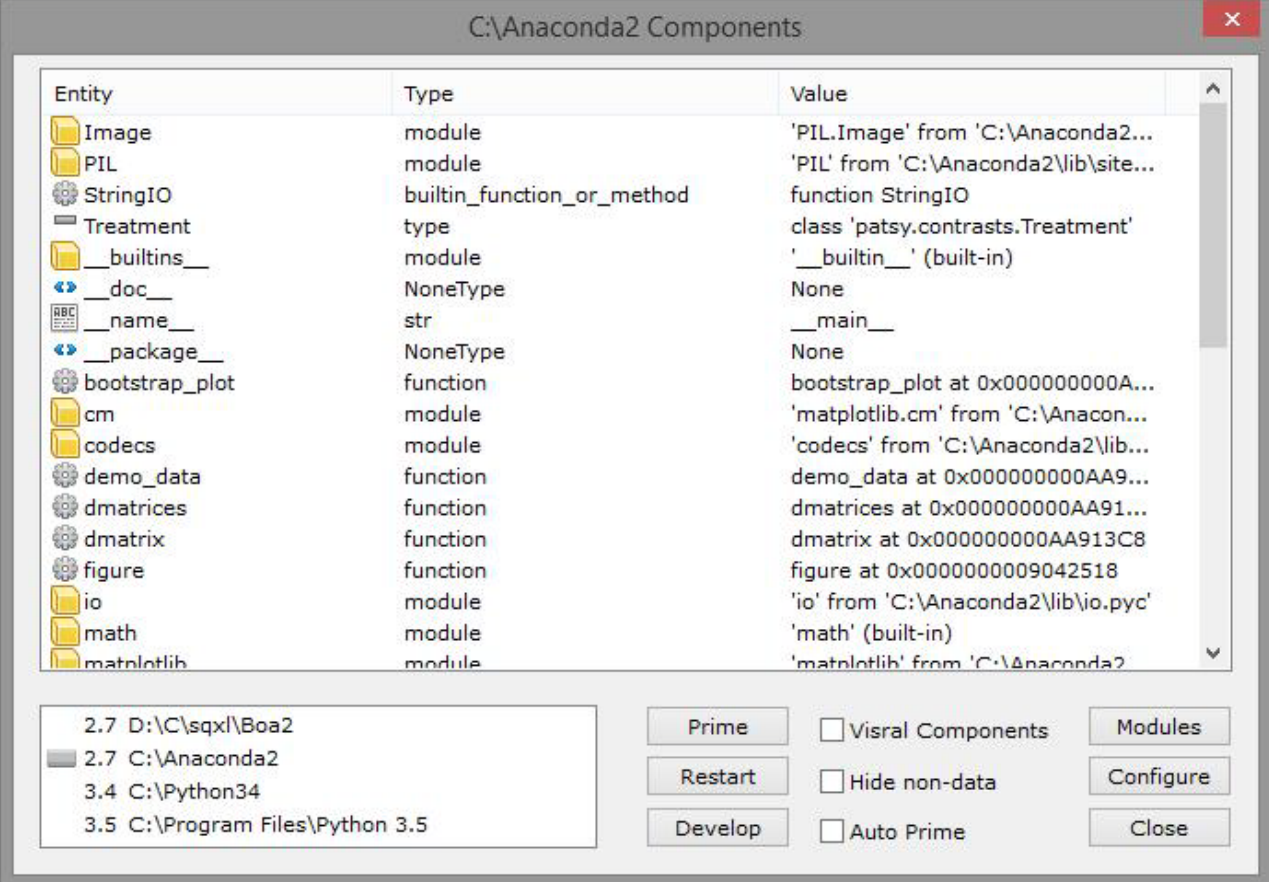
These are pretty standard aliases and code supplied with Visral assumes their assignments exist. It is possible some program examples may be found on the Internet that do not employ these, in which case it will be necessary to update their codes before running them.

It is possible to use different assignment by changing the default Python. Of course, Visral supplied code that depended on the above assignment may not function properly.

Python Components

 The **Orange Cone** now replaces the Python symbol on the main toolbar. Clicking on it will cause the morphing toolbar to display four symbols, the first of which is the Python. Selecting it will bring up the Python components menu shown below.

 This menu shows available variables (pointers to content) and their type along with the first part of the content value. Also displayed are code created functions and loaded modules. Clicking on the column headings will cause alphabetical sorting of rows.



Entity	Type	Value
Image	module	'PIL.Image' from 'C:\Anaconda2...
PIL	module	'PIL' from 'C:\Anaconda2\lib\site...
StringIO	builtin_function_or_method	function StringIO
Treatment	type	class 'patsy.contrasts.Treatment'
__builtins__	module	'__builtin__' (built-in)
__doc__	NoneType	None
__name__	str	__main__
__package__	NoneType	None
bootstrap_plot	function	bootstrap_plot at 0x000000000A...
cm	module	'matplotlib.cm' from 'C:\Anacon...
codecs	module	'codecs' from 'C:\Anaconda2\lib...
demo_data	function	demo_data at 0x000000000AA9...
dmatrices	function	dmatrices at 0x000000000AA913...
dmatrix	function	dmatrix at 0x000000000AA913C8
figure	function	figure at 0x0000000009042518
io	module	'io' from 'C:\Anaconda2\lib\io.pyc'
math	module	'math' (built-in)
matplotlib	module	'matplotlib' from 'C:\Anaconda2...

2.7 D:\C\sqx\Boa2
 2.7 C:\Anaconda2
 3.4 C:\Python34
 3.5 C:\Program Files\Python 3.5


Prime Restart Develop
 Visral Components
 Hide non-data
 Auto Prime
 Modules
 Configure
 Close

- ✓ Double clicking on an entry will cause its value (or type description) to be printed in the Trace.
- ✓ All available Python installations are displayed in the lower list box. Double clicking on its entries will change the default Python being used by all operations. The top entry is always the embedded Python, which is located the Boa2 directory.
- ✓ The Prime button will configure Python installations to support Visral operations. The following modules are required for this to be effective: matplotlib, numpy, pandas, patsy, Pillow, ~~pyarsing~~, ~~python-dateutil~~, pytz, pywin32, scipy, seaborn, ~~six~~, statsmodels, and xlrd. (The embedded Python includes these and is automatically primed.)

- ✓
- ✓ The Restart button allows Python to be re-started should it for some reason become hung. (This type of behavior might occur if Python code formed an infinite loop.) The Prime button may need to be selected and any previous Python data that might have existed will be lost. Dataset sheets will remain as they are not directly linked to Python but will require executing the [Generate DataFrame from Spreadsheet](#) Operator to recover its DataFrame image.
- ✓ The Develop button will allow the re-start of Python but in this case the bridge program will be visible. It will display all of the transactions between Python and Visral. The Prime button may need to be selected and any previous Python data that might have existed will be lost. Dataset sheets will remain as they are not directly linked to Python but will require executing the [Generate DataFrame from Spreadsheet](#) Operator to recover its DataFrame image.
- ✓ The Modules button will list the installed modules of the current Python.
- ✓ The Visral Components check box will enable displaying Visral specific Python components.
- ✓ Hide non-data check box will prevent the display of function, module, and other non-data entities.
- ✓ The Auto Prime check box and the Configure button are used to automate the activation of the non-embedded Pythons.
- ✓ The Refresh entry in the context menu will update the listing with any new entities that may have been created since the menu was displayed.

Python is embedded. It is not required to install Python or any modules.

Optional: Using other versions of Python

Besides the embedded Python, Visral bridges to all copies of Python 2.7, 3.4, and 3.5 that can be detected via the Registry. A list of detected Pythons can be found in the [Python Components](#) menu by clicking in the  button. Double clicking on the specific Python entry in the list will change it to be the default.

The modules in the following table are included as part of the embedded Python. However any separately installed Python where the Visral supplied solutions are intended to be used will need to install them. The site <http://www.lfd.uci.edu/~gohlke/pythonlibs/> contains a large alphabetized collection of Python modules in “wheel format”. Using the 32 bit or 64 bit table below, locate and download the entries from the website that match entries in the table. Then drag the files from the download directory to Visral, which will then install them in the Python set as the default. See the paragraph above for setting the default.

The following is a table of modules for the 64 bit Python installation. The strikethrough items are included in the latest release of matplotlib and do not need to be installed separately. (Ignore the version number shown in [orange](#) as it is frequently updated.)

Module	Install for 64 bit only
matplotlib	matplotlib-1.4.3-cp27-none-win_amd64.whl
numpy	numpy-1.9.2+mkl-cp27-none-win_amd64.whl
pandas	pandas-0.16.0-cp27-none-win_amd64.whl
patsy	patsy-0.3.0-py2.py3-none-any.whl
Pillow	Pillow-2.8.1-cp27-none-win_amd64.whl
pyparsing	pyparsing-2.0.3-py2-none-any.whl
Python-date	python_dateutil-2.4.0-py2.py3-none-any.whl
pytz	pytz-2015.2-py2.py3-none-any.whl
pywin32	pywin32-219-cp27-none-win_amd64.whl
scipy	scipy-0.15.1-cp27-none-win_amd64.whl
Seaborn*	seaborn-0.5.1-py2.py3-none-any.whl
six	six-1.9.0-py2.py3-none-any.whl
statsmodels	statsmodels-0.6.1-cp27-none-win_amd64.whl
Xlrd*	xlrd-0.9.3-py2.py3-none-any.whl

NOT REQUIRED

*These modules are listed under Miscellaneous, toward the end of the page.

The following is a table of modules for the 32 bit Python installation.

Module	Install for 32 bit only
matplotlib	matplotlib-1.4.3-cp27-none-win32.whl
numpy	numpy-1.9.2+mkl-cp27-none-win32.whl
pandas	pandas-0.16.0-cp27-none-win32.whl
patsy	patsy-0.3.0-py2.py3-none-any.whl
Pillow	Pillow-2.8.1-cp27-none-win32.whl
pyparsing	pyparsing-2.0.3-py2-none-any.whl
Python-date	python_dateutil-2.4.0-py2.py3-none-any.whl
pytz	pytz-2015.2-py2.py3-none-any.whl
pywin32	pywin32-219-cp27-none-win32.whl
scipy	scipy-0.15.1-cp27-none-win32.whl
Seaborn*	seaborn-0.5.1-py2.py3-none-any.whl
six	six-1.9.0-py2.py3-none-any.whl
statsmodels	statsmodels-0.6.1-cp27-none-win32.whl
Xlrd*	xlrd-0.9.3-py2.py3-none-any.whl

NOT REQUIRED

*These modules are listed under Miscellaneous, toward the end of the page.

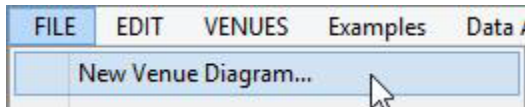
Visral Diagrams – Venues/Panels/Operators

Covers diagrams:

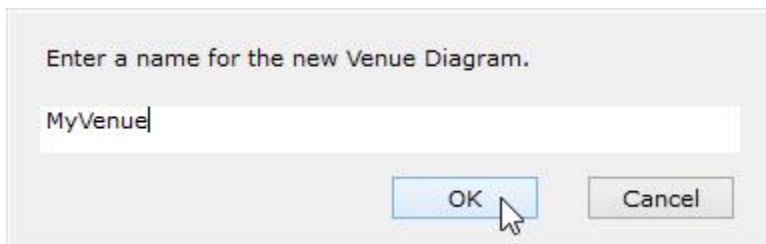
- ✓ Quick Start Tutorial
- ✓ Venue Construction Diagrams

Creating Applications

Creating a new Venue by first selecting the Diagram window and then selecting the **New Venue Diagram** entry from the main menu FILE.



This will cause a popup to appear where the name of the Venue may be entered. A file whose name is that of the Venue concatenated with ".vn.xml" will be created when the diagram is finally saved.



Clicking OK will cause a new diagram to be displayed with two elements in it.



The first (left) element holds the name that will appear in the main menu between the VENUE and HELP entries when this Venue is loaded. The second element holds the name that will appear as an entry in the dropdown associated the main menu entry and show up as a tab in the Panel window if it selected.


Left click on background to KEEP changes. Right click to CANCEL changes.




Clicking on the body of each element put them in edit mode and allows their content to be changed. Left clicking on the background of the diagram will cause edit mode to be exited and any changes to be saved to the diagram. Right clicking on the background will cancel any changes. (These changes are NOT save to the source file until the Floppy icon on the toolbar is clicked.)



Now it is time to add an Operator. Operator elements contain the primary programs and commands.

A monetary left click on the Operator element in the menu to the right will cause the cursor to change shape and appear as a mini element. 

When the black bar of the cursor touches an edge of an existing  element, its appearance will change; now in red with the word **Attach**. A left click will cause the new element to become attached to that edge. The first Operator must be connected to the backend of the target element.



Left clicking on the body of the element will put it in edit mode where its name can be changed. Left clicking on the icon will cause the *EDITOR* Panel to be displayed where both the name and a program can be entered.



The *EDITOR* Panel will appear as below with five specific title fields which are used to represent the various components of an element. (See construction diagrams for more detail.)

Now code can be entered into the program field, which then will be executed when the Operator is displayed and clicked in the Panel window. (The code below can be captured from the PDF file and pasted into the PROGRAM field.)

Left clicking on the **Set Element** Operator will cause edit mode to be exited. Now the Venue can be saved and/or emulated.



Element Components

Set Element

Identity

My Operator

Reference

Text entered here is what is displayed along side of the icon. It is generally used to describe the element's operation, a URL to further explanation, or the original author and source of any associated code.

Presets

This text is used to describe immediate content.

Program

```
frozenRV = sp.stats.norm(5,3)
x = np.linspace(-5, 15, 101)
y = frozenRV.pdf(x)
plt.close()
plt.plot(x, y, label = 'norm pdf')
plt.title('Distribution')
plt.xlabel('X')
plt.ylabel('PofX')
plt.legend()
plt.show()
```

Payload


Auto load


- Set this element as read only
- Collapse child elements
- Gray, hide this element
- Optional, selection not required

Element Parameters *readonly*


Routing Parameters *readonly*

2D Bidirectional Link-list *readonly*

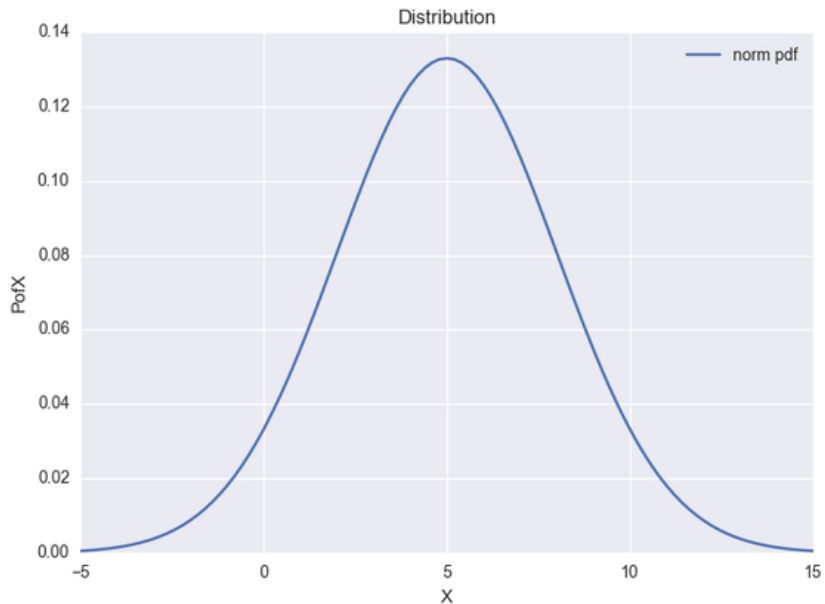
To save the file click the floppy icon on the toolbar . A directory menu will show itself already set to the default directory and file name. All that is necessary is to click the save button, and next time Visral is run the new Venue will be installed under the VENUE main menu entry.

It is not required to save the Venue to try it. Clicking on the Simulate Panel  icon in the morphing toolbar will cause the following Panel to appear.

In the current example the Panel will show the following single Operator. Clicking it will result in the Trace displaying the subsequent plot including the six lines of text. Clicking the top button to the left will send it to the Report, but with the six lines removed by Visral.



My Operator [`<matplotlib.lines.Line2D object at 0x0000000014F46390>`
`<matplotlib.text.Text object at 0x0000000014EEE780>`
`<matplotlib.text.Text object at 0x0000000014D8AB38>`
`<matplotlib.text.Text object at 0x0000000014EDFA90>`
`<matplotlib.legend.Legend object at 0x0000000014F46AC8>`
`<matplotlib.legend.Legend object at 0x0000000014F46AC8>`]

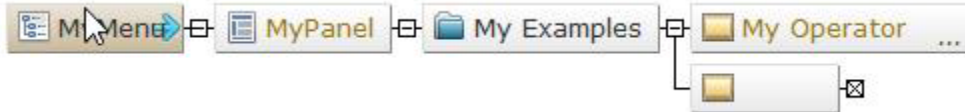


Now for Inputs

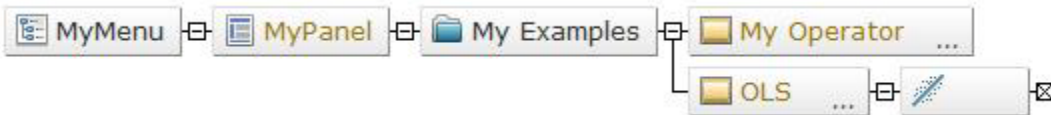
Start by adding another Operator. This time the Operator must be attached to the lower (or upper) edge of the existing Operator resulting in the following appearance.



Make it a little more interesting by inserting a Folder element. Attach it to the backend of the Panel element and give it a name.




Name the new Operator OLS and enter the following code into its Program field. (Save time; copy and paste from this PDF file.) Then add a Patsy element to the right of the Operator.



```
mod = sm.OLS.from_formula($p, $pd)
results = mod.fit()
results.summary()
results.params
results.tvalues
```

The Patsy element expects a Patsy formula to be entered, such as: "Z ~ X + Y" (with or without quotes). See <http://patsy.readthedocs.org/en/latest/formulas.html> for more details. It is assumed the DataFrame source is identified by the name of the spreadsheet that has focus. To select a different source enter @otherdf prior to the expression. The placeholders \$p and \$pd will receive the formula expression and the DataFrame name respectively.

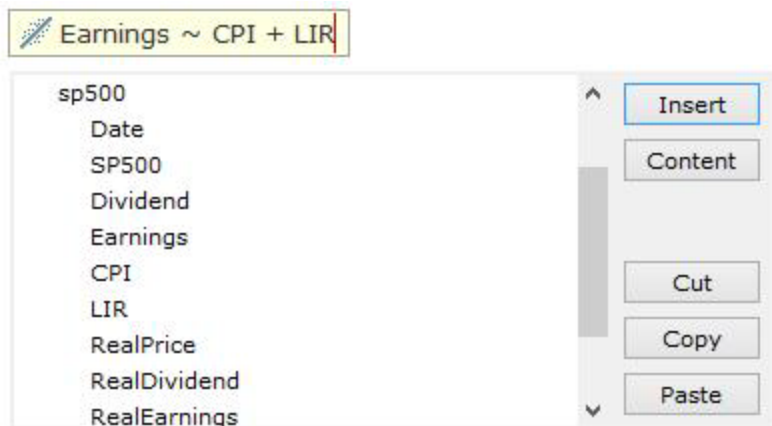
Save the file again .

Now click on the Simulate Panel  icon in the morphing toolbar, which will cause the following Panel to appear.



Next step is to enter a Patsy formula composed of columns from the spreadsheet that currently has focus and then execute the Operator. (The following example is to illustrate procedure and not necessarily meant to be of statistical significances.)

Enter the edit mode of the Patsy element and right click on it. The following popup will appear from which the columns can be selected and Patsy operators added. Clicking outside the popup will close it and exit the edit mode.



If the formula expression is typed in by hand, it will need to be preceded by @sp500 to specify the source of the columns.



Left clicking on the OLS Operator will lead to the following printouts.

```
<class 'statsmodels.iolib.summary.Summary' >
"""
                                OLS Regression Results
=====
Dep. Variable:                    Earnings    R-squared:                    0.452
Model:                            OLS        Adj. R-squared:                0.451
Method:                            Least Squares    F-statistic:                    709.2
Date:                            Mon, 18 May 2015    Prob (F-statistic):            2.56e-225
Time:                             15:33:18        Log-Likelihood:                 -18917.
No. Observations:                  1722          AIC:                           3.784e+04
Df Residuals:                      1719          BIC:                           3.786e+04
Df Model:                           2
Covariance Type:                   nonrobust
=====
               coef      std err          t      P>|t|      [95.0% Conf. Int.]
-----
Intercept    193.4643    658.572     0.294     0.769    -1098.223  1485.152
CPI           0.2822     0.008    37.439     0.000     0.267  0.297
LIR          1.2152     0.140     8.653     0.000     0.940  1.491
=====
Omnibus:                        799.851    Durbin-Watson:                 0.456
Prob(Omnibus):                   0.000    Jarque-Bera (JB):              8166.550
Skew:                            1.914    Prob(JB):                      0.00
Kurtosis:                       12.958    Cond. No.                      9.34e+04
=====

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 9.34e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
"""

Intercept    193.464290
CPI           0.282212
LIR           1.215161
dtype: float64

Intercept    0.293763
CPI          37.438604
LIR           8.653136
dtype: float64
```

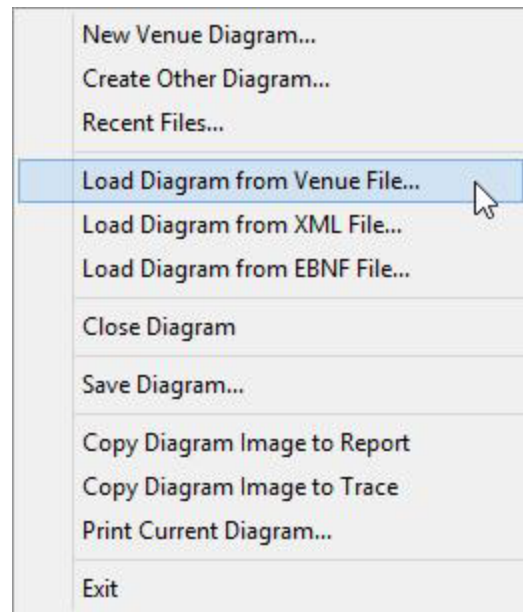
Venue Construction Diagrams

To load an existing Venue construction file, click on the FILE entry of the main menu, and then click on the [Load Diagram from Venue File](#).

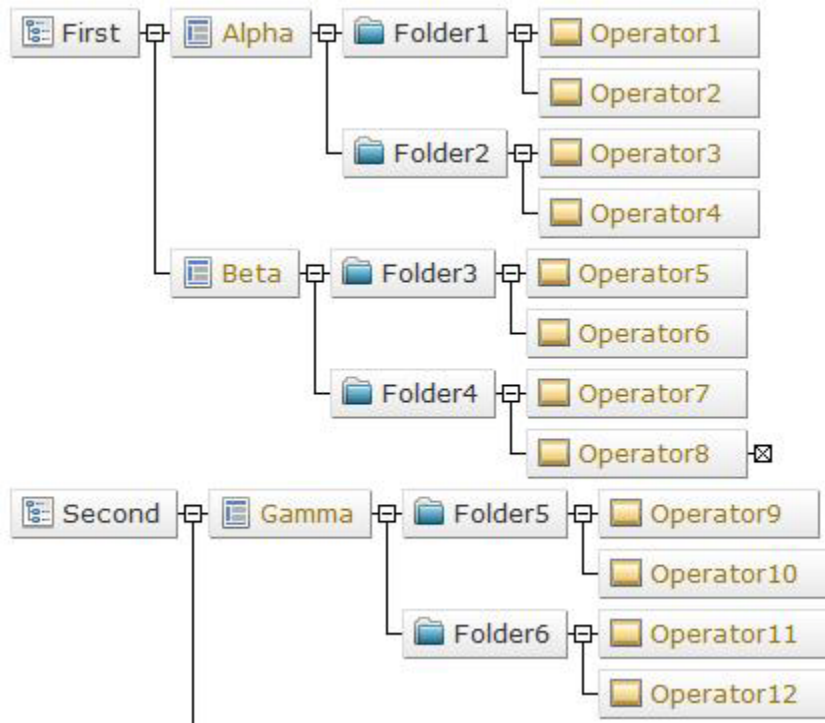
Access: Users\...\Documents\Visral 3\Venues

(To access system Venues, depress the control key when selecting [Load Diagram from Venue File](#).)

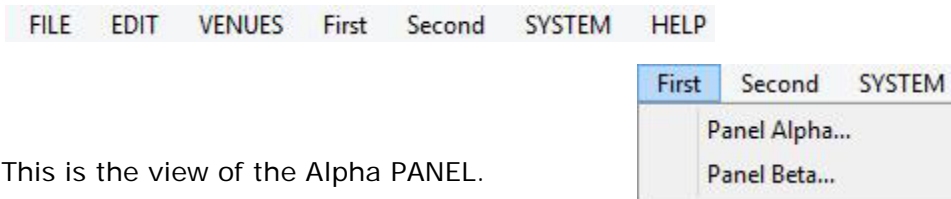
Access: Program Files (x86)\Visral...\Venues)



This diagram illustrates how the first four levels of elements are interpreted.



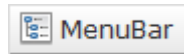
If this Venue were loaded, it will be listed as an entry under the main menu's VENUE. If selected, the **MenuBar** element becomes an entry in Visral's main menu between VENUE and HELP. The **Panel** elements then become sub-menu entries and the **Folder** element becomes a collapsible grouping of solutions. In other words, the above diagram would result in the following:



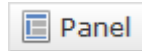
This is the view of the Alpha PANEL.



Panel Elements



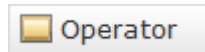
The name of this element becomes a main menu entry if the particular Venue is installed and loaded.



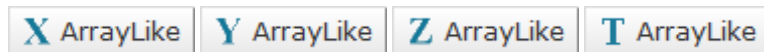
This becomes a submenu entry to the parent Menu Bar. Required as a child of the MenuBar element.



Used for grouping Operators within a folder.

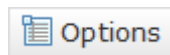


The Operator: This where the executable code of the function is located. If there are child elements involved the icon will appear with a mini folder in the right lower corner when displayed in a Panel. Folder icons within a Panel have the ability to expand and collapse by clicking them, exposing or hiding child features.

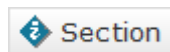


These elements expect a DataFrame name or names of one or more DataFrame columns.

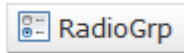
The content of this element will automatically be substituted for placeholder \$X, \$Y, \$Z, and \$T respectively.



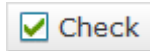
This element is used to provide optional choices by means of a popup. The content is expected to have the form: $\$A \text{ Name}=['option0', 'option1', \dots]$ where \$A represents the intended placeholder, optionN represents the choices, and Name is the attribute to which to assign the choice. To perform a straight substitution without an assignment use $\$A ['option0', 'option1', \dots]$.



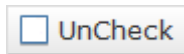
Used for grouping Operators within a heading.



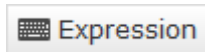
This is for grouping Check and Uncheck into a Radio button configuration. Use \$A Name= or just \$A.



\$A Name=(True appended)
If part of Radio Group then Response assigned to Radio name.



\$A Name=(False appended)
If part of Radio Group then no contribution to Radio name.



Used for entering ASCII data.
\$A Name=(entries appended)

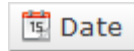


Used for entering ASCII data that may be separated by commas. The parentheses are automatically added when content is substituted for the placeholder.



Similar to the Data In element except indicates the input is expected to be a Patsy formula expression. It accepts column names to form expressions, such as: "Z ~ X + Y". See <http://patsy.readthedocs.org/en/latest/formulas.html> for more details. It is assumed the DataFrame source is identified by the name of the spreadsheet that has focus. To select a different source enter @otherdf prior the expression.

Unique placeholders are assigned to Patsy entries. \$p will receive the formula expression and \$pd will receive the DataFrame name.

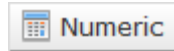


This element recognizes the following formats of date and time, converting them to the Python `dt.datetime()` format.

mm/dd/yyyy
Month dd, yyyy
mm/dd/yy
yyyy-mm-dd
yyyy/mm/dd
dd-Mon-yy
mm.dd.yyyy
Mon. dd, yy
dd Month yyyy
Month yy
Mon-yy
mm/dd/yyyy hh:mm AM
Mon-yy
mm/dd/yyyy hh:mm:ss AM
hh:mm:ss AM
hh:mm AM
hh:mm:ss

dd : one or two digit day of the month
mm : one or two digit month of the year
yy : two lower digits of year 20XX
yyyy : four digits of year
Month : full name of month name
Mon : abbreviated month name
hh : one or two digit hour of the day
mm : one or two digit minute of the hour
ss : one or two digit second of the minute
AM : AM or PM

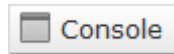
All entries are case insensitive. Required number of month characters, only enough to differentiate.



Used to enter numeral expressions.



Used to specify the destination the results should be stored in.



Used in combination with image to create a popup panel. The console is where an Operator and any support elements are connected.



Used to create an image button, which when clicked will pull up a console.







Content of this element will be displayed as information in the Panel.

Diagram Superscript Descriptions

U	URL or other reference
R	Recursive
M	Multiple rule elements have this name
I	Interpret as case insensitive
H	Interpret as hex representation. In the case of an Atom/Non-terminal element indicates the hexadecimal value should represent the contents and not the lateral. In the case of a Repeat element in a one-or-more-times configuration, the back path has priority if in conflict with the forward path.
L	In the case of a Repeat element in a one-or-more-times configuration, the forward path has priority if in conflict with the back path.
F	Indicates a function name or formula was entered.
T	Indicates a trigger was set. Only affective on elements with assigned states.
\$	Only present following a build. Indicates that this element has the possibility of generating an output term in intermediate postfix results.

Auxiliary Operators

The following icons may be found to the right of some Operators.

	This provides access to information by sending the default browser to a previously assigned web page. (Does not cause the Operator to execute.)
	Clicking (toggle) this will instruct the Operator to produce a plotted output when executed.
	Clicking this will cause an example to be executed illustrating the functions of the Operator. vr.ask(vrlist) will respond with True if set; else it will respond with False.
	The toggle icon whose function varies with individual Operators. vr.ask(vrflip) will respond with True if blue arrow points up to the right; else it will respond with False. (Does not cause the Operator to execute.)

The following are used with elements to assign arguments to placeholders. They are case insensitive.

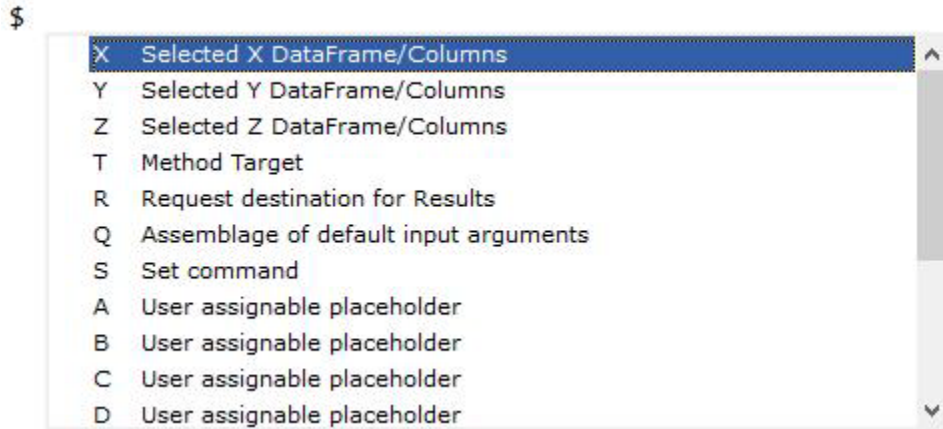
Placeholder	Description
\$X	selected X columns
\$Y	selected Y columns
\$Z	selected Z columns
\$T	Target function results
\$A-\$J	Unassigned user placeholders

Placeholders

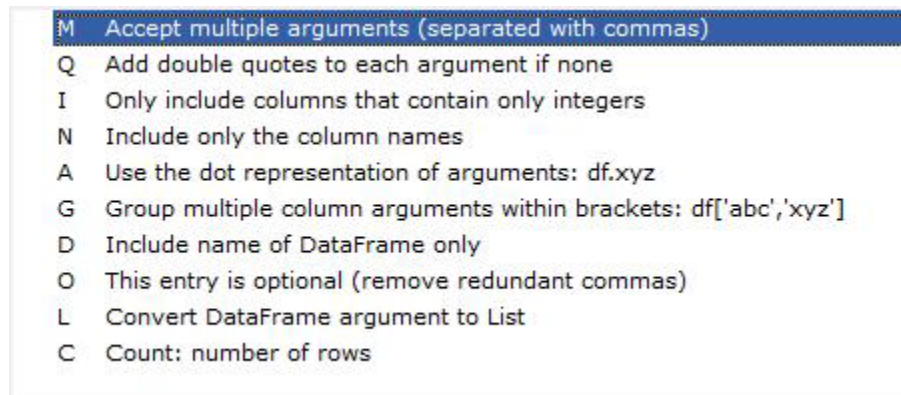
Placeholders are similar to the arguments of a function in that it is not necessary to know their actual value until the program is running. In Visral they are indicated by a dollar sign followed by one or more letters.

\$X, \$Y, and \$Z are replaced by the columns selected in the current spreadsheet or the content of the reference type. \$A through \$J are user assignable.

Typing a dollar sign (\$) while in a Pad or Formula editor will bring up the following menu of selections. A double click on the appropriate entry or simply typing the letter will enter the letter and then advance to a second menu of options.



The \$P is reserved for input from the Patsy module. The \$S is reserved for the vr.set command, which can be processed without Python installed. All placeholders and modifiers are case insensitive. Certain modifiers are only applicable specific placeholders.

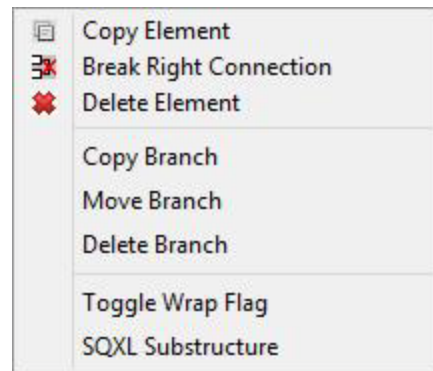






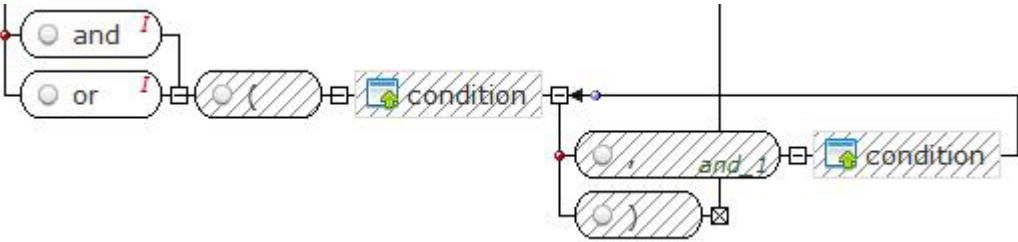
The S modifier following the M causes the arguments to be separated by semicolons.

The default form of replacement for \$X, \$Y, and \$Z is DataFrame['column'];

When executing an expression, the dollar sign (\$) is not interpreted as a placeholder if it is located inside single or double quotes, preceded by a backslash (escape) or followed by any character other than a letter or forward slash (/). This permits solutions that use the long form placeholders \${...} of the Python templates to be implemented in Visral without conflict. It should also not conflict with the regular expression's end of string match.

Right click on an operator element will bring up the following menu.




Entry	Description
Copy Element	<p>If this entry is clicked the cursor will take the shape of an empty element.  When the black bar of the cursor touches the edge of the target element, its appearance will change.  In this way the highlighted element can then be copied to a new location. The element being copied will have a hashed appearance until the activity is completed or canceled.</p> <p>This operation can be cancelled by right clicking on the background of the graphics window. The undo button will restore the original connections.</p>
Break Right Connection	<p>Disconnect the existing join connection on the right side of the element. It will not leave an element to the right floating; i.e. it will not perform the action.</p>
Delete Branch or Element	<p>Delete this element. If the element is the head of the rule, the complete rule will be deleted. The undo button will restore the element or branch. The undo button will restore the original connections.</p>
Copy or Move Branch	<p>If this entry is clicked the cursor will take the shape of an empty element.  When the black bar of the cursor touches the edge of the target element, its appearance will change.  In this way the highlighted path of elements can then be copied or moved to a new location. The branch being copied will have a hashed appearance until the activity is completed or canceled.</p>  <p>This operation can be cancelled by right clicking on the background of the graphics window. The undo button will restore the original connections.</p>

Visral Diagrams - Universal

Covers diagrams:

- ✓ Loading, Saving, and selecting files
- ✓ Moving about a diagram
- ✓ Selecting an Element Groups
- ✓ Inserting and Editing Elements
- ✓ Wiring 2-port Elements

Diagrams are brought into view by first selecting the  button; following with reading in a file or choosing a previously loaded or created by clicking its tab.

Description

Visral Diagrams use the tree structure of XML syntax in conjunction with specific XML semantics to represent hierarchical directed graphs. The structure supports production rule sets, computational models, and relationships between facts, entities, observables, and operators.

Not to be confused with plots and charts outputted from Python, Visral Diagrams are interactive graphics, for conveying information to applications.

The nature of their visual presentation depends upon the particular type of diagram being displayed and interacted with. The same is true as to whether their elements are 2-port or multiport, and their processing, which fall into two categories, formulation and compilation.

Formulation is where the diagram is interpreted and converted to a DataFrame or array for computation by Python or another language. In compilation, diagrams create active models or automaton, which are subsequently engaged by other operations or applications. This brings significantly more power to models as it allows the integration of complex and non-linear elements. (Refer to Automaton for more complete information.)

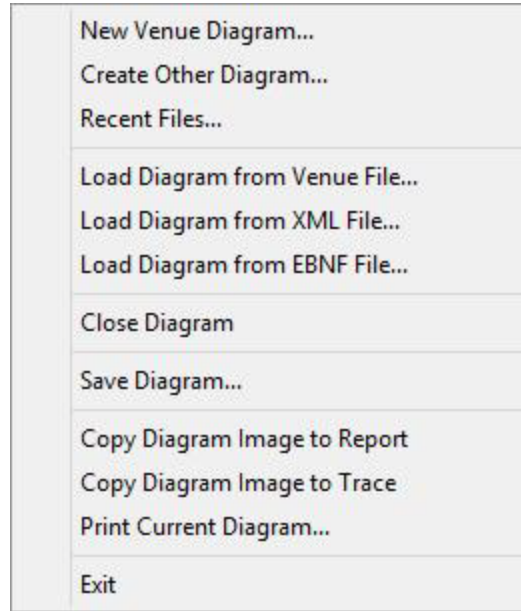
File Extensions	Description
*.xml	Visral diagrams (Others: markup edit only)
*.sqxl	Visral Spreadsheet
*.html, *.htm, php, ...	Markup edit only
*.bnf, *.ebnf	Extended Backus–Naur Form (ASCII)

Loading, saving, and selecting files

Diagrams can be loaded from files or created from scratch, possibly from portions of other diagrams or with the aid of diagram generating functions.

Selecting the DIAG button on the toolbar will bring up Tabs of previously loaded diagrams, as well as associate certain toolbar buttons and the FILE menu with diagram actions.

To create or load a file, click on the FILE entry of the main menu, which will in turn display a dropdown similar to that the right.



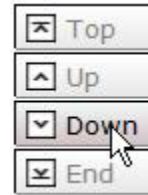
Default directory. Users\...\Documents\Visral 3\Specimens

Element Group	Convert	Compile	Interpret	Description
Panel			✓	Passive
Markov	✓	✓		
Petri Net	✓	✓		
Path Model	✓	✓		
Business	✓	✓		
Grammar		✓		
Internet			✓	Active
Markup				
Production			✓	Active
Stat101		✓		Active statistical models
SPICE		✓		
Mimic			✓	Passive
Depend		✓		
Map		✓		
Policy			✓	Passive

Moving about the Diagram

There are four means by which to move around diagrams:

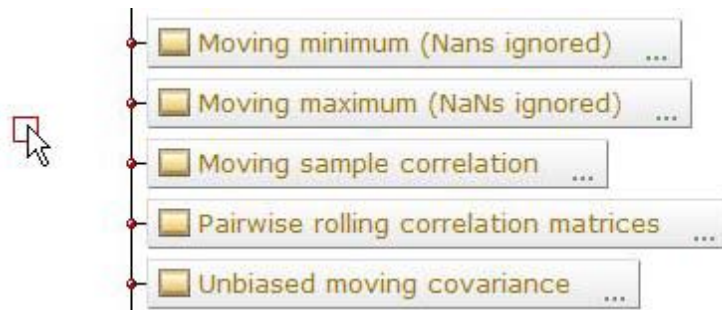
Double click on the background and a Top/Up/Down/End menu will appear under the cursor. Clicking the Up or Down will cause a page up or page down. The Top brings the view of the top left area of the graphics and the End will bring the view of the bottom left. As long as the cursor is on the menu it will remain visible and functional. Once the cursor moves off of the menu it will disappear.



Using the scroll wheel on the mouse will cause the view of the graphics to be scrolled up or down.

Pressing and holding the left button on the background while dragging it will cause the graphics to move about the window. This provides fine control over the two dimensional position of the view.

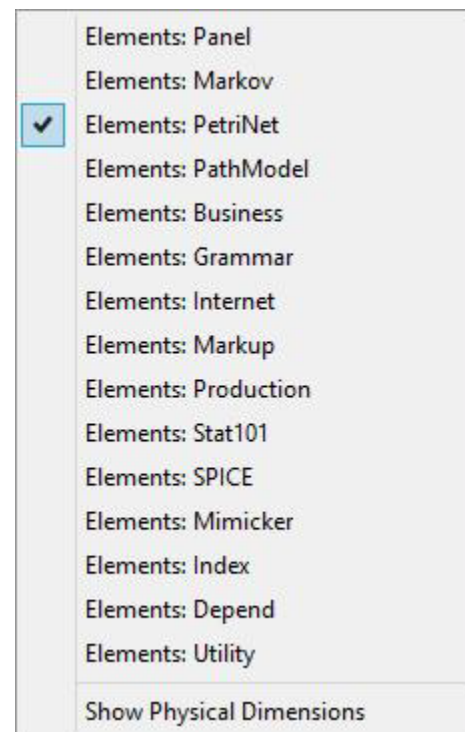
Pressing and holding the left button on the small floating red box while dragging it will cause the window to move about the graphics. When clicked the size of the box changes to represent the relative size of the view to the total graphics. The box replaces traditional scroll bars and is much more convenient for two dimensional movements.



Selecting an Element Group

The content of each diagram is assessed when loaded, setting its type and selecting the appropriate element list and available operations. (The selection is determined by the first rule head encountered.)

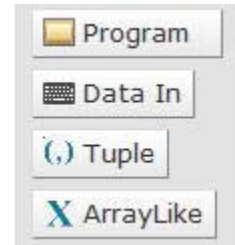
Right clicking on the background will bring up the element selection menu. This allows different element groups and their associated operations to be applied to any diagram.



Inserting Elements

The segments below reflect the result of inserting new elements as represented by entries 1-6. To insert an element left click on the desired one. (Do not drag it. Dragging on the backplane will cause the diagram to move.) After having clicked on an element in the panel, the cursor will change to the element symbol.

When the black bar of the cursor touches the edge of a target element to which it is to be attached, its appearance will change; now in red with the word **Attach**.



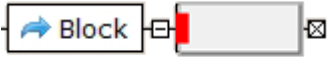
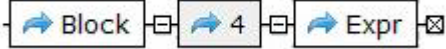

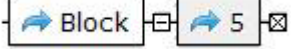
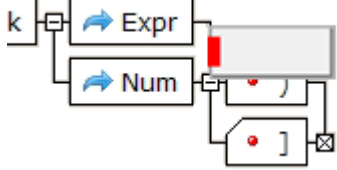
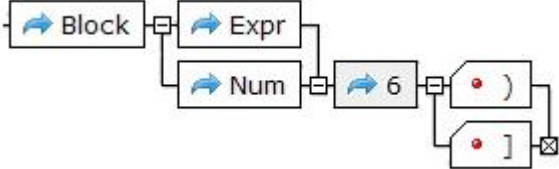
This means a left mouse button click will cause the new element to be attached to the top, left, right, or bottom of the target depending on the particular edge. The autorouter will then make the appropriate adjustments to the display.

When the cursor is over the center of an element the attachment point will turn blue with the word **Replace**.

This indicates a click here will cause the current element to be replaced by the new one.

To abort the action right click on the background and the cursor will revert to the arrow. A left click will not abort but will allow the diagram to be moved around if held down and dragged.

		<p>Before contacting an element edge or center. When the attachment point turns red or green, a left mouse click will cause it to attach or replace at that point.</p>
1		<p>Right contact leads to the element being attached to the right side. If there were already elements to the right, the new one would be inserted between.</p>
2		<p>Top contact leads to the element being connected to the left side and placed above. Any already above would be pushed up.</p>
3		<p>Bottom contact leads to the element being connected to the left side and placed below. Any already below would be pushed down.</p>



<p>4</p>		<p>Left contact leads to the element being inserted to the left side, between the current and previous elements.</p> 
<p>5</p>		<p>Center contact leads to the new element replacing the current element. Only the element type is changed. The content remains the same as before.</p> 
<p>6</p>		<p>Contacting a vertical connection will also cause the attachment point to turn red. If the mouse is clicked at this point the new element will be inserted after the join portion of the vertical and before any forks.</p> 

Note: Some Petri Net and Markov Chain elements will insert a pair that includes a directional element. See the description in Volume 2 for details.

Junctions

The junction boxes between or at the end of elements will contain one of four different symbols; a minus (\ominus), a plus (\oplus), an X (\boxtimes), and a blank (\square). If there is a minus and the box is click on, it will change to a plus and cause all of the following elements to disappear. Clicking on a plus will reverse the condition. The boxes with an X indicate a stop. Any element that is activated will continue to the next element unless the element's execution fails or a stop is encountered. Blank boxes as with execution failure cause backtracking to find an alternate solution. The junction red spheres represent either joins (alternate paths) or forks.

To avoid any confusion, the diagram's autorouter directs vertical lines behind elements or around any junction it is not meant to connect to. One case handled differently is with the rule end or X box (\boxtimes).

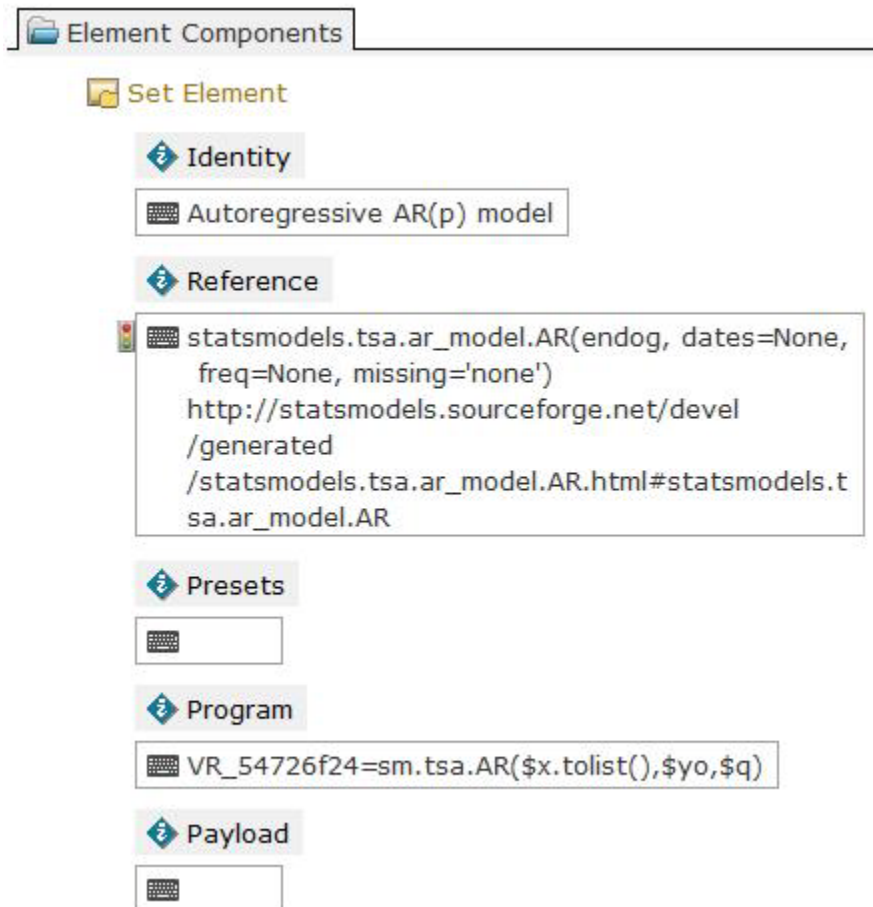
<p>Here the X box (\boxtimes) doubles as a join.</p>	
<p>In this case the vertical connection is simply passing behind the stop.</p>	

Editing Element Formulas

The cursor normally shows as an arrow when in the diagram window. However, when it touches the body of an element, it changes to a pointing hand. Left clicking on the body of an element will cause the element to enter the edit mode.

When the content is being edited the background of the element will become light yellow and a red vertical line will become visible representing the point of entry for new characters. A single left click will reposition the red insertion pointer.


Left clicking on the icon of an element will bring up the *EDITOR* Panel and will display and make available for editing the greater content of the element. The following is an example of its content for an Operator element.





Five titles separate the different fields of an element. The comments and their number may differ depending on diagram, element, and cell types. The following is what can be expected from clicking on an empty element.


 Element Components

 Set Element



 Identity


 The text following the Identity image is the same as that showing in the element in the diagram and can be edited both places. However, changes in the diagram will have precedence. (Changes to any other components will still be implemented.)

 Reference


 Text entered here is what is displayed along side of the icon. It is generally used to describe the element's operation, a URL to further explanation, or the original author and source of any associated code.

 Presets

  The text following the Preset image contains the values of the last run parameters. These values will be used as a last resort if no others are offered. The primary use is for when a solution is retrieved from a document by selecting a Folder or Operator button. It contains all of the information necessary to reproduce the same results as the original computation.

 Program

 This is where program code is entered.


 Payload


 Code of the optional example goes here.

 Auto load

- Set this element as read only
- Collapse child elements
- Gray, hide this element
- Optional, selection not required

 Element Parameters *readonly*

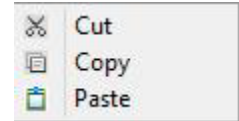
 Routing Parameters *readonly*

 2D Bidirectional Link-list *readonly*

Editing element text directly in the diagram

While the element background is light yellow (in edit mode), the text can be selected by holding the left button down while dragging the cursor. Individual words can be selected by double-clicking on the word. The entire expression from the beginning to the point of the cursor can be selected by double-clicking at the end of any word.

The selected text can be deleted with the backspace or delete key, or, cut, copied, or pasted over by right clicking to bring up the options menu. Depending on the element type, text editing may end with a carriage-return. To enter a carriage-return into those element types, simply hold down the shift key while typing the return.







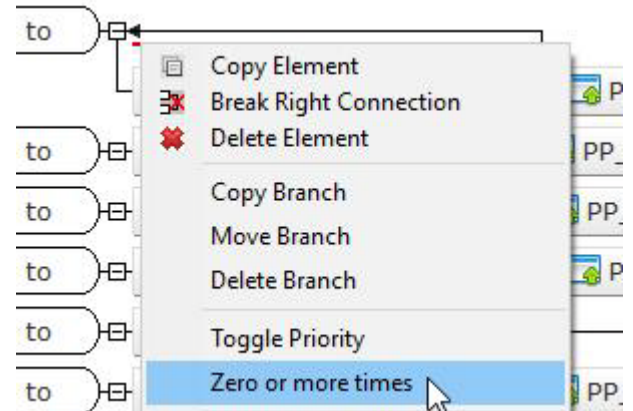
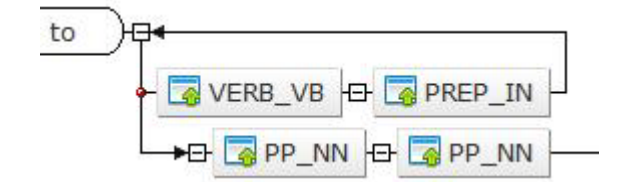
The XML special characters & < > " ' will show as themselves and can be used directly in the elements. They are automatically converted to and from & < > " ' respectively when files are either inputted or outputted. In edit mode the new line character is represented by the pi symbol (¶) and certain unprintable characters are represented by a center dot (•). (Control Z will not back delete but rather be entered as a special character.)



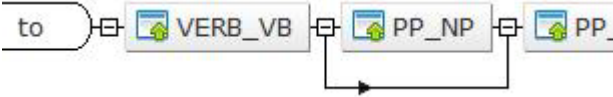
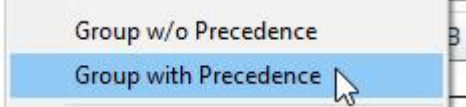
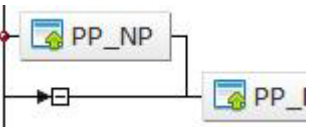
To exit edit mode simply left click on the background to save the edits or right click on the background to discharge them.

All arguments are interpreted as ASCII. It is up to the particular element's processor to understand the actual meaning of the content. This allows the mixing of numerical and literal expressions. As an example the Timer element understands the difference between "20 min" and "20 hours". (It also understands "5/24/2011", "15:30", "Tuesday", "Tues", and many other conventional representations of time and timing.)


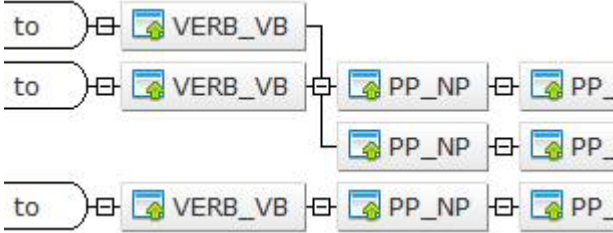
Wiring Two Port Elements

Wiring of elements is not required for creating the construction diagrams used to generate Visral Panels. However, it does become necessary when creating most diagrams that represent models and rules. The following describes the wiring behavior for elements that are in line with one another.

Example	Description
	<p>Moving cursor over an element will cause a grey button to appear to the right.</p>
 <p>The end where the grey button is located identifies the originating source for subsequent connections until a right click is performed on the diagram backplane.</p>	<p>Clicking the grey button will cause the diagrammer to enter wiring mode. The cursor will change to a cross hairs at that point and a blue arrow at the end where the grey button was will become visible.</p>
	<p>Now as the cursor is moved over elements, a blue arrow will appear.</p>
	<p>Clicking on a blue arrow when the cursor changes from cross hairs to a pointer will cause the feedback connection to be installed.</p>
	<p>In the case of syntax diagrams the diagram can be altered it from One-or-more-times to represent Zero-or-more-times by right clicking on the left facing arrow element. This will bring up a menu from which the zero or more time entry can be selected.</p>
	<p>This illustrates the results from selecting the zero or more.</p>

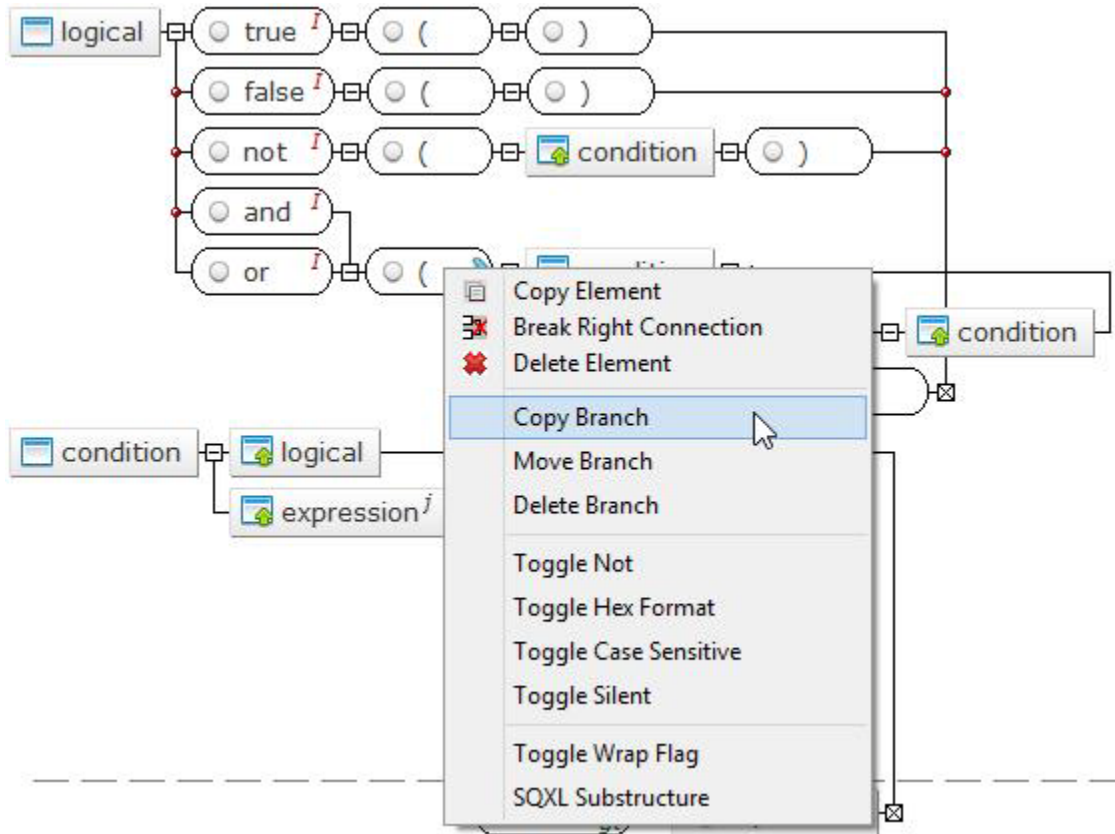
Example	Description
	<p>Moving cursor over an element will cause a grey button to appear to the right. Clicking the grey button will cause the diagrammer to enter wiring mode.</p>
	<p>Now as the cursor is moved over elements to the right, a blue arrow will appear.</p>
	<p>Because it's to the right of the originating element the connection will be installed as an ELSE.</p>
	<p>A grouping may be required before the diagram can be fed to a processor.</p>
	<p>This is how the above ELSA might appear following a grouping command to remove redundant elements and establish proper form.</p>



The following describes the wiring behavior for elements that are NOT in line with one another.

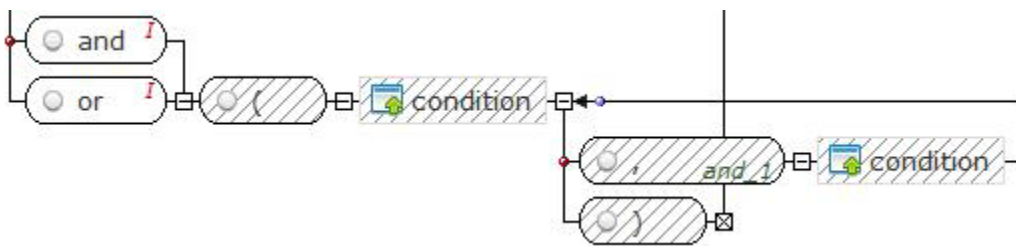
Example	Description
	<p>As the cursor is moved over elements, a blue arrow will appear. The cursor will change to a cross hairs at that point and a blue arrow at the end where the grey button was will become visible.</p>
	<p>Clicking on a blue arrow when the cursor changes from cross hairs to a pointer will cause the feedback connection to be installed.</p>

Note: This particular operation is not possible in Petri Net and Markov Chain diagrams. See the description in Volume 2 for details.

A branch or rule can be copied, moved, or deleted. The illustration below shows a branch of a rule being selected to be copied.




After left clicking on the entry Copy Branch, those elements to be copied or moved will be crosshatched until the cursor, which has changed to  has been attached to another element. Right click on the background will  cancel the operation.



Automation (Preview)

Covers:

- ✓ Automation
- ✓ Introduction to Automatons

The Automation window is selected by first selecting the  button and then choosing the particular process of interest from the available tabs.

The Automation window is used to control and monitor automaton activity. Automatons are constructed via diagrams, and then compiled producing a Panel Operator. When the Operator is executed the automaton becomes active with this window controlling and displaying its progress.

Introduction to Automatons

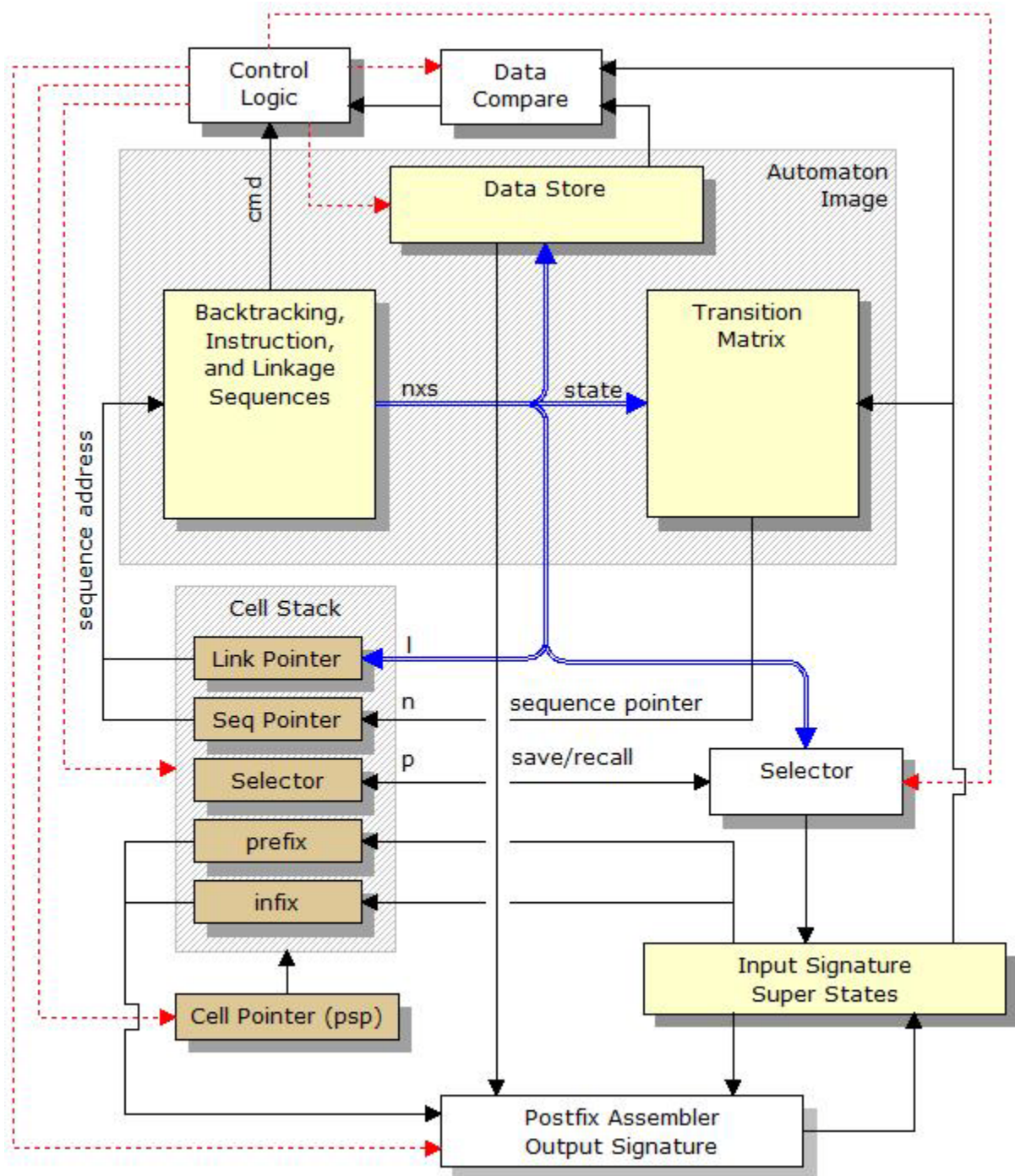
What is an automaton? Most programs are intended to complete their task as speedily as possible and then return control to the user. That means they expect to have access to whatever resources are needed to succeed. On the other hand, an automaton tends to go about its business with little user involvement or disruption except for the occasional notification that it has completed its task or some portion of it.

As an example, an automaton may constantly assemble rows of disparate data from across the internet, appending the acquisitions to a data set, and not bothering the user until some moving analysis breaches specific criteria. This is similar to how Microsoft's Outlook operates; quietly checking email servers in background and only issuing a notification when something new shows up.

The Visral automaton compiler produces executable sequences, which respond to local and remote event collections in accordance with rules represented by Visral Diagrams.

Automaton Instantiation

The following block diagram illustrates the structure of a Visral's configurable automaton. As part of the compilations all possible backtracking sequences are pre-computed and merged with application specific operations. (Super States refer to the likes of Petri Net places and Business Rule states that require simultaneous updating.)



Visral Diagrams compile to an image file that is use to configure a universal automaton. The files are saved in the subdirectory:

- *C:\Users\...\Documents\Visral 3\Automatons\AutoImages.*

The file name is the lead rule’s name with “\$1” appended to it; i.e. leadname\$1.c. The section of memory map below shaded in green represents the image of a specific automaton. The remainder of the map is space dynamically assigned by the emulator’s multitasking controller.

Multitask control descriptor
Configuration word
Automaton image pointer (Memory Map)
Allocated size of input signature space in bytes
Allocated size of output signature space in bytes
Allocated size of cell stack space in bytes
Allocated size of microcode space in bytes
Signature size in bytes
Suspended signature pointer
Suspended results pointer
Suspended stack pointer
Suspended current function pointer
Total trace counter (external trace buffer)
Error code or process id

Memory Map
Sequences
Transition
Data
Flags
Sequence Location
Transition Location
Total Length
Input Signature space
Output Signature space
Cell Stack space
Microcode space

Notes:

1. Microcode space is only applicable in the case of semiconductor implementations.
2. An Automaton DLL is available for use with third party applications.

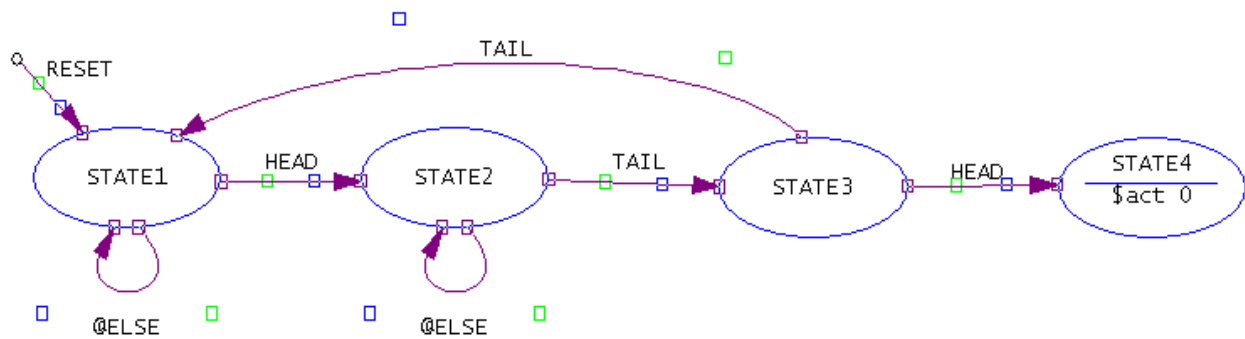
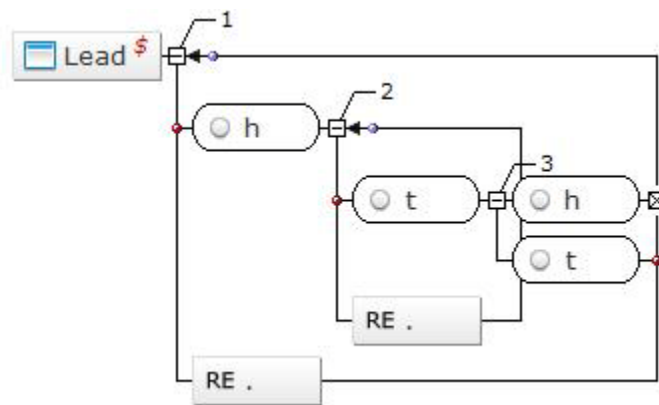
The listing below is a sample file image from the compilation of a simple three state automaton. It represents a state machine looking for a coin flip sequence head-tail-head 'hth' as illustrated in the Visral Diagram rule 'Lead' and the state diagram below that.

```

unsigned short map[ ] = {
0x0039, 0x0000, 0x0022, 0x0000, 0x0027, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000,
0x000d, 0x0000, 0x000c, 0x0068, 0x000d, 0x0069,
0x000f, 0x0000, 0x000e, 0x0074, 0x000f, 0x0075,
0x0000, 0x0000, 0x0010, 0x0068, 0x0000, 0x0069, 0x0011, 0x0074, 0x0000, 0x0075,
0x0000, 0x0000,

0x0008, 0x000a, 0x0010, 0x0016, 0x0020,

0x0083, 0x0020, 0x0001, 0x0004, 0x0009, 0x6461, 0x6c01, 0x6165, 0x0064, 0x656c,
0x6461, 0x0000, 0x0051, 0x0031, 0x0071, 0x0051, 0x0093, 0x0031};
    
```



This implementation utilizes precedence to represent the grammar. The 'RE .' element at state 1 will accept any character, but because the 'h' element is above it, it has precedence. The end result is, the 'RE .' element will never see an 'h' character. The same is true for the next stage 'RE.' in parallel with the 't' element.

Appendix

Covers:

- ✓ Product Specifications

Product Specifications

item	Approx Max	Description/Comments
Elements	30,000	Per each displayed diagram
Text	30,000 bytes	Maximum length of text within an element
Levels	4,000	Diagram Levels
Rules	4,000	Per each displayed diagram
States	2,000	Per each displayed diagram
Loaded Diagrams	16	Maximum number of diagrams open at one time
Undo actions	1000	Per diagrams
Columns	4,000	Maximum number of columns
Rows	1,000,000	Maximum number of rows
Row data	30,000 bytes	Maximum bytes per Row
Recent Files	60 files	