

**ANALYTICA**

# **User Guide**

**Analytica® for Macintosh**  
**Release 1.1.1**

Lumina Decision Systems, Inc.

---

Developed by:

Lumina Decision Systems, Inc.  
26010 Highland Way  
Los Gatos, CA 95033  
Phone: (650) 212-1212; (877) 6-LUMINA  
Fax: (650) 240-2230  
www.lumina.com; info@lumina.com



## Copyright notice

Information in this document is subject to change without notice and does not represent a commitment on the part of Lumina Decision Systems, Inc. The software program described in this document is provided under a license agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license agreement. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the licensee's personal use, without the express written consent of Lumina Decision Systems, Inc.

This document is © 1993-2003 Lumina Decision Systems, Inc. All rights reserved. The software program described in this document, Analytica, is copyrighted:

© 1982-1991 Carnegie Mellon University

© 1992-2003 Lumina Decision Systems, Inc., all rights reserved

Analytica was written using MacApp®: © 1985-1990 Apple Computer, Inc.

The Analytica software is licensed from Carnegie Mellon University exclusively to Lumina Decision Systems, Inc., and includes software proprietary to Lumina Decision Systems, Inc. Carnegie Mellon University and Lumina Decision Systems, Inc., make no warranties whatsoever, either express or implied, regarding the Analytica product, including warranties with respect to its merchantability or its fitness for any particular purpose.

The MacApp software is proprietary to Apple Computer, Inc., and is licensed to Lumina Decision Systems, Inc. only for use in combination with the Analytica program. Apple Computer, Inc., makes no warranties whatsoever, either express or implied, regarding this product, including warranties with respect to its merchantability or its fitness for any particular purpose.

Analytica is a registered trademark of Lumina Decision Systems, Inc.



# Table of Contents

---

<i>Chapter 1</i>	<i>Examining a Model</i>	<i>1-1</i>
1.1	Opening, closing, and switching models	1-1
1.2	The tool palette	1-3
1.3	Browsing with Input and Output Nodes	1-5
1.4	Examining an influence diagram window	1-7
1.5	Recognizing nodes	1-9
1.6	Selecting nodes	1-11
1.7	The Object window	1-12
1.8	The Attribute panel	1-14
1.9	Showing mid values	1-16
1.10	Printing	1-18
<i>Chapter 2</i>	<i>Viewing Results</i>	<i>2-1</i>
2.1	The Result window	2-1
2.2	Viewing a result as a table	2-5
2.3	Viewing a result as a graph	2-7
2.4	Uncertainty view options	2-9

---

2.5	Comparing results	2-14
<b>Chapter 3</b>	<b><i>Analyzing Model Behavior</i></b>	<b>3-1</b>
3.1	Varying input parameters	3-1
3.2	Analyzing model behavior results	3-4
<b>Chapter 4</b>	<b><i>Creating and Editing a Model</i></b>	<b>4-1</b>
4.1	Creating and saving a model	4-1
4.2	The model Object window	4-2
4.3	Creating and editing nodes in a diagram	4-3
4.4	Drawing arrows in a diagram window	4-7
4.5	Drawing arrows between variables in different modules	4-10
4.6	Creating alias nodes	4-12
4.7	Using an alias node	4-14
4.8	Editing an attribute of a node	4-15
4.9	Changing the class of a node	4-17
4.10	Preferences dialog box	4-19
4.11	Using stationery	4-22
<b>Chapter 5</b>	<b><i>Building Effective Models</i></b>	<b>5-1</b>
5.1	Creating a model	5-2
5.2	Testing and debugging a model	5-7

5.3	Expanding your model	5-10
<b><i>Chapter 6</i></b>	<b><i>Creating Lucid Influence Diagrams</i></b>	<b><i>6-1</i></b>
6.1	Guidelines for creating lucid and elegant diagrams	6-2
6.2	Organizing a module hierarchy	6-8
6.3	Color in influence diagrams	6-9
6.4	Changing background or node colors	6-10
6.5	Diagram Style dialog box	6-11
6.6	Node Style dialog box	6-12
6.7	Changing the size of the diagram	6-14
6.8	Taking screenshots of diagrams	6-15
<b><i>Chapter 7</i></b>	<b><i>Formatting Graphs and Tables</i></b>	<b><i>7-1</i></b>
7.1	Graph Setup dialog box	7-1
7.2	Graphing Tool setup option	7-2
7.3	Graph Frame setup option	7-3
7.4	Graph Style setup option	7-5

---

7.5	DeltaGraph options setup option	7-7
7.6	Number Format dialog box	7-9
<b>Chapter 8</b>	<b><i>Creating and Editing Definitions</i></b>	<b>8-1</b>
8.1	Creating or editing a definition	8-1
8.2	How a valid definition may change the diagram	8-4
8.3	The Expression popup menu	8-5
8.4	Object Finder dialog box	8-7
8.5	Pasting from a library in the Definition menu	8-10
8.6	Checking the validity of a variable's values	8-12
<b>Chapter 9</b>	<b><i>Creating Models to be Used by Others</i></b>	<b>9-1</b>
9.1	Using input nodes	9-2
9.2	Creating a popup menu	9-3
9.3	Using output nodes	9-5
9.4	Changing display style	9-6
9.5	Using form modules	9-7
9.6	Using balloon help	9-9
9.7	Adding icons to nodes	9-9
9.8	Adding graphics, frames, and text to a diagram	9-11
<b>Chapter 10</b>	<b><i>Using Expressions</i></b>	<b>10-1</b>
10.1	Numbers	10-1
10.2	Text values	10-3
10.3	Boolean or logical values	10-4
10.4	Operators	10-4
10.5	Math functions	10-7
<b>Chapter 11</b>	<b><i>Modeling with Arrays and Tables</i></b>	<b>11-1</b>
11.1	Introduction to Arrays	11-1
11.2	Operations on arrays	11-5
11.3	Creating an index	11-10

---

11.4	List vs. List of Labels	11-14
11.5	Editing a list	11-15
11.6	Creating an array with an Edit table	11-16
11.7	Editing a table	11-20
11.8	Calculating with arrays	11-23
11.9	Comparison and logical operations	11-25
11.10	Conditional operators	11-26
<b>Chapter 12</b>	<b><i>Array Function Reference</i></b>	<b><i>12-1</i></b>
12.1	Overview	12-1
12.2	Functions that create lists	12-5
12.3	Functions that create arrays	12-6
12.4	Array-reducing functions	12-11
12.5	Transforming functions	12-16
12.6	Functions that select part of an array	12-20
12.7	Interpolation functions	12-23
12.8	Other array functions	12-26
12.9	Matrix functions	12-28
12.10	Control functions	12-31
<b>Chapter 13</b>	<b><i>Expressing Uncertainty</i></b>	<b><i>13-1</i></b>
13.1	Choosing an appropriate distribution	13-2
13.2	Defining a variable as a distribution	13-6
13.3	Including a distribution in a definition	13-8
13.4	Overview of built-in probability distributions	13-9
13.5	Probabilistic Calculation	13-9
13.6	Uncertainty Setup dialog box	13-11
<b>Chapter 14</b>	<b><i>Using Continuous Probability Distributions</i></b>	<b><i>14-1</i></b>
14.1	Continuous distribution functions	14-1
14.2	Related function	14-9
<b>Chapter 15</b>	<b><i>Using Discrete Probability</i></b>	<b><i>15-1</i></b>



15.1	Using a probability table	15-1
15.2	Other discrete distribution functions	15-7
15.3	Using a deterministic conditional table	15-10
<b><i>Chapter 16</i></b>	<b><i>Analyzing Uncertainty and Sensitivity</i></b>	<b><i>16-1</i></b>
16.1	Statistical functions	16-1
16.2	Sensitivity analysis functions	16-9
16.3	X-Y results	16-11
16.4	Scatter plots	16-13
16.5	Importance analysis	16-15
<b><i>Chapter 17</i></b>	<b><i>Modeling Changes over Time</i></b>	<b><i>17-1</i></b>
17.1	The Time index	17-1
17.2	Using the Dynamic function	17-2
17.3	More about the Time index	17-4
17.4	Initial values for Dynamic	17-8
17.5	Using arrays in Dynamic	17-8
17.6	Dependencies with Dynamic	17-9
17.7	Uncertainty and Dynamic	17-11
<b><i>Chapter 18</i></b>	<b><i>Importing and Exporting Data</i></b>	<b><i>18-1</i></b>
18.1	Copying and pasting	18-1
18.2	Importing and exporting	18-3
18.3	Publish and Subscribe	18-4
18.4	Printing to a file	18-7
18.5	Sending a model via electronic mail	18-8
18.6	Edit table data import/export format	18-9
<b><i>Chapter 19</i></b>	<b><i>Working with Large Models</i></b>	<b><i>19-1</i></b>
19.1	Show module hierarchy preference	19-1
19.2	The Outline window	19-2
19.3	Finding variables	19-5

---

19.4	Managing attributes	19-6
19.5	Invalid Variables	19-9
19.6	Using filed modules and libraries	19-9
19.7	Add Module dialog box	19-12
19.8	Combining models into an integrated model	19-13
19.9	Managing windows	19-18
<b><i>Chapter 20</i></b>	<b><i>Building Functions and Libraries</i></b>	<b><i>20-1</i></b>
20.1	Creating a function	20-1
20.2	Attributes of a function	20-2
20.3	Parameter qualifiers	20-3
20.4	Example function	20-5
20.5	Libraries	20-6
<b><i>Appendix A</i></b>	<b><i>Menus</i></b>	<b><i>A-1</i></b>
A.1	File menu	A-1
A.2	Edit menu	A-3
A.3	Object menu	A-4
A.4	Definition menu	A-6
A.5	Result menu	A-10

A.6	Diagram menu	A-11
A.7	Window menu	A-13
<i>Appendix B</i>	<i>Analytica Specifications</i>	<i>B-1</i>
<i>Appendix C</i>	<i>Memory</i>	<i>C-1</i>
C.1	Memory usage	C-1
C.2	Increasing memory allocation	C-2
<i>Appendix D</i>	<i>Selecting the Sample Size</i>	<i>D-1</i>
<i>Appendix E</i>	<i>Error Message Types</i>	<i>E-1</i>
E.1	Warning	E-1
E.2	Lexical error	E-2
E.3	Syntax error	E-2
E.4	Evaluation error	E-3
E.5	Invalid number	E-4
E.6	System error	E-4
E.7	Out of memory error	E-4
<i>Appendix F</i>	<i>Reserved Words</i>	<i>F-1</i>
<i>Appendix G</i>	<i>References</i>	<i>G-1</i>
<i>Chapter</i>	<i>Glossary</i>	<i>GL-1</i>
<i>Chapter</i>	<i>Index</i>	<i>15</i>

# Introduction

---

Welcome to the *Analytica User Guide*. This document describes how to use Analytica 1.1.1 for the Macintosh. Use the *Analytica Tutorial* for a hands-on introduction to the basics of Analytica, then refer to this guide when you need more detailed information.

## If you don't read manuals

You may find that you can successfully use many of Analytica's features without reading this *User Guide*, especially after going through the *Analytica Tutorial*. If so, you can use this *User Guide* mainly as a reference when you get stuck. You may still find it valuable to look at a few sections, because they give valuable tips and techniques. In particular, we recommend Chapters 5, 6, 11, and 13.

*Chapter 5* provides general tips on the process of building a model, distilled from the experience of master modelers (including Newton and Einstein). It suggests guidelines for building effective models that are clear, comprehensible, and reliable, and that focus on what really matters — the decisions, objectives, and key uncertainties. These tips are helpful with any modeling software, but we have designed Analytica to make them especially easy to adopt. For example, you can use Analytica's hierarchical influence diagrams to organize a complex model.

*Chapter 6* explains how to create diagrams that are truly lucid — and how to avoid creating spaghetti diagrams.

*Chapter 11* provides an overview of Analytica's intelligent arrays™. With intelligent arrays you can create and analyze large multidimensional models with surprising power and ease. However, there are some subtleties to the effective use of arrays. Prior experience with spreadsheets or arrays in programming

languages may actually mislead you about how to use arrays in Analytica. We suggest that you look at Chapter 11 if you plan to create models with extensive use of arrays.

*Chapter 13* discusses how to select an appropriate probability distribution to express the uncertainty in a quantity. It also provides an overview of how Analytica computes probability distributions using Monte Carlo and other random sampling methods, and your options for controlling and displaying probabilistic values.

## Requirements

To use Analytica, you need the following configuration:

- Power Macintosh or equivalent computer running the Macintosh Operating System.
- A hard drive with at least 4 megabytes of free disk space.
- At least 8 megabytes of RAM; large models may require 16 megabytes or more.
- Macintosh Operating System 7.0 or later.

To install Analytica, double click on the file 'Analytica Installer' and follow the instructions.

## Conventions used in this guide

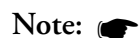
This guide uses the following conventions:

### Typographic conventions

Example	Meaning
<i>behavior analysis</i>	Analytica terms when introduced. Most of these terms are included in the Glossary
<b>Diagram</b>	menus and menu commands
Sequence( )	functions
Price - DownPmt	expressions, definitions
<i>enter</i>	key on keyboard
<i>Foxes at end</i>	title or identifier of a variable or other Analytica object

### Terminological conventions

Term	Meaning
click	press and release the mouse button one time
double-click	press and release the mouse button two times in quick succession
drag	press and hold down the mouse button, move the cursor to a new location on the screen, and then release the mouse button
press	press and hold down the mouse button
select	click on an interface object, such as a node in a diagram or a cell in a table; selected objects appear highlighted



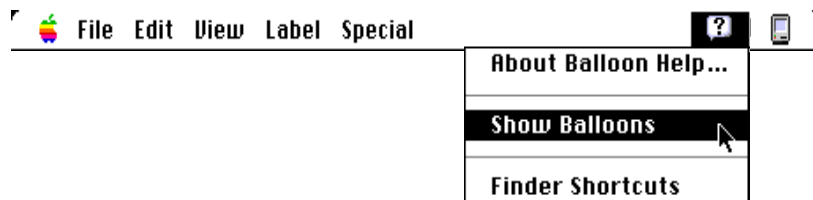
Notes tell you useful or important information.

## Using Balloon Help

Balloon Help is available only if your Macintosh has System 7. When Balloon Help is turned on, Analytica provides help text for all features. When you move the cursor over any feature (for example, an icon, button, or menu item), explanatory text appears in a cartoon-like balloon pointing to the feature. We recommend that you use Balloon Help when you first start using Analytica.


To turn Balloon Help on, do one of the following:

- Select **Show Balloons** from the  menu.



- Press the *help* key.
- Press  $\text{⌘}-?$ .

To turn Balloon Help off, do one of the following:

- Select **Hide Balloons** from the  menu.
- Press the *help* key.
- Press  $\text{⌘}-?$ .

### Note:

If after turning on Balloon Help, you move the cursor over the Analytica diagram and see empty balloons, the Help file is missing from the Analytica folder. Reinstall Analytica to replace the Help file.

## User Guide Examples Folder

In the Examples folder distributed with Analytica is a folder of User Guide Examples. This folder contains an Analytica model for each of Chapters 9, 10, 11, 12, 14, 15, and 16, and three Analytica models for Chapter 17. Use these models to more closely examine the examples shown in this *User Guide*.

---

# 1

## Examining a Model

---

This chapter introduces the basics of interacting with a model in Analytica. It describes how to start up and explore a model, including its Diagram, Object, and Result windows, and how to print the contents of windows.

### 1.1 Opening, closing, and switching models

**Models** A *model* is a collection of variables and modules used to represent a situation of interest. Between sessions, a model is stored in an Analytica document file.

**Opening a model** There are two ways to open an existing model from the Macintosh Finder, as with any Macintosh application:

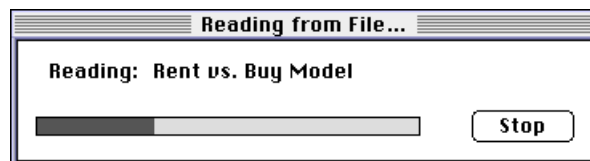


- Double-click on the icon of the document file containing the model.




- Double-click on the Analytica icon to start up Analytica. A file dialog prompts you to locate and open a model.

As the model is read, a dialog box indicates progress:

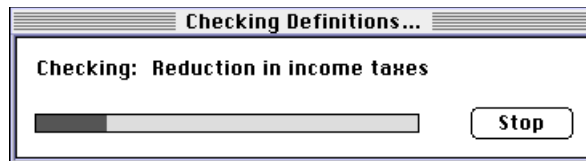





**Note:**  Clicking on the **Stop** button halts the model reading process and results in a partially loaded model (the diagram will be incomplete).

---

After reading in the model, Analytica checks the definitions.



**Note:**  Clicking on the **Stop** button halts the checking of definitions and results in a diagram with missing arrows.

---

If the model contains variables without syntactically correct definitions, the “invalid variables” window displays (see Section 19.5)

## Closing a model

To close a model, select **Close Model** from the **File** menu. If you have made any changes to the model, a dialog box asks you whether you want to save the changes before closing.

## Switching to another model

Only one Analytica model can be open at a time. To switch to another model, first close the model. Then select **Open Model** from the **File** menu. A dialog box prompts you to locate and open another model.

## Quitting Analytica

To quit Analytica, select **Quit** from the **File** menu. If you have made any changes, a Save dialog asks you whether you want to save your model before quitting.

## 1.2 The tool palette

When you open a model, the tool palette appears in the upper left of your screen. It provides a set of buttons to navigate around a model, to open different views of a model, and to change between browse and edit modes.



The top five buttons on the tool palette apply to the active Analytica window (the frontmost window whose top bar contains dark stripes).



### Parent Diagram button

Opens a Diagram window for the module or model containing the active diagram window or variable. This button is grayed out if you are looking at the Diagram window for the top-level (or root) model.



### Outline button

Opens the Outline window and highlights the selected node or active module in the outline. See “The Outline window” on page 19-2.



### Object button

Opens an Object window for the selected node or active module. See “The Object window”, Section 1.7.

**Result button**

Opens a Result window for the selected variable. (See “The Result window” on page 2-1.) This button is grayed out if no variable is selected. The keyboard equivalent for the result button is ⌘-R.

**Definition button**

Opens a view of the definition of the selected variable. If the variable is defined as a distribution or sequence, the Object Finder opens (see Section 8.4); if it is defined as a table or probability table, its Edit Table window opens (see page 11-4). Otherwise, an Attribute panel (see Section 1.8) or an Object window (see Section 1.7) opens, depending on the Edit Attributes setting in the Preferences dialog box (see Section 4.10.) This button is grayed out if no variable is selected. The keyboard equivalent for the definition button is ⌘-E.

The three tool buttons determine the mode of interaction with the model. One mode is always selected.

**Browse tool button**

Use to interact with the model in Browse mode. See “Browsing the window” on page 1-5.

**Edit tool button**

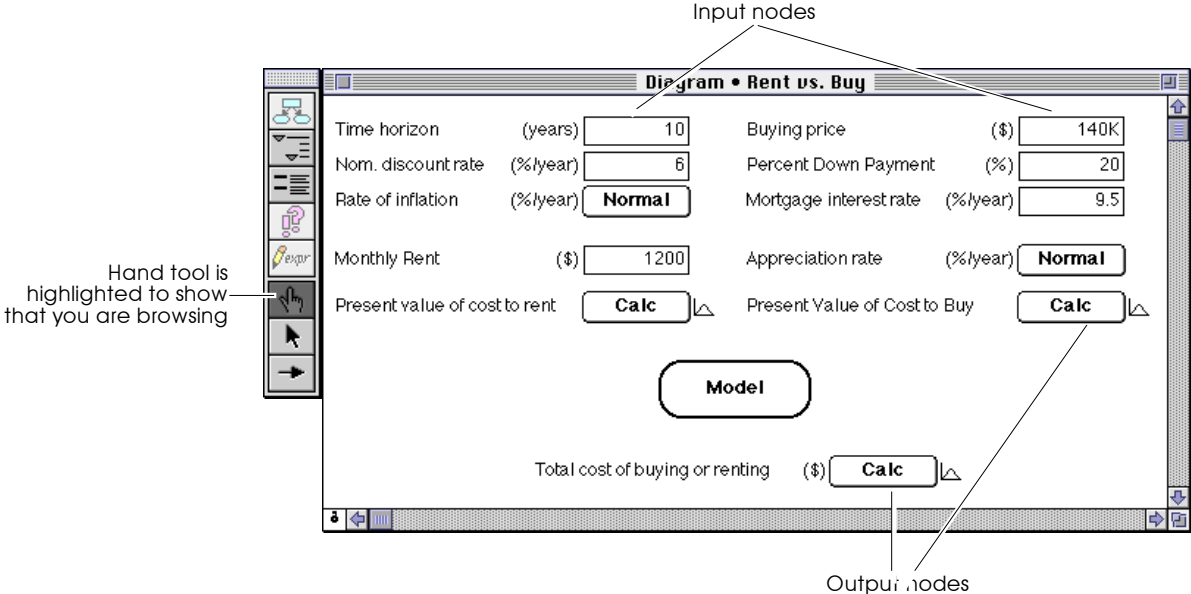
Use to interact with the model in Edit mode. See Chapter 4, “Creating and Editing a Model.”

**Arrow tool button**

Use to interact with the model in Arrow mode, to add or delete arrows (dependencies) among the components of the model. See Section 4.4, “Drawing arrows in a diagram window.”

# 1.3 Browsing with Input and Output Nodes

When you open a model with input and output nodes, Analytica first displays a top level Diagram window, similar to the example here.



The *input nodes* show values that you can change to see their effects on the final values. The *output nodes* each show a **Calc** button. At least one node contains the details of the model.

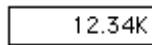
## Browsing the window

An existing model opens in Browse mode. In Browse mode, the Browse tool button is highlighted in the tool palette, and the cursor is a hand (☞).



Use the Browse tool to change input node values, view output node results, and examine the model by opening windows to see more detail.

## Viewing input node values



An input field lets you see a single numeric or text value. Click in the box to change the value.



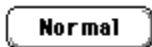
A popup menu lets you choose from a menu of alternatives. To see the alternatives, press on the popup menu. To select an alternative, move the cursor to the alternative and release the mouse.



A **List** button lets you see an ordered set of values. To see the values, click on the button. To change a value, click in its cell. For more about lists, see “Editing a list” on page 11-15.

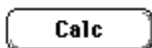


An **Edit Table** button lets you see a multidimensional set of values in one or more tabular spreadsheet-like windows. To see the values, click on the button. To change a value, click in its cell. For more about tables, see “Editing a table” on page 11-20.



A **Distribution** button lets you see a probability distribution. To see the distribution and its parameters, click on the button. For more about probability distributions, see Chapter 13, “Expressing Uncertainty.”

## Viewing output node values



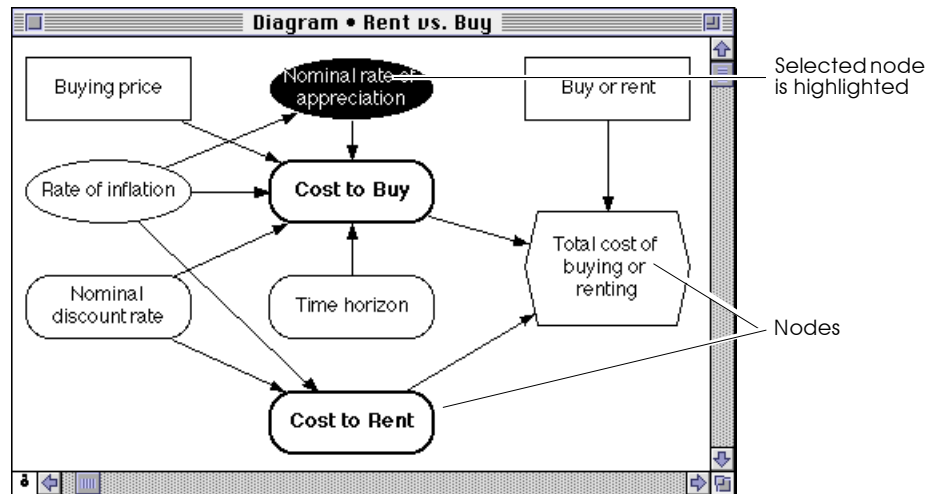
If the value of an output node has not yet been computed, the **Calc** button appears in the node. Click on the **Calc** button to compute and display the value. Chapter 2, “Viewing Results,” describes how to interpret and redisplay results.

## Opening model details

To see the structure of the model, double-click on the details (rounded, thick-outline) node. A diagram window showing an influence diagram will display (see Section 1.4).

## 1.4 Examining an influence diagram window

When you open a model detail window, or a model without input and output nodes, Analytica displays a Diagram window for the model. The Diagram window depicts the model as an *influence diagram*. An influence diagram is an intuitive graphical view of the structure of a model, consisting of nodes and arrows. Each node depicts a variable or a module.



A *variable* is any object that has a value or can be evaluated. Nodes with thin outlines depict variables. A *module*, with a thick outline, contains its own influence diagram.

The arrows in a Diagram window depict the *influences* among the variables. An influence arrow from one variable to another means that the value of the first variable directly affects the value (or probability distribution) of the second variable.

For example, in the preceding diagram, the arrow from *Buying price* to *Cost to Buy* means that the price of the house affects the overall cost of purchasing it. A higher price means a greater cost, given a fixed *Rate of inflation* and *Nominal discount rate*. The influence diagram shows the essential qualitative structure of the

model, unobscured by details of the numbers or mathematical formulas that may underlie that structure.

For more on using influence diagrams to build clear models, see Chapter 6, “Creating Lucid Influence Diagrams.”

## Opening details from a diagram


To see more details of a model, double-click on nodes in the Diagram window:

- Double-click on a variable (thin outline) node to open its Object window. See Section 1.7, “The Object window.”
- Double-click on a module (thick outline) node to see its Diagram window, showing the next level of detail of the model.

## Going to the parent diagram

To see the diagram that contains the active module or variable, click on the Parent Diagram button in the tool palette. The module or variable will be highlighted in the parent diagram.



Note: 

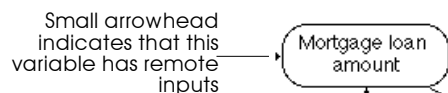
---

If the active diagram is of the top model, it has no parent diagram, and the Parent Diagram button is grayed out.

---

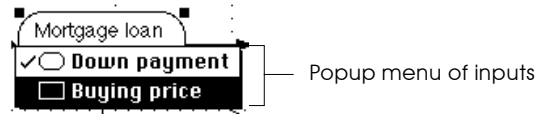
## Finding remote inputs and outputs

When a variable depends on a remote variable, that is, a variable in another module that is not visible, a small arrowhead appears to the left of the node. If a variable has a remote output, a small arrowhead appears to the right of the node.



To go to the Diagram window containing a remote variable:

1. Press on the small arrowhead. A popup menu appears listing all inputs (or outputs), including those that are not remote.





2. Drag the cursor to the desired input (or output) and release the mouse button. The Diagram window containing the remote variable opens, and the remote variable's node is highlighted.

## Viewing results



Click on a variable node to select it; it becomes highlighted. Then click on the Result button in the tool palette to show the value of the variable as a table or graph in a Result window. Chapter 2, “Viewing Results,” discusses how to interpret and redisplay results.

**Note:** 

If the value has not already been computed, you may need to wait while the result calculates, and the waiting cursor () appears.

## 1.5 Recognizing nodes

Each node shape in a diagram represents a different class of objects. Here are the classes and their corresponding node shapes:



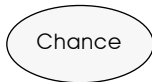
A rectangular node depicts a *decision variable*—that is, a quantity that the decision maker can control directly. For example, whether or not you take an umbrella to work is your decision. If you are bidding on a contract, how much you bid is your decision.





Variable

A rounded, thin-outline node depicts a *general variable*—that is, a quantity whose class is not determined more precisely, or a quantity that the decision maker cannot affect directly and that is not defined as probabilistic. Use a general variable initially if you're not sure what kind of variable you'll need, then change the node class later, if appropriate.



Chance

An oval node depicts a *chance variable*—that is, a variable that is uncertain and that the decision maker cannot control directly. A chance variable is usually defined by a probability distribution. For example, whether or not it rains is a chance variable (unless you are a rain god). And whether or not your bid is the winning bid is a chance variable in your model, although it is a decision variable for the person or organization requesting the bid.



Objective

A hexagonal node depicts an *objective variable*—that is, a quantity that evaluates the relative desirability of possible outcomes of combinations of decision and chance variables. Most models should contain a single objective node, although the objective can comprise several subobjectives.



Module

A rounded, thick-outline node depicts a *module*—that is, a collection of nodes organized as a separate diagram. Modules can themselves contain nested modules.



Index

A parallelogram-shaped node depicts an *index variable*. An index is used to define a dimension of an array. For example, *Year* is an index for an array containing the U.S. GNP for the past 20 years. Or *Nation name* is an index for an array of GNPs for a collection of nations (see Section 11.1, “Introduction to Arrays”). Index values appear in the row and column headers of a table, and in the *x* axis and key of a graph.

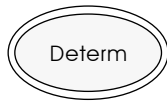


Constant

A trapezoid-shaped node depicts a *constant*—that is, a variable whose value is fixed. A fixed value has no inputs and is not computed. Examples of numerical constants are the atomic weight of oxygen or the number of feet in a kilometer. It is good practice to define such values as constants, so you can refer to them by name; otherwise, you must type their numerical values into each expression that includes them and search for the values when you need to change them.



A node resembling an arrow tail pointing right depicts a *function*. You can define functions to augment the functions provided in Analytica; see Chapter 20, “Building Functions and Libraries.”



An oval node with a double rounded outline depicts a *determ (deterministic) variable*—that is, a variable whose value cannot be directly controlled by the decision maker, and that is not uncertain or probabilistic. This node class is not included in the node palette to encourage use of the general variable.

## 1.6 Selecting nodes

To perform an operation on a diagram, you must first select a node (or a set of nodes), then select the operation to perform. There are various ways to select nodes (similar to selecting icons in the finder):

**To select one node** Click once on the node to select it. The selected node is highlighted.

You can also press the *tab* key to select one node at a time.

**To select multiple nodes** Select one node, then click on another node while holding down the *shift* key. This operation adds the new node to the set of selected nodes, highlighting them all. By repeating this process, you can select as many nodes as you wish.

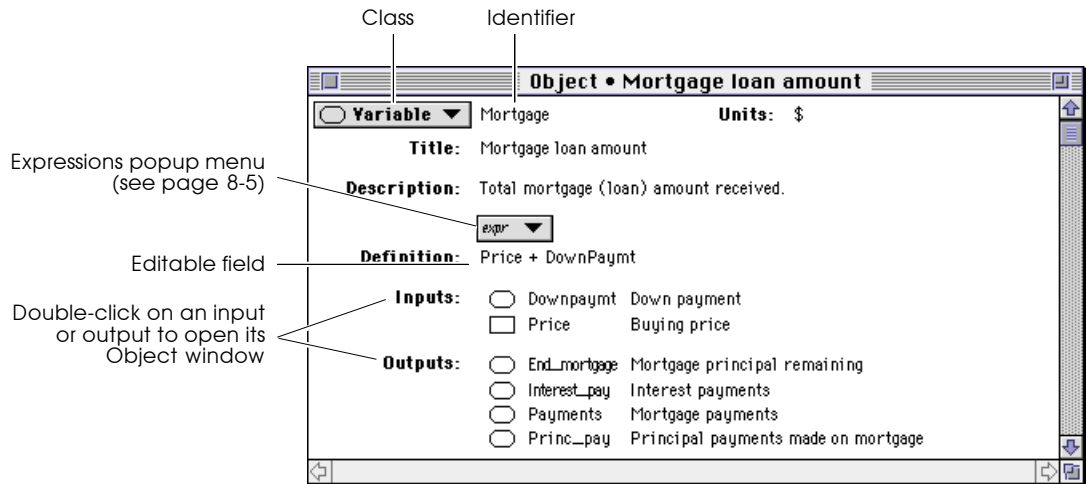
If the nodes are close together, you can also select them by dragging a selection rectangle around them.

**To deselect one node** Click on a selected node while holding down the *shift* key. This operation removes the node from the selection, leaving the remaining nodes selected.

**To deselect all nodes** Click on the background of the diagram to deselect all nodes.



## 1.7 The Object window

The *Object window* shows the attributes that together specify a node. For a variable, as shown in the following figure, these attributes include the class, identifier (a brief, unique name), title, units, description, definition, inputs, and outputs. See the Glossary for descriptions of the attributes.



### Opening an Object window

There are several ways to open the Object window for an object titled *X*:

- Double-click on *X*'s node in its Diagram window (see page 1-7).
- Select *X* in its Diagram window and click on the Object button (  ) in the tool palette.
- Double-click on the entry for *X* in the Outline window (see page 19-2).
- If a Result window for *X* is displayed, click on the Object button (  ) in the tool palette.
- Double-click on the entry for *X* in the Inputs or Outputs field of another variable.

## Examining inputs and outputs

When you are looking at an Object window for a variable, you can easily view the Object window for any of the variable's inputs or outputs. Double-click on a node symbol, identifier, or title of a variable in the list of inputs or outputs.

## Returning to the parent diagram

Click on the Parent Diagram button in the tool palette to see the diagram that contains this node, with the node highlighted.

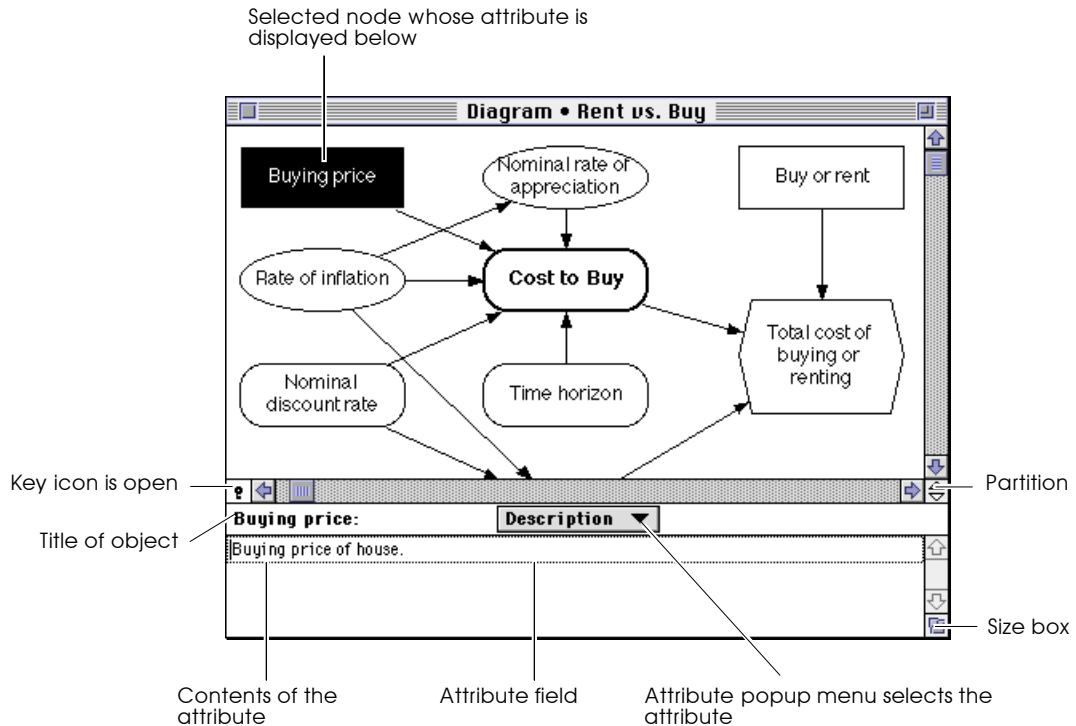


## Displaying additional attributes

- To display the value of a variable and its inputs, see “Showing mid values”, Section 1.9.
- To display additional attributes and create new attributes, see “Managing attributes”, Section 19.4.

## 1.8 The Attribute panel

The *Attribute panel* provides an alternative way to view attributes of a node. The attribute panel appears as an extension below a Diagram window.



### Displaying the attribute

1. Click on the Key icon (🗝️) to display the Attribute panel.
2. Select a node to examine by clicking on it in the diagram.

#### Note: 🖱️

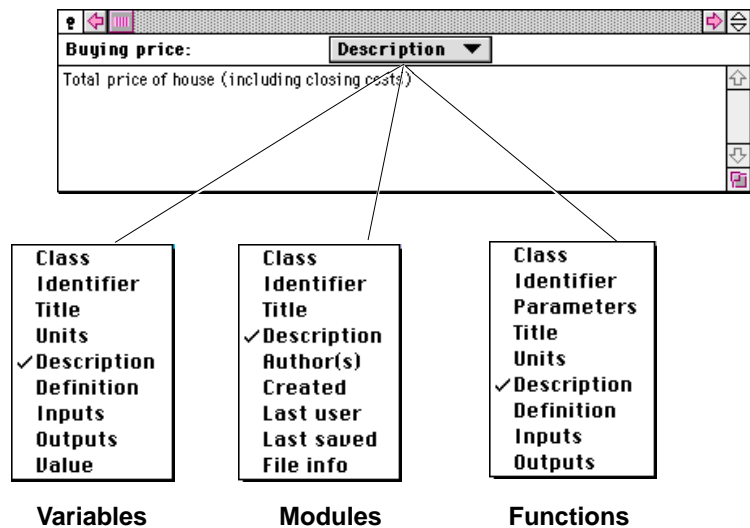
If multiple nodes are selected, click on the diagram background to deselect them, then click on the node you wish to examine.

3. To examine a different attribute, press on the Attribute popup menu, and select the desired attribute.

4. To examine the same attribute for another node, select that node in the Diagram window. If no node is selected (you have clicked on the background of the diagram), the corresponding attribute for the module is displayed.

## The Attribute popup menu

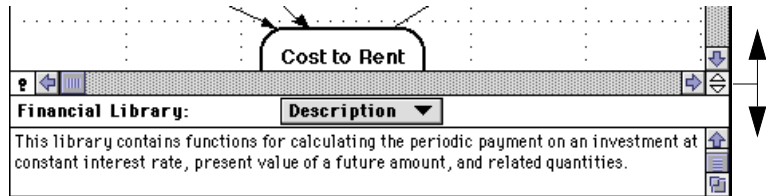
When the Attribute panel is displayed, the Attribute popup menu is located at the top center of the Attribute panel. Use this popup menu to select another attribute to view. Variables, modules, and functions have different sets of attributes, as shown here:



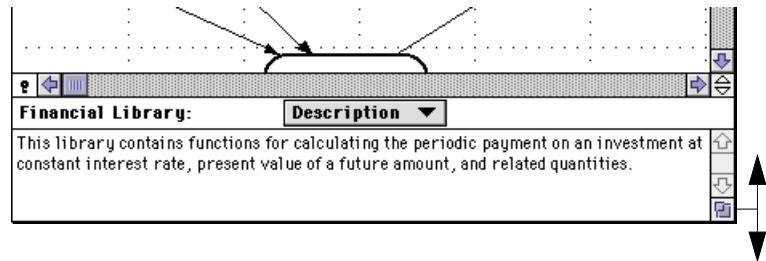
See the Glossary for descriptions of these attributes. To display other attributes or to add new ones, see “Managing attributes”, Section 19.4.

## Changing the panel size

To change the height of the diagram in relation to the diagram's Attribute panel, drag the partition up or down.



To change the size and shape of the Attribute panel, drag the size box up or down; the height of the Diagram panel remains fixed.



To change the width and shape of the Diagram and Attribute panels, drag the size box right or left.

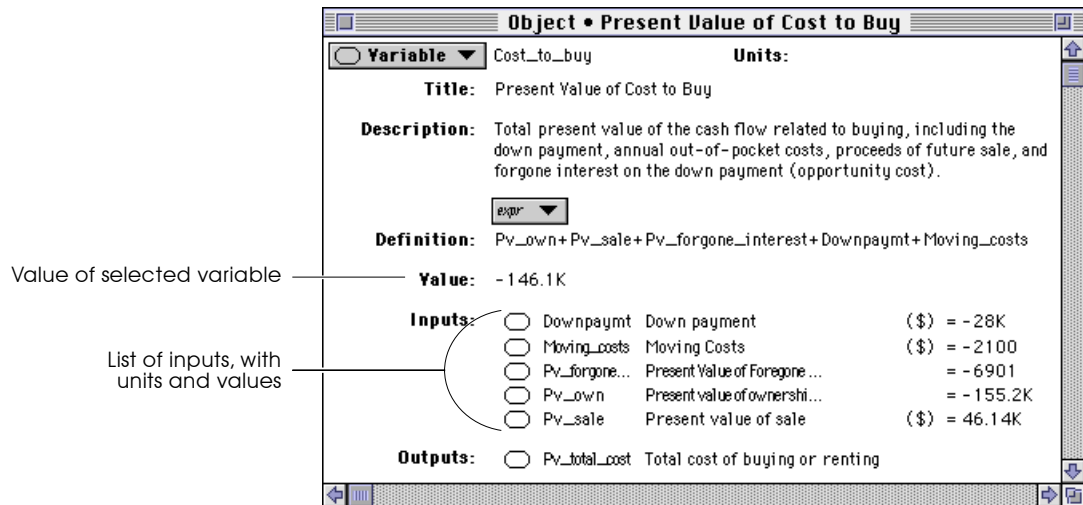
## Closing the Attribute panel

To close the Attribute panel, click on the Key icon (🔑).

## 1.9 Showing mid values

The *mid value* or deterministic result of a variable is computed by holding each uncertain (probabilistic) input at a single, central value. The mid value for a probability distribution is its median. The mid value of a variable is computed by using the mid value of each input. If all inputs are certain (nonprobabilistic), the calculated value is still referred to as the mid value in Analytica.

To show the mid values of variables in the value attribute, select **Show with Values** from the **Object** menu. Mid values that are single values will display in object windows and the attribute panel. You can display these values to check that a calculation is performing correctly, or to find errors in the model.



**Array values** If the mid value of the selected variable is an array, rather than a single number, the **Result** button appears.

**Result** indexed by Buy or rent

To display the array, click on the **Result** button. (This is equivalent to clicking on the Result button on the tool palette, described in Section 1.2.) A Result window opens, showing the values either as a graph or as a table. For information about the Result window, see Chapter 2, "Viewing Results."



## Values of inputs

When **Show with Values** is turned on, the list of inputs displays one of the following to each input:

- The mid value, if it is a single number (or text) and has been computed.
- **Calc** if it has not been computed. To calculate and view the value, select the input as the variable you are examining and display its object window or value attribute.
- **Result** if the mid value is an array. To display it, select the input node as the variable you are examining. The value attribute displays the **Result** button; click on the **Result** button to see the values.

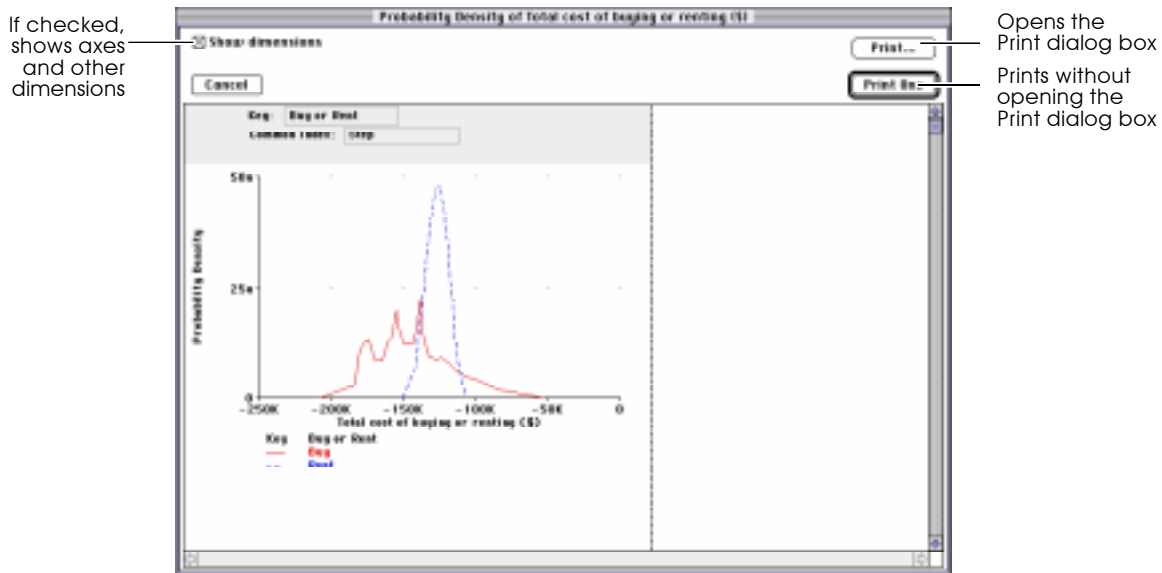
## 1.10 Printing

### Printing a Diagram, Object, or Outline window

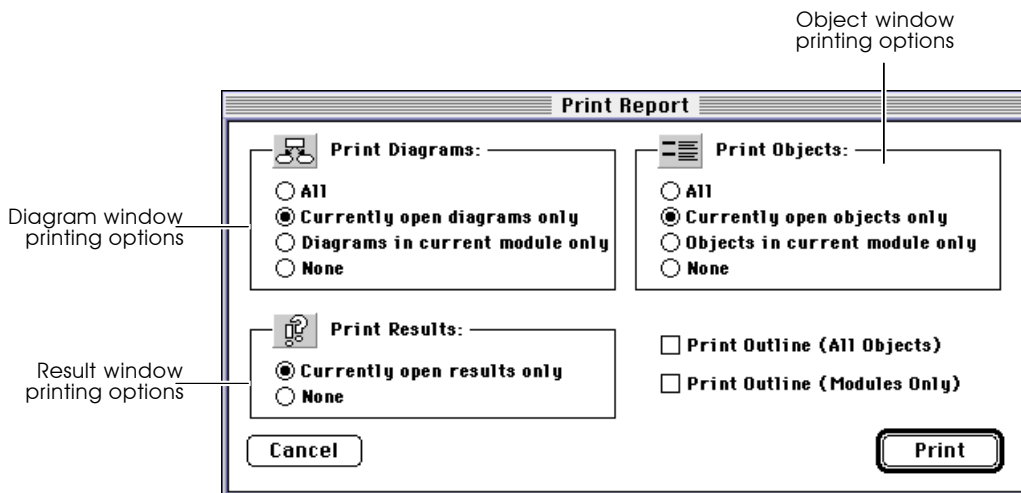
To print the contents of a Diagram, Object, or Outline window, use the **Print** command from the **File** menu. Set options such as page orientation using the **Page Setup** command from the **File** menu. See your printer or system software documentation to learn about the options that appear in Print dialog boxes.

### Previewing and printing a result

When you print a result, table, or graph using the **Print** command, a Preview window appears before printing. This window shows the result as it will be printed, and provides an option for showing or hiding the Index variable titles in the printed version.



**Print multiple windows** To print the contents of several windows at one time, use the **Print Report** command in the **File** menu.



In addition to the Diagram, Result, and Object window options, there are two checkboxes:

**Print Outline (All Objects)**

If checked, this option prints an outline of all of the objects in the model. It prints a list of all nodes by title, indented to show their position in the module hierarchy.

**Print Outline (Modules Only)**

If checked, this option prints the model hierarchy as an indented list containing only the modules.

---



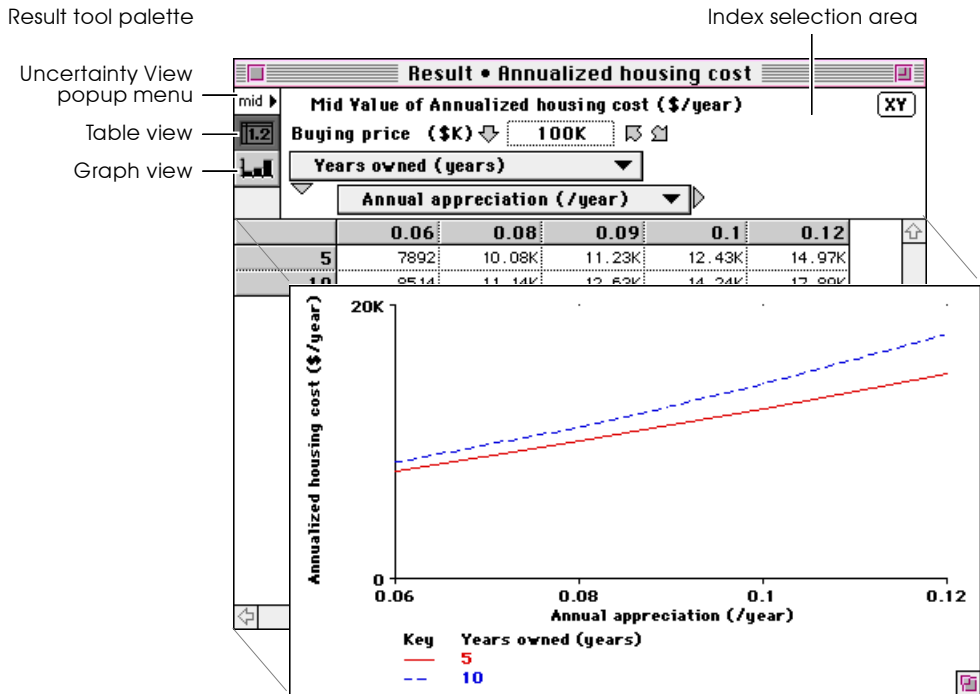
# Viewing Results

---

This chapter describes the elements of Result windows, how to view results as graphs and as tables, and how to view results that have more than two dimensions. It also discusses how to select a method for displaying uncertainty about probabilistic values.


## 2.1 The Result window

Analytica computes the value of a variable from that variable's definition and displays the value in a Result window. If the value is probabilistic or an array, or both, it is displayed in a Result window as either a table or graph. The following figure shows a Result window of an array with the graph superimposed on a table.



## Opening a Result window

To open a Result window for a variable in an influence diagram, select the variable and do any of the following:

- Click on the Result button (  ).
- Select **Show Result** from the **Result** menu.
- Select an uncertainty view option (such as Mid Value or Mean Value) from the **Result** menu.
- Select **Value** from the Attribute panel's popup menu.
- Select **Probvalue** from the Attribute panel's popup menu.
- Press ⌘-R.

To open a Result window for an output node, click on the **Calc** or **Result** button.

## The Result tool palette



- The Uncertainty View popup menu. Pressing this popup menu brings up a menu of options for viewing the result data (see page 2-9).



- The Table View button. Clicking this button displays the results as a table (see page 2-5).

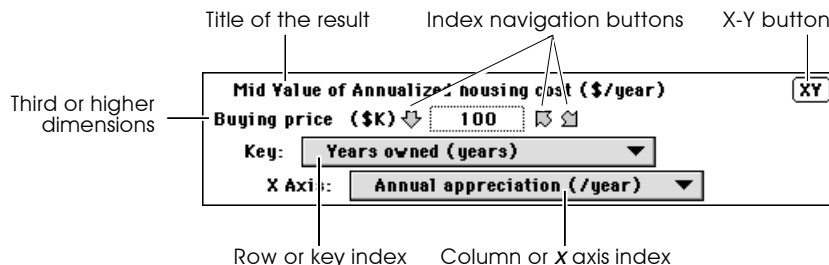


- The Graph View button. Clicking this button displays the results as a graph (see page 2-7).

Toggle between the table and graph views using the Table View and Graph View buttons.

## Index selection area

The top portion of a Result window is the *index selection area*, which identifies the rows and columns of a table, or the *x*-axis and key of a graph. Buttons and popup menus allow you to rearrange the table or graph by interchanging indexes.

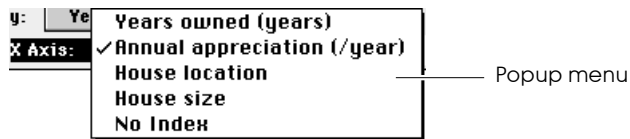


The index selection area contains these items (example variables and indexes in the following text refer to the figure above):

- The title of the result, including the active uncertainty view, title of the variable and units, if any; here, it is *Mid Value of Annualized housing cost (\$/year)*.
- Index navigation buttons, if the result has three or more dimensions (see the next page).

- The third or higher dimensions, if any (*Buying Price* is a third dimension).
- The row or key index (*Years owned*).
- The column or *x*-axis index (*Annual appreciation*).

Click on the popup menu (indicated by the down arrow (▼)) for either index (row or column) to select a different index (or “No Index”).



- The X-Y button (see Section 16.3, “X-Y results”).

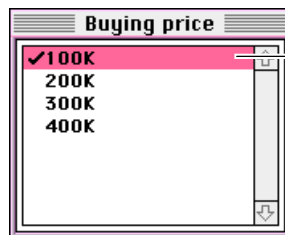
## Index navigation buttons

When the result has three or more dimensions, not all values can display on the table or graph. For each index, or dimension, beyond the second, a set of navigation buttons appears in the index selection area.

Use the index navigation buttons to move among index values for the third (or higher) dimension of results:

- Click the left arrow (←) to select the previous index value.
- Click the right arrow (→) to select the next index value.
- Click the down arrow (▼) to open a dialog box showing all the values for this index.

Dialog box listing all values for the selected index (*Buying Price*)

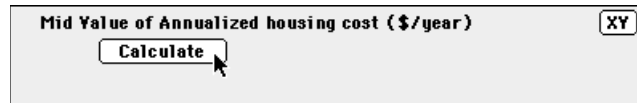


Selected value is 100K  
Click on a value to select it and to close the dialog box

**The default view** When you first display a Result window, Analytica displays the result as a graph, if possible, and otherwise as a table. You can change the default option with the “Default result view” setting in the Preferences dialog box (see Section 4.10).

When you display the Result window again, the view to be displayed is recalled from earlier in the session or from the previously saved session.

**Recomputing results** When a Result window is open, and then the value of an input to the evaluated variable is changed, a **Calculate** button appears in the index selection area.



Click on the button to recalculate the result.

## 2.2 Viewing a result as a table

**Displaying a table** If a graph is displayed, change it to the corresponding table by clicking on the Table View button ()

**Display of values** The table view displays one or two dimensions of a variable.





Three-dimensional table

The display options depend on the number of dimensions in the variable.

### One dimension

The index is displayed vertically (there are no options).

### Two dimensions

You can choose which index is displayed horizontally using the column index popup menu (  ), and which index is displayed vertically using the row index popup menu (  ).

### Three or more dimensions

You can select a two-dimensional view using the row and column popup menus. Select specific values for the third or higher dimensions using the index navigation buttons.

## Formatting numbers

You can specify the number format for a table's contents, using the Number Format dialog box.

To display the Number Format dialog box:

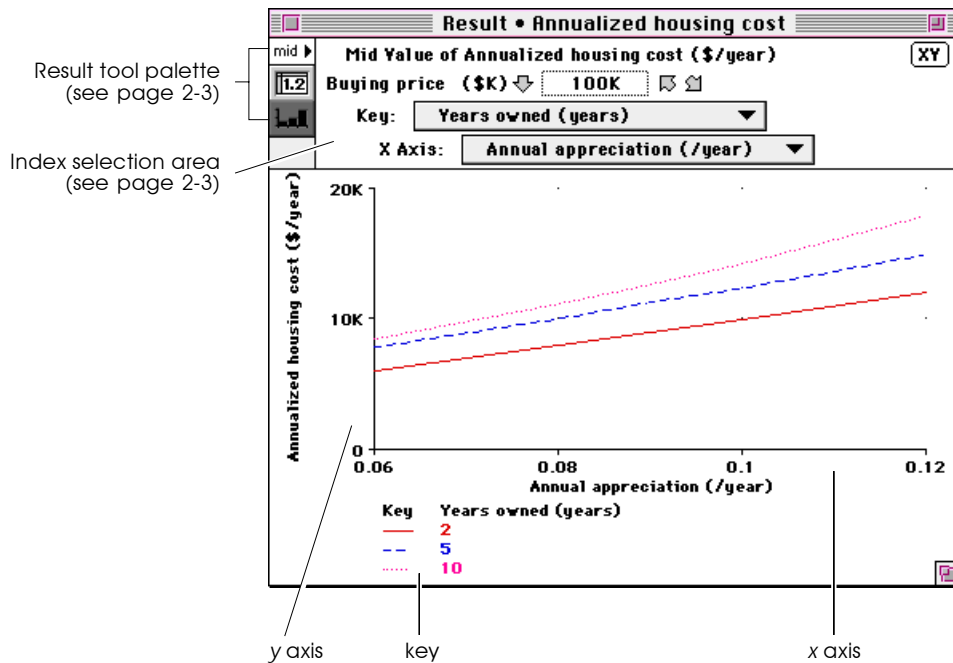
1. Select the row(s), column(s), or cell that you wish to format.
2. Choose **Number Format** from the **Result** menu.

See Section 7.6, "Number Format dialog box" for details on the number format options.

## 2.3 Viewing a result as a graph

**Displaying a graph** If a table is displayed, change it to the corresponding graph by clicking on the Graph View button ()

**Display of values** A *graph* displays the values from an array.



The vertical *y* axis shows the variable's values.

The display options depend on the number of dimensions in the variable.

### One dimension

The values of the dimension are shown horizontally, along the *x* axis (there are no options).

### Two or more dimensions

You can choose which index is displayed along the  $x$  axis using the  $x$  axis popup menu, and which index produces multiple curves using the key index popup menu.

The *key* shows the value of the index variable that corresponds to each curve, indicated by pattern or color.

Select specific values for the third or higher dimensions using the index navigation buttons.

### Changing graph ranges and styles

You can override the default ranges and styles for a graph, or you can change the default for all graphs, using the Graph Setup dialog box.

To display the Graph Setup dialog box, do one of the following:

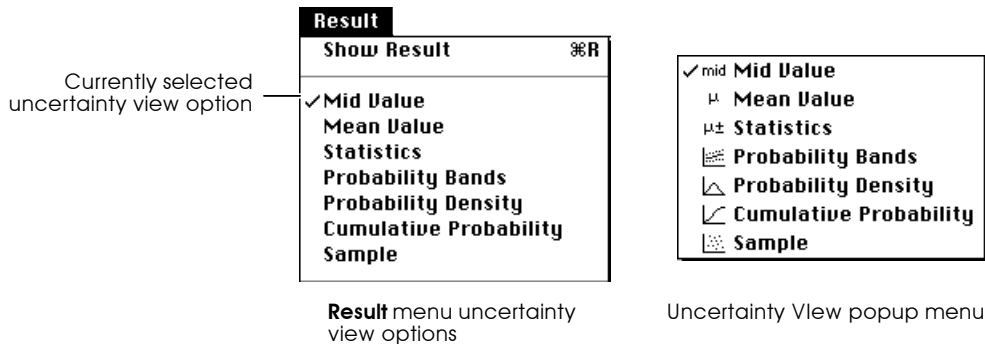
- Select **Graph Setup** from the **Result** menu.
- Double-click anywhere on a graph in the Result window.

See Section 7.1, “Graph Setup dialog box” for details on the Graph Setup options.

## 2.4 Uncertainty view options

The value of a variable in an Analytica model can be either *certain* (deterministic) or *uncertain* (probabilistic). An uncertain value can be viewed in several different ways. Select the uncertainty view in either:

- the **Result** menu
- the Uncertainty View popup menu (top left corner of the Result window).



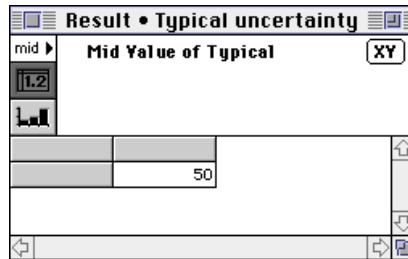
The check mark indicates the currently selected uncertainty view option. If the active window is not a Result window, selecting an uncertainty view option opens a Result window for the selected variable.

The uncertainty view options are described briefly here. For more information on them, see their entries in the Glossary and consult any standard statistics textbook.

The following examples use the variable *Typical Uncertainty*, which is defined as a normal distribution having a mean of 50 and a standard deviation of 30.

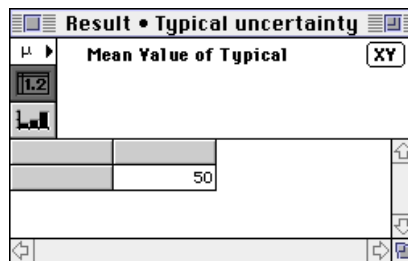
## Mid Value

The mid value is the deterministic value, computed by holding probability distributions at their median values. This value is computed very quickly compared to the uncertainty values. The mid value is the only option available for a certain (nonprobabilistic) variable.



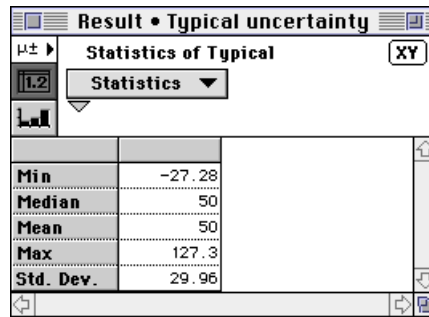
## Mean Value

The mean value is an estimate of the expected value of the uncertain value. For a symmetrical distribution, such as a normal distribution, the mean is the same as the median (mid) value.



## Statistics

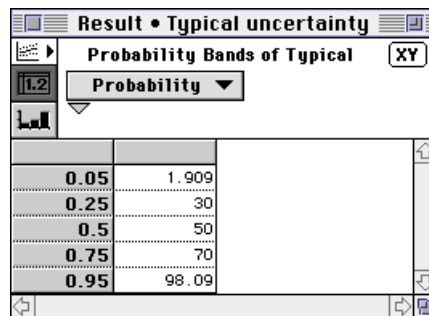
Statistics for the uncertain value, such as mean and standard deviation, are provided in a table. Select the statistics to be calculated using the Uncertainty Setup dialog (see “Statistics option” on page 13-16).



Statistics of Typical	
Min	-27.28
Median	50
Mean	50
Max	127.3
Std. Dev.	29.96

## Probability Bands

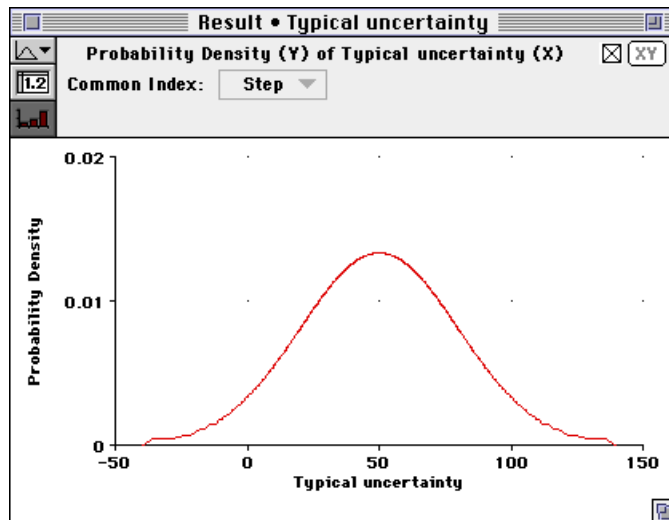
Probability bands are specified at given percentile values. Select the probability bands you wish to show using the Uncertainty Setup dialog (see “Probability Bands option” on page 13-17).



Probability Bands of Typical	
0.05	1.909
0.25	30
0.5	50
0.75	70
0.95	98.09

## Probability Density or Probability Mass

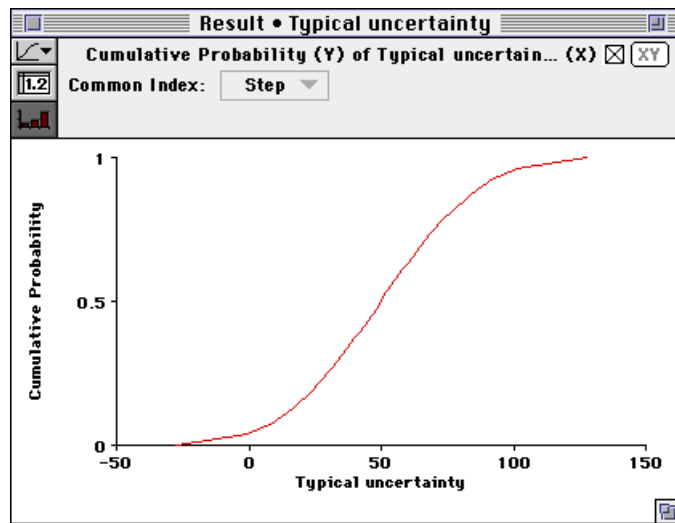
If the quantity is a continuous probability distribution, Analytica displays a probability density function. The horizontal ( $x$ -) axis plots possible values of the uncertain quantity. The height of the curve (probability density) is proportional to the likelihood that the quantity will have the  $x$  value. The highest point on the curve is the most likely value (the mode). Where the curve is at zero height or invisible, there is zero probability that the quantity will have that value. (For a discrete probability distribution, Analytica graphs the probability mass.)



The “common index” of Step is a counter to relate the variable’s values to the probability density; at the first and last value of Step, the probability density is zero.

## Cumulative Probability

The cumulative probability distribution plots the possible values of the uncertainty quantity along the horizontal ( $x$ -) axis. The height of the graph at each value of  $x$  shows the probability that the quantity will be less than or equal to that  $x$  value. The cumulative distribution ranges from a probability of 0 on the left to probability of 1.0 on the right, without decreasing. The steeper the curve, the more likely the quantity will have a value in that region.



The “common index” of Step is a counter to relate the variable’s values to the cumulative probability; at the first value of Step, the cumulative probability is zero and at the last value of Step, the cumulative probability is 1.0.

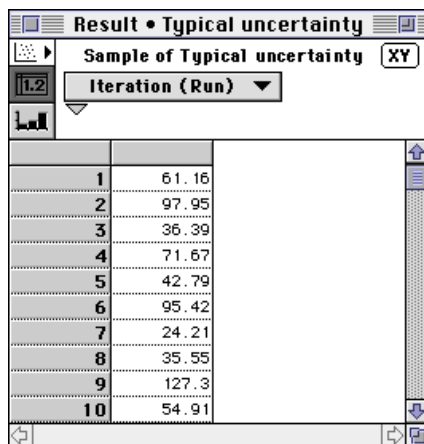
## Sample

A sample is an array of the random sample of values generated by the sampling process. The sample is the underlying representation for an uncertain quantity.

All other representations of uncertainty are estimated from the sample. The precision of the estimates depends on the sample



size and the sampling method (see Appendix D for selecting the sample size and Section 13.6 for setting the sample size).

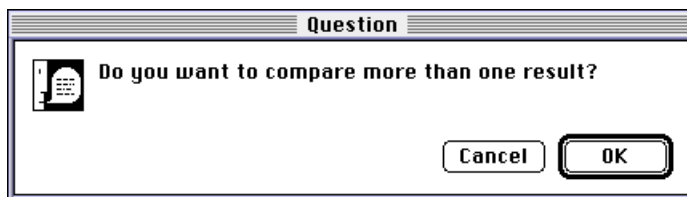


The screenshot shows a window titled "Result • Typical uncertainty". Inside, there is a section labeled "Sample of Typical uncertainty" with a sub-label "Iteration (Run)". Below this is a table with 10 rows and 2 columns. The first column contains iteration numbers from 1 to 10, and the second column contains numerical values. The values are: 61.16, 97.95, 36.39, 71.67, 42.79, 95.42, 24.21, 35.55, 127.3, and 54.91.

Iteration	Value
1	61.16
2	97.95
3	36.39
4	71.67
5	42.79
6	95.42
7	24.21
8	35.55
9	127.3
10	54.91

## 2.5 Comparing results

To directly compare the values of two or more variables in one table or graph, select the variables and open a Result window (see page 2-2). A dialog box asks for confirmation.



Analytica creates a new node with a default title and displays the values in one table or graph.

---



# Analyzing Model Behavior

---

A potent source of insight into a model is examining the behavior of its outputs as you systematically vary one or more of its inputs. This technique is called *model behavior analysis*. Each input that you vary systematically is called a *parameter*, and so this technique is also known as *parametric analysis*. Analytica makes it simple to analyze model behavior in this way. All you have to do is to assign a list of alternative values to each input parameter. When you view the result of any output, Analytica computes and displays a table or graph showing how the output values vary for all combinations of the values for each input.

This chapter describes how to select variables as parameters, how to specify alternative values for the parameters, and how to examine the results.

## 3.1 Varying input parameters

The first step in analyzing model behavior is to select one or more input variables as parameters and to assign each parameter a list of possible values.

### Which inputs to vary

You can vary any numerical input variable of your model, including decisions and chance variables. Often you will want to vary each decision variable to see which value gives the best results according to the objectives. You may also want to vary some chance variables to see how they affect the results. It is often best to look first at the decision or chance variables that you expect to have the largest effect on the model outputs. In complicated models, you may want to start with an importance analysis, to

identify which chance variables are likely to be most important. (See Chapter 16, “Analyzing Uncertainty and Sensitivity.”) You can then select the most important variables as the parameters to vary to analyze model behavior.

## How many values to assign

Usually it is best to assign a list of three alternative values to each parameter, a low, medium, and high value. In some cases, two values may be sufficient. If you have a special interest in a particular parameter (for example, if you suspect it may have a strongly nonlinear effect) you may want to assign more than three values to examine in more detail the model behavior as the parameter varies. Naturally, the computation time increases with the number of values.

## Creating a list

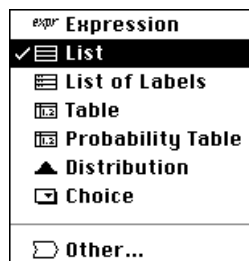
To create a list of values for a variable, change its definition following these steps:

1. Select the variable by clicking on its node in the influence diagram.

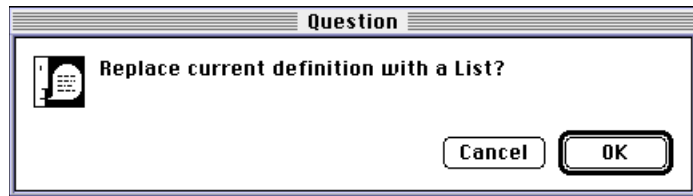


2. Display the variable's definition by clicking on the Definition button in the floating palette.

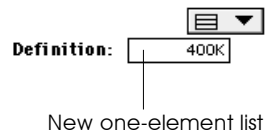
3. Press the Expressions popup menu above the definition and select the **List** option. (Do *not* select the **List of Labels** option.)



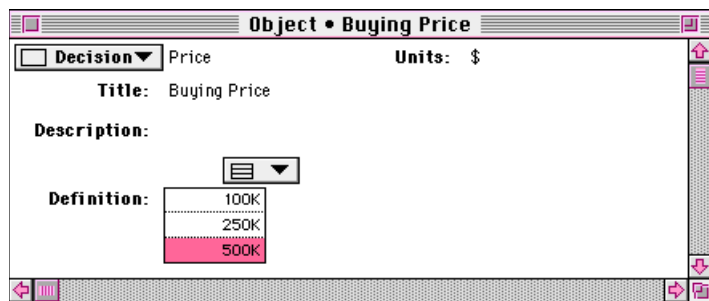
4. A dialog box asks for confirmation. Click on “OK”.



Analytica displays a list with one element, containing the old definition of the variable.



5. Select the element by clicking on it.
6. Type in the lowest value for the variable.
7. Press *return* and type in the next value.
8. Repeat step 7 until you have all the values you want.



Note that after you have entered two or more values into a list, the next item receives a default value. For example, if the last two values are 10 and 20, Analytica offers 30 as the next value.

For details on how to edit a list, see “Editing a list” on page 11-15.

If you want to create a list with a large number of evenly spaced values, use the `Sequence()` function (see page 12-5).

## How many inputs to vary

Typically you should start a model behavior analysis by varying just one input variable, the one you expect to be most important. Vary additional variables one at a time, in order of their expected importance. If a variable turns out to have little effect, you may restore it to its original value or probability distribution. If you have many inputs whose effects on model behavior you would like to explore, vary just a few at a time, rather than trying to vary them all simultaneously.

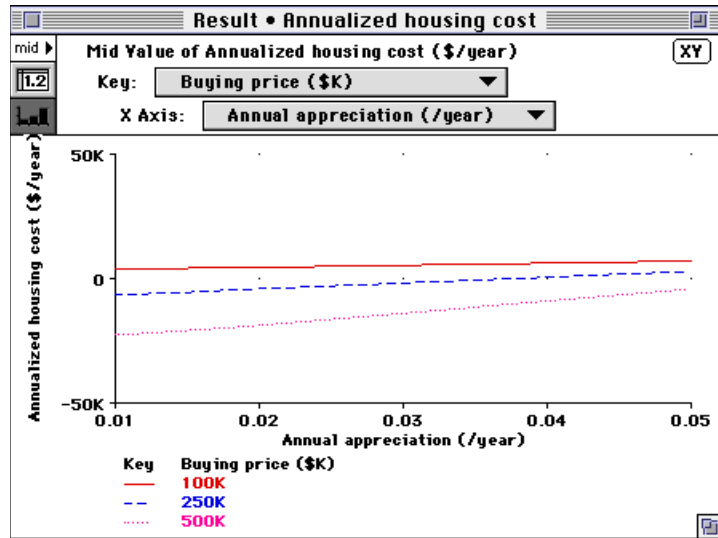
Each parameter that you vary becomes a new dimension of your output result array. The computation time and memory needed increase roughly exponentially as you add parameters. Moreover, you may find it hard to interpret an array with more than three or four dimensions. Remember that the goal is to obtain insight into what affects the model behavior and how.

## 3.2 Analyzing model behavior results

Once you have assigned a list to one or more inputs, you can examine their effect by viewing the result on an output variable. If your model has an objective, you might start by looking at that variable.

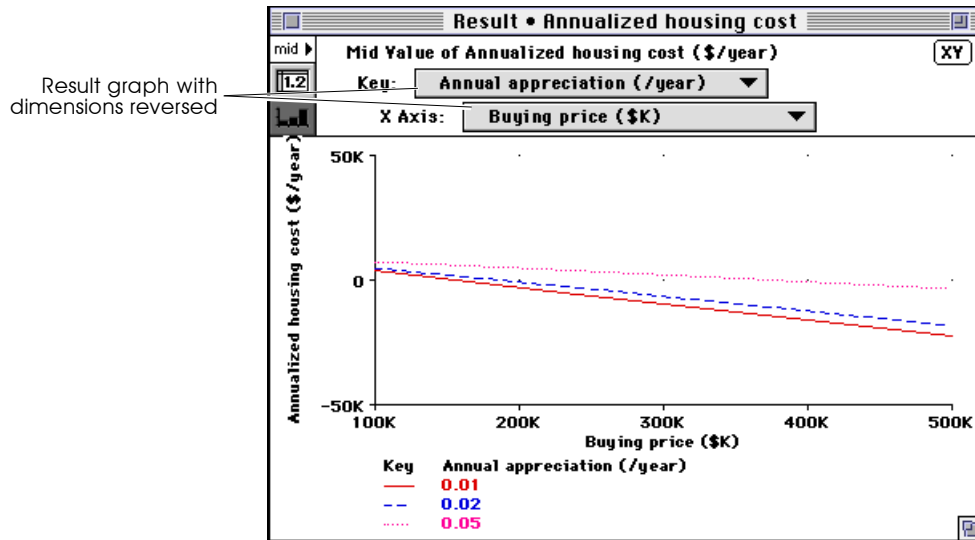
1. Select the variable you wish to view by clicking on its node in the diagram.
2. View the result by clicking on the result button in the tool palette. The result displays as a table or graph.





The result is an array with a dimension for each input parameter that you have varied (in this example, *Buying price* and *Annual appreciation*). If an input parameter does not appear as a dimension of the result, it implies that the result variable does not depend on the input. The result may also have other dimensions that are not input parameters you have varied, for example, *Time* for a dynamic model.

It is generally easiest to look first at the result graph to see the model's general behavior. You need to look only at the result table if you want to see the precise numerical values. If you are varying more than one input parameter, try rearranging the dimensions (see “Index selection area” on page 2-3) to get additional insights into model behavior.



## Understanding unexpected behavior

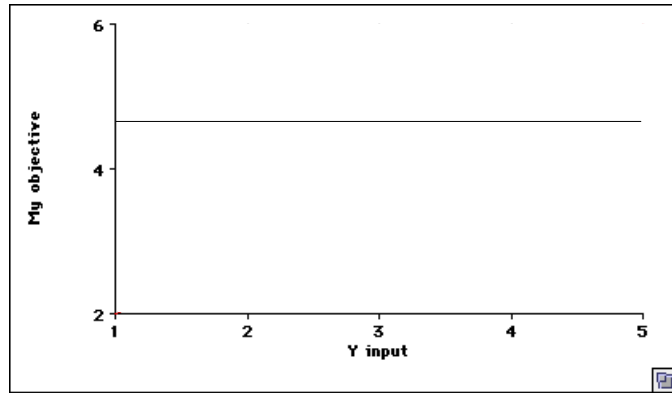
If you find the model's behavior unexpected or inexplicable, you may want to look more deeply into how the behavior arises. An easy way to do this is simply to look at the results for other variables between the input(s) and the output(s) in which you're interested. You can work forwards from an input towards the output, or backwards from the output towards the inputs. Look at the behavior of each intermediate variable, and see if you can understand why the inputs affect it the way they do.

Typically, the reason for unexpected behavior will quickly become clear to you. It may be that some intermediate relationship has an effect different from what you expected. It may turn out that there is an error in a definition. In either case, this kind of exploration can be very revealing about the model. You may end up improving the model or gaining a deeper understanding of the system it represents.

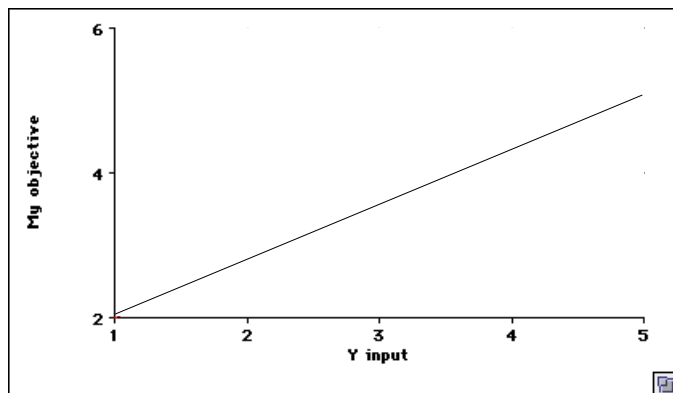
## Understanding model behavior

By examining result graphs, you can learn if each input affects the output, if the effect is linear or non-linear, and if there are interactions among inputs in their effect on the output. Below are some typical graph patterns and their qualitative interpretations.

- A horizontal line shows that changes in the input over the specified range have no effect on the output.



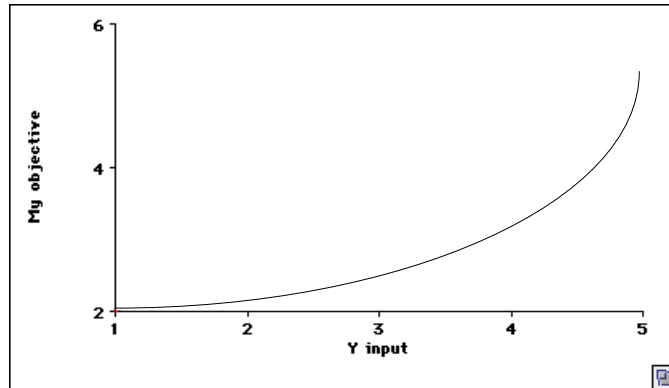
- A straight line shows that the output depends linearly on the input—provided that you have specified more than two different values for the input.





- A bent or curved line shows that there is a nonlinear dependence.

(Note that if you have only two values for the input, the graph will be a straight line even if there is a nonlinear dependence).



---

# 4

# Creating and Editing a Model

---

This chapter introduces you to the elements for building influence diagrams. It describes how to create a new model and save changes, how to create and edit nodes, and how to draw arrows and make aliases.

## 4.1 Creating and saving a model

You can create new models in Analytica, as well as edit your models and save the changes that you make.

### Creating a new model



To create a model, do one of the following:

- Double-click on the Analytica application in the Finder to open it, and click on **New** in the Open Model dialog box.
- If a model is already open, close it first, then select **New Model** from the **File** menu, or press ⌘-N.

### Saving a model

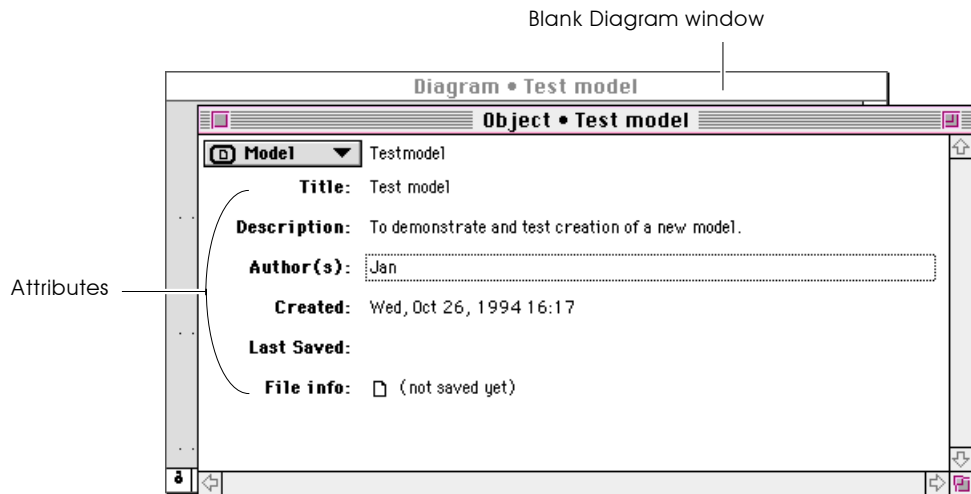
To save changes to the model, select the **Save** command from the **File** menu (⌘-S).

To save the model file under a new name, select the **Save As** or **Save a Copy In** command from the **File** menu. After you use the **Save As** command, selecting the **Save** command will save the model with the new name. After you use the **Save a Copy In** command, selecting the **Save** command will save the model with its original name.

## 4.2 The model Object window

The model Object window shows information about the model, such as the author(s), and creation and save dates; it also includes space for a description of the model's purpose.

When you create a model, an Object window is displayed for the new model, initially untitled, with the fields shown in the following figure. Enter information as appropriate.



See the Glossary for descriptions of the attributes.

After entering information into the model Object window, bring the Diagram window to the top in any of three ways:



- Click on the Parent Diagram button.
- Click anywhere in the Diagram window behind the Object window.
- Click on the Object window's Close box.

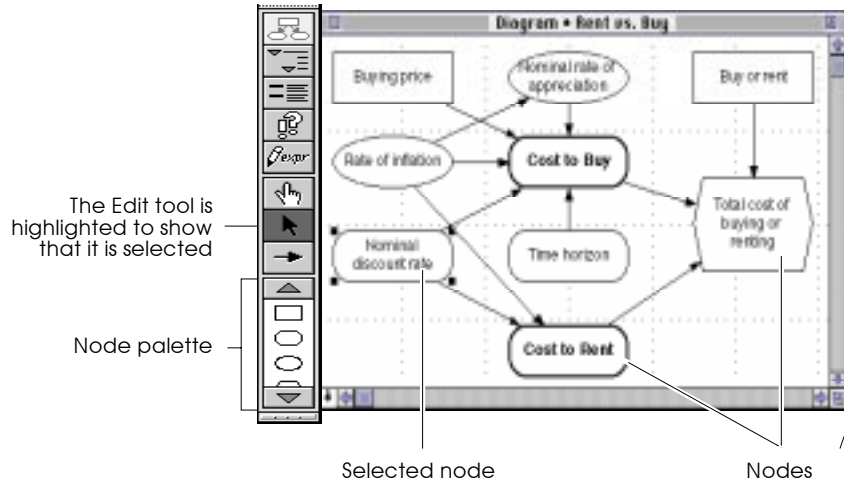
You can now draw a diagram for the new model (see Section 4.3).

## 4.3 Creating and editing nodes in a diagram

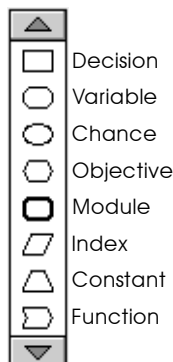
To create new nodes, or move or modify existing nodes, the Edit tool must be selected.

When a Diagram window for a new model is first opened, the Edit tool is selected by default. When a Diagram window for an existing model is first opened, the Browse tool is selected (see “Browsing the window” on page 1-5), so you can examine, but not change, the diagram.

To begin editing a diagram, click on the Edit tool () , if it is not selected.



When you are in Edit mode or Arrow mode, the scrollable *node palette* appears at the bottom of the tool palette.



The node palette is displayed when either the Edit tool or Arrow tool is selected.

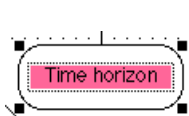
For details about the node classes in the node palette, see “Recognizing nodes”, Section 1.5.

## Creating a node

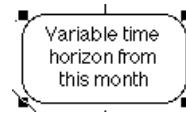
To create a new node, press on the appropriate icon in the node palette, then drag the outline into the diagram. After placing the node in the diagram, use the keyboard to enter its title.

## Editing a node title

To edit the title of a node, first select the node, then click on that node’s text field. Pause between the mouse click to select the node and the mouse click to select the text; otherwise, your action may be interpreted as a double-click, opening the node’s Object window. The node’s appearance changes to match the illustration of the node on the left, below. When you have finished typing, press *enter* to accept the title change and resize the node to fit the text.



You can edit the title when the node looks like this

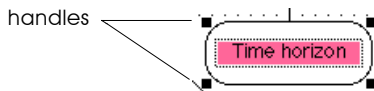


The node is resized to fit the text

When a node’s title is first created, its *identifier* is created from the first 20 characters of the title (underscores replace spaces). The node’s identifier is the name used to refer to this node in the mathematical definition of other nodes. The node keeps this identifier until it is explicitly edited, unless the Change Identifier preference is set (see page 4-20).

## Selecting nodes

To select a node, single-click on it. Handles indicate that you have selected the node. To deselect a selected node, click anywhere outside of it.



To select or deselect multiple nodes, shift-click. You can also select a group of nodes by dragging a rectangle around them. Move the cursor to a corner of the diagram (not in a node), press the mouse button, and drag the mouse to draw a rectangle. When you release the button, all the nodes completely inside the rectangle are selected.

**Working with nodes** You can manipulate the nodes in a diagram in a variety of ways.

#### **Moving a node**

To move a node, press the mouse while the cursor is inside the node but not on a handle, then drag the node.

You can also move a selected node using the arrow keys (up, down, left, right).

#### **Moving a node into a module**

To move a node into a module in the diagram, drag the node onto the module until the module becomes highlighted.

When you release the mouse button, the node goes into the module.

Alternatively, double-click on the module to open its diagram window. Move the module diagram window so both it and the node to be moved are visible. Then drag the node onto the module diagram window.

#### **Changing the size of a node**

To change the size of a node, drag a handle until the node is the size you desire.

By default, a node is resized keeping the center in place (that is, all four corners expand or contract). This helps to keep nodes on the grid and lined up with each other. To turn off the default so one corner at a time can be resized, uncheck the

**Resize Centered** option in the **Diagram** menu.

### Deleting a node

To delete a node, first select it. Then, choose **Clear** from the **Edit** menu, or press the *delete* key. You will be prompted to confirm your intentions before the node is deleted.

### Copying and pasting nodes

To create nodes that have substantial information in common, you can use the traditional Macintosh Copy and Paste commands. Initially, the copies are identical except for their identifiers (which have numbers appended to them to make them unique).

**Cutting a node**     Select **Cut** from the **Edit** menu (⌘-X).

**Copying a node**     Select **Copy** from the **Edit** menu (⌘-C).

**Pasting a node**     Select **Paste** from the **Edit** menu (⌘-V).

### Duplicating nodes

To create two sets of nodes that have substantial information in common, create the first set. Select the nodes, then choose **Duplicate Nodes** from the **Edit** menu. This is equivalent to using Copy and Paste, without writing to the clipboard.

## Aligning to the grid


When the grid is on (the default), each node that you create or move is centered on a grid point. This default makes it easier for you to position nodes so that arrows are exactly horizontal or vertical when nodes are aligned vertically or horizontally.

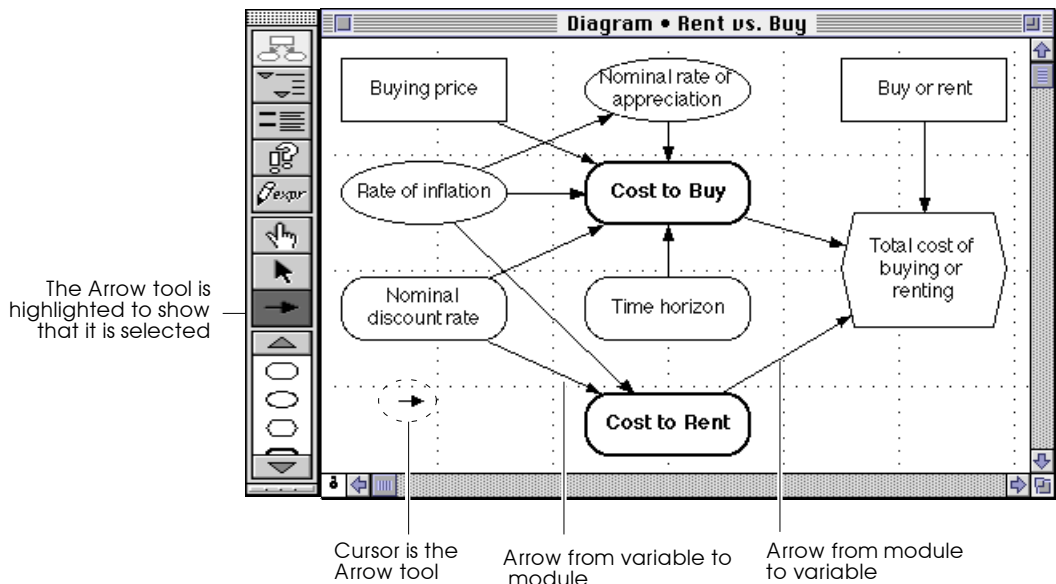
To recenter nodes, select **Align to Grid** from the **Diagram** menu (⌘-J).

To turn the grid off in edit mode, select **Turn Grid Off** from the **Diagram** menu. When the grid is off in edit mode, the grid is still visible; you can move the nodes pixel by pixel.

## 4.4 Drawing arrows in a diagram window

Use the Arrow tool to draw or remove arrows (influences) between variable nodes.

**Arrow tool** The Arrow tool must be selected before you can manipulate arrows. To select the Arrow tool, click on the arrow button (  ) in the floating tool palette.



### Arrows and nodes Arrow from variable node to variable node

Indicates that the target variable depends on the origin variable.

### Arrow from variable node to module node

Indicates that at least one variable in the target module depends on the origin variable.



**Arrow from module node to variable node**

Indicates that the target variable depends on at least one variable in the origin module.

**Arrow from module node to module node**

Indicates that the target module contains at least one variable that depends on at least one variable in the origin module.

**Double-headed arrow between module nodes**


Indicates that each module contains at least one variable that depends on at least one variable in the other module.

**Small arrowhead to the right or left of a variable node**

Indicates that the variable has a remote input or output—a variable that is not inside the displayed variable's module (see page 1-8).



## Creating and removing arrows


To draw an arrow, first be sure the arrow tool (  ) is selected.

1. Drag from the origin node (it becomes highlighted) to the destination node (which also becomes highlighted).
2. When you release the mouse button, the arrow is drawn.

To draw multiple arrows to a single destination node, select all the origin nodes. Then drag from any origin node to the destination node.

To remove an arrow, do one of the following:

- Select the arrow, then press the *backspace* or *delete* key.
- Repeat the process of drawing an arrow from the origin node to the destination node.

**Note:** 

An arrow is also drawn whenever the identifier of a variable is added to the definition of a variable (see “Creating or editing a definition” on page 8-1).

## Model changes when creating an arrow

Creating an arrow between two nodes changes the model. The change depends on the classes of the two nodes.

### Arrow between two variable nodes

When you draw an arrow from a variable node *A* to another variable node *B*, *A* becomes an input of *B*. You can then use this input in creating or editing the definition of *B*. (See Chapter 8, “Creating and Editing Definitions.”)

### Arrow between a variable node and a module node

When you draw an arrow from a variable into a module, or from a module to a variable, Analytica creates an alias of the variable inside the module (see “Creating alias nodes” on page 4-12). You can then open the module and draw arrows between the alias and other variables in the module.

### Arrow between two module nodes

When you draw an arrow from one module node to another, Analytica creates a new variable node in the first module and an alias of that variable in the second module (see “Creating alias nodes” on page 4-12). You can then open each module and draw arrows between the new nodes and other variables in the module.

## Model changes when deleting an arrow

If *B* already has a definition that includes *A* and you delete the arrow from *A* to *B*, Analytica removes *A* from *B*’s definition. For example, if *A* is an index and *B* is defined as a table, Analytica removes *A* as an index of *B*.

In some cases, removing *A* does not leave a valid definition for *B* (for example, when *A* is part of a mathematical expression such as  $1/A$ ). Under these circumstances, *A* is replaced with the keyword

Expr and the entire expression is surrounded with `FunctionOf()`. This notation indicates that the definition is invalid and must be edited.

**Cyclic dependency** A *cyclic dependency* occurs when a variable depends on itself directly or indirectly so that the arrows form a directed circular path.

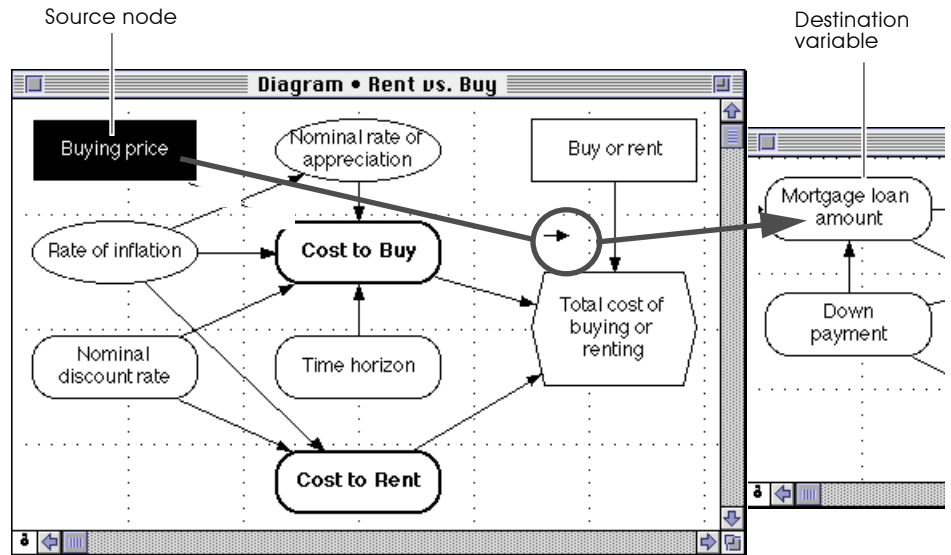
A cyclic dependency is permitted only in a dynamic model (see Chapter 17, “Modeling Changes over Time”), provided that the variable depends on its value in an earlier time period.

## 4.5 Drawing arrows between variables in different modules

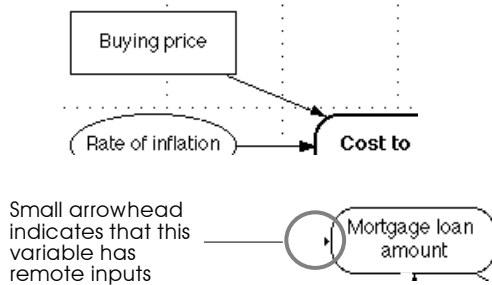
There are two direct methods and two indirect methods for drawing an arrow between two variables in different modules. The following examples demonstrate the direct methods of drawing an arrow from the variable *Buying price* to the variable *Mortgage loan amount* in another module (see the following figure).

### Drawing arrows across windows

1. Open both module Diagram windows and bring to the top the module Diagram window containing the origin variable, *Buying price*.
2. Position the Diagram windows so the variable *Mortgage loan amount* is exposed in the window underneath.
3. Draw an arrow from the origin variable (*Buying price*) to the second variable (*Mortgage loan amount*).



**Result:** An arrow points from *Buying price* to the *Cost to Buy* module; a small arrowhead points into *Mortgage loan amount* to indicate a source node is in a different diagram.



## Moving, drawing, and moving back

1. Select *Mortgage loan amount*, then choose **Move Into Parent** from the **Diagram** menu to move the variable into the parent diagram.
2. Draw an arrow from *Buying price* to *Mortgage loan amount*.

3. Move *Mortgage loan amount* back into its module by dragging it onto the *Cost to Buy* module node.

### Indirectly drawing an arrow

The two indirect methods of drawing an arrow between two variables in different modules are:

- Edit the definition of the target variable, entering the identifier of the input variable directly (see “Creating or editing a definition” on page 8-1). The arrow appears when the definition is accepted.
- Use an alias node (see Section 4.6).

## 4.6 Creating alias nodes

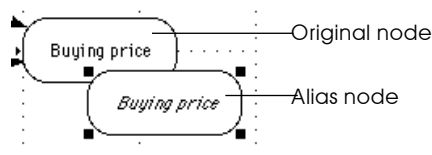
An *alias* is a copy of a node, referring to the same variable or module as the original node. You can use an alias node to display the same variable in more than one module diagram. For example, often the inputs to a variable are in one module, while its outputs are in other modules. To display an input variable’s node in the modules containing the outputs, create an alias in each of those modules.

A variable or module can have only a single original node. You can create an unlimited number of alias nodes for any original node.

Create an alias in any of the three following ways:

### Use the Make Alias command

Select the original node. Then choose the **Make Alias** option from the **Object** menu (⌘-M). The alias node appears next to the original node. You can then move it into another module by dragging it.



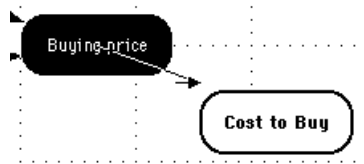
Use this method to make an alias of a module, if you want to show a module node in more than one diagram.

## Draw arrow between variable and module

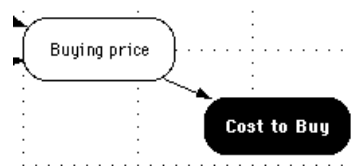
Draw an arrow from the original node to a module node, or from the module node to the original node. An alias for the original node appears in the module.

### Example:

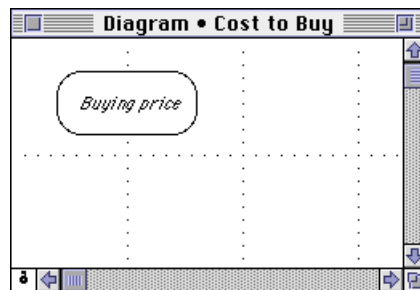
Draw an arrow from a variable (*Buying price*) to a module (*Cost to Buy*).



An arrow is displayed from the *Buying price* variable to the *Cost to Buy* module.

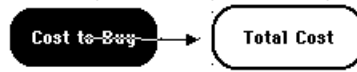


The new alias node appears in the diagram for the *Cost to Buy* module.

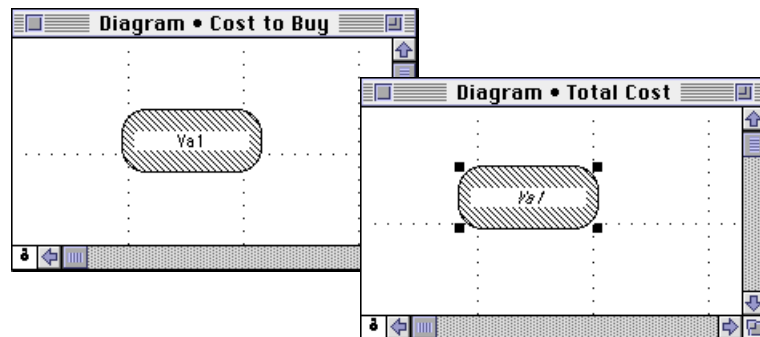


## Draw arrow between two modules

Draw an arrow from one module node (*Cost to Buy*) to another module node (*Total Cost*).



Analytica creates a new variable node with a default title, such as *Va1*, in the first module, and creates an alias of *Va1* in the second module.



## 4.7 Using an alias node

An alias looks and behaves similar to the original node that it is derived from, except that its title is in italics. (The Macintosh Finder also uses italics to distinguish file and folder aliases from the original file or folder icon.)

**Note:** 

An alias of a module does not display any input or output arrows.

You can treat an alias node just as if it were the original node. Click once on an alias to select it (for example, to display its

result). Double-click on an alias to open the Object window for the original node.

To use a variable with an alias as an input to another node, draw an arrow either from the original node or from its alias.

To create a new input to a variable with an alias, draw an arrow either to the variable or to its alias. An arrow to an alias of a variable creates a corresponding arrow to its original node in its diagram, if the original node is in the same module as the new input or an alias of the new input.

### Modifying an alias node

When you create an alias node, it looks just like its original node, including its node shape, color, label font, and icons. The only difference is that the title is in italics.

If you edit the title of an alias, the title of the original node changes to match the alias. Conversely, if you edit the title of the original node, the alias's title changes to match the original.

To change the appearance of an alias node alone, use the **Set Node Style** option from the **Diagram** menu (see “Node Style dialog box” on page 6-12). If you use the Node Style dialog box to change the appearance of an alias node, its original node does not change. Similarly, using the Node Style dialog box to change the appearance of an original node does not affect any of its previously created aliases.

## 4.8 Editing an attribute of a node

You can edit a user-modifiable attribute either in the Attribute panel (see Section 1.8) or in the Object window (see Section 1.7). To change a node's class, see Section 4.9.

To edit an attribute:

1. If in the attribute panel, select the attribute in the attribute popup menu.



2. Click in the attribute field. If a gray outline appears around the attribute and the cursor is blinking, the attribute is user-modifiable.
3. Edit the attribute using the standard text-editing methods.

The edited text is stored when you click anywhere outside the attribute field, or when you press *enter*.

### Attribute changes

Any changes to attributes propagate to all other displayed windows. For example, if you change the title of an object, the new title is displayed in that object's diagram. If you change the definition of a variable, the arrows are redrawn to reflect changes in dependencies.

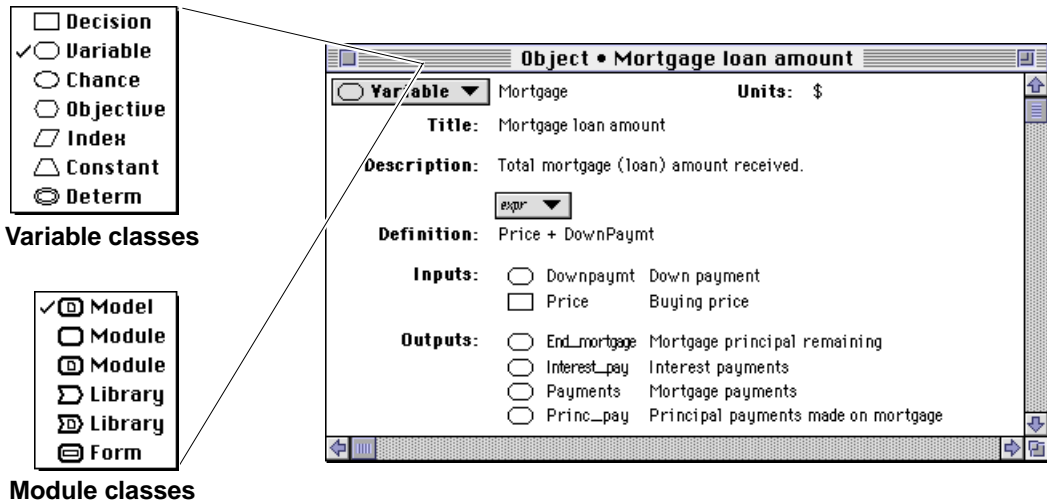
### Cancel and Undo

While you are editing an attribute, you can cancel and revert to the previous value at any time by pressing the *esc* (escape) key. If you have finished entering the value of the attribute, but now want to revert to the previous value of the attribute, select **Undo** from the **Edit** menu (⌘-Z).


## 4.9 Changing the class of a node

Use the Class popup menu to change the class of a node. This menu appears:

- In the top left corner of the Object window
- In the Attribute panel when **Class** is the selected attribute.



The contents of the Class popup menu depend on whether the node is a variable or a module (see the preceding figure).

**Note:**  You cannot change a variable into a module, or vice versa. You also cannot change a function into a variable or module, or vice versa.

To change a node's class, press on the Class popup menu and select another class.

The variable classes are described in Section 1.5. The module classes are described below.



### Model

A module or hierarchy of linked modules that you work on during a session with Analytica. A model is saved in a file (an Analytica document) between sessions. Only a model saves preferences (see Section 4.10) and uncertainty options (see Section 13.6).



### Module

A collection of nodes that are displayed in a single diagram. A module is depicted on its parent diagram as a rounded node with a thick outline.



### Filed Module

A module whose contents are saved in a file separate from the model that contains it. A filed module can be shared among several models, without having to make a copy for each model. See Section 19.6.



### Library

A module that contains functions and/or variables. User libraries appear in the **Definition** menu below the system function libraries, giving easy access to their contents. See Section 20.5.



### Filed Library

A library whose contents are saved in a file separate from the model that contains it. A filed library can be shared among several models, without having to make a copy for each model. See Section 19.6.



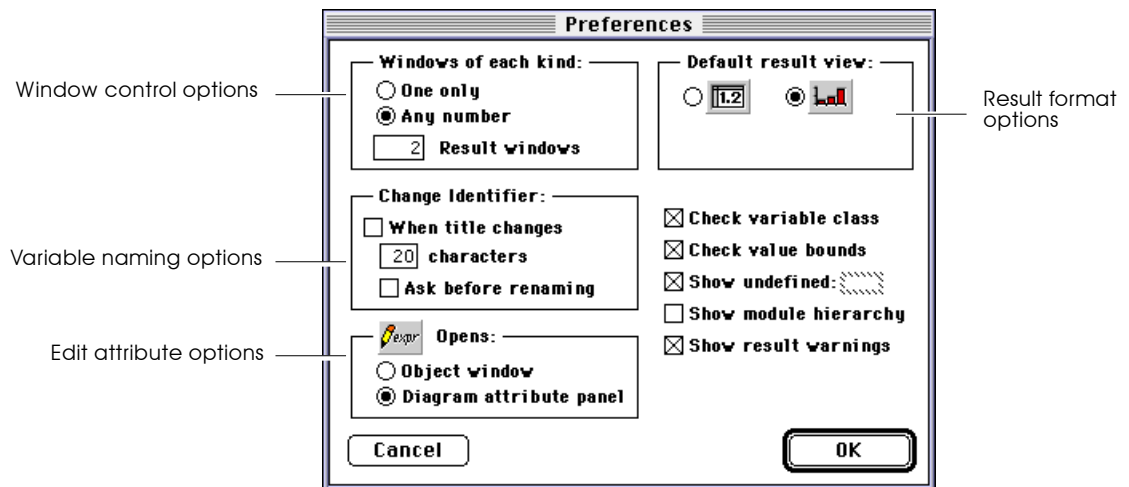
### Form

A module that creates an input node alias or an output node alias when you draw an arrow from a node to the form. See Chapter 9, “Creating Models to be Used by Others.”

## 4.10 Preferences dialog box

Use the Preferences dialog box to inspect and set a variety of preferences for the operation of Analytica. All preference settings are saved with a model of class Model.

To display the Preferences dialog box, select **Preferences** from the **Edit** menu.



### Windows of each kind

Use the options in this box to control how many windows of various kinds are displayed at once (see Section 19.9).

#### One only

Check this box to close an existing window (if there is one) whenever you open a new window, unless you press the *command* key ( $\text{⌘}$ ) as you create the new window.

#### Any number

Check this box to keep all windows open until you explicitly close them.

### Result windows

Enter a value in this field to indicate the number of Result windows that you can keep open simultaneously. The default (and minimum) number is 2; the maximum number is 20.

**Change identifier** Use the options in this box to control the changing of identifiers. See “Creating and editing nodes in a diagram” on page 4-3 for a description of how identifiers are initially assigned.

### When title changes

Check this box to change a variable’s identifier whenever you change its title. Analytica uses up to the number of specified characters (20 by default, range from 2 to 20), replacing spaces and returns with an underscore character (\_).

If the box is not checked, the identifier is changed only when you explicitly edit it.

### Ask before renaming

Check this box to see a confirmation dialog box before automatic changing of a variable’s identifier.



### opens

Use the options in this box to specify the window that displays when you select the edit definition button (⌘ -E). When you are prompted (for example, via an error message) to edit a definition, and you click on OK, this preference setting determines the window to display.

### Object window

Select this option to open the Object window and select the definition text.

### Diagram attribute panel

Select this option to open the Attribute panel on the appropriate Diagram window and select the definition text.

**Default result view** Use the options in this box to control how data appear in Result windows the first time a result is calculated for a node (see “The Result window”, Section 2.1).



Select this option to display a table.



Select this option to display a graph.

**Checkboxes** Use these checkboxes to control various aspects of how Analytica looks and behaves.

### Check variable class

If checked, a warning displays for the following inconsistencies between a variable’s class and definition:

- A non-chance variable whose definition includes a probability distribution.
- A constant whose definition is dependent on any other variables. However, a constant defined as a table may have indexes as inputs.
- An index variable defined as a single value, array, or any function other than `Sequence ( )`.

### Check value bounds

If checked, the check attributes are computed. See “Checking the validity of a variable’s values” on page 8-12.

### Show undefined

If checked, nodes without a valid definition display with a cross-hatch pattern.



Node is filled with diagonal pattern: the definition is missing or is syntactically incorrect.

### Show module hierarchy

If checked, a bar at the top of each Diagram window indicates the hierarchy depth of the active module. See “Show module hierarchy preference” on page 19-1.

### Show result warnings

If checked, when a warning condition is encountered during result evaluation, evaluation is interrupted, and a warning message displays for action. If unchecked, when a warning condition is encountered during result evaluation, no warning message is displayed, and evaluation continues.

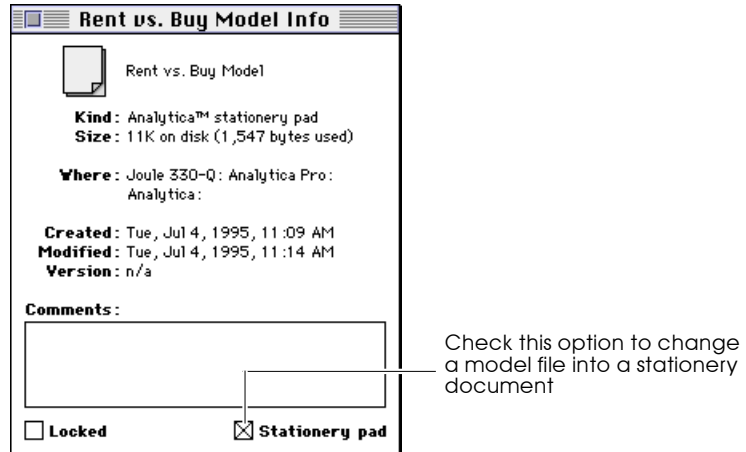
## 4.11 Using stationery

A *stationery document* is a template that can include preferences, diagram settings, default node settings, and nodes—anything you can have in an Analytica document. This is useful if you have nodes or preferences you use in many models.

**Creating stationery** To create a stationery document, follow these steps :

1. Create an untitled model in Analytica.
2. Change your preferences to the desired settings (including default graph setup and uncertainty preferences).
3. If you wish to create a template model, add any variables or modules to it.
4. Save the model, giving it a descriptive name, and quit Analytica.
5. In the Macintosh Finder, select the file containing the model, and select **Get Info** from the **File** menu to open that file’s Info window.
6. Click on the Stationery pad checkbox, and close the Info window. Your model file is now a stationery document.





To edit a stationery document uncheck the Stationery pad checkbox in the Info dialog, edit it and save it, then re-check the Stationery pad checkbox.

**Using stationery** To create a model that uses the preferences or the template model in your stationery document, open the stationery document as you would any other Analytica model. A copy of it appears, which you can set up for a specific model; then use the **Save As** command to rename it.





---



# Building Effective Models

---

Creating useful models is a challenging activity, even for experienced modelers; effective use of influence diagrams can make the process substantially easier and clearer. This chapter provides tips and guidelines from master modelers (including Newton and Einstein) on how to build a model that is effective—that focuses on what matters, and that is simple, clear, comprehensible, and correct. The key is to start simple and progressively refine and extend the model where tests of initial versions suggest it will be most important.

Most of the material in this chapter, unlike the other chapters in this *User Guide*, is not specific to Analytica. These guidelines are useful whether you are using Analytica, a spreadsheet, or any other modeling tool. However, Analytica makes it especially easy to follow these guidelines, using its hierarchical influence diagrams, uncertainty tools, and Intelligent Arrays.

These guidelines have been distilled from many years of experience by master modelers, using Analytica and a variety of other modeling software. However, they are general guidelines, not rules to be adhered to absolutely. We suggest you read this chapter early in your work with Analytica and revisit it from time to time as you gain experience.

## 5.1 Creating a model

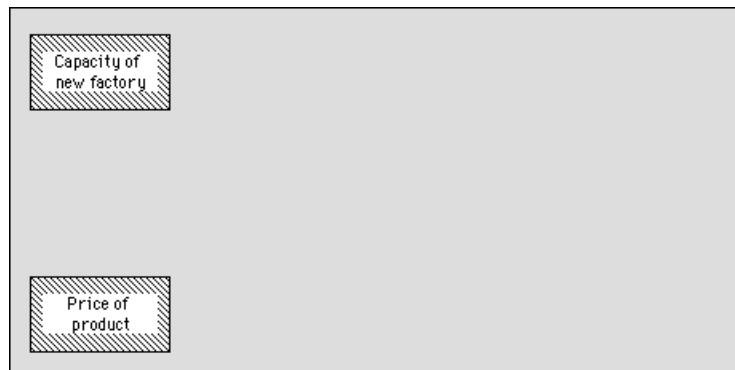
Below are general guidelines to help you build models that provide the greatest value with the least effort.

### Identify the decisions

The purpose of modeling is usually to help you (or your colleagues, organization, or clients) discover which decision options will best meet your (or their) objectives. You should aim, therefore, to include the decisions and objectives explicitly in your model.

First, identify the key decision variables. A decision variable is one that the decision maker can affect directly—which computer to buy, how much to bid on the contract, which medical treatment to choose, when to start construction, and so on. Occasionally, people want to build a model just for the sake of furthering understanding, without explicitly considering any decisions. Most often, however, the ultimate purpose is to make a better decision. In those cases, the decision variables are where you should start your model.

When starting a new influence diagram, put the decision variables—as rectangular nodes—on the left of the diagram window, leaving space for the rest of the influence diagram to the right.

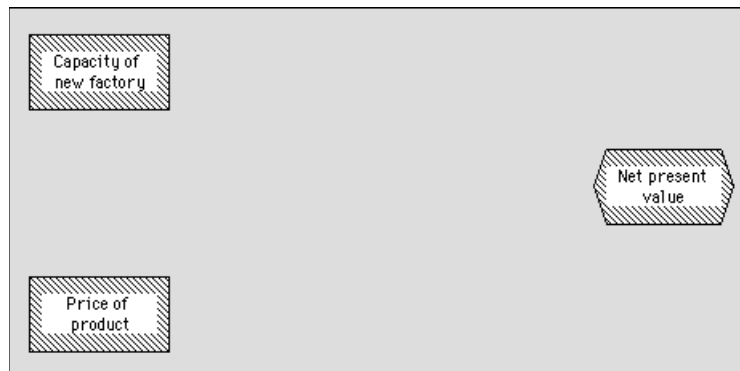


---

## Identify the objectives

What are the objectives of the decision maker? Sometimes the objective is simply to maximize expected monetary profit. More often there are a variety of other objectives, such as maximizing safety, convenience, reliability, social welfare, or environmental health, depending on the domain and the decision maker. Utility theory and multiattribute decision analysis provide an array of methods to help structure and quantify objectives in the form of utility. Whatever approach you take, it is important to represent the objectives in an explicit and quantifiable form if the objectives are to be the basis for recommending one decision option over another.

You should put the objective variable or variables (hexagonal nodes) on the right of the diagram window, leaving space between the right and left sides for the rest of the diagram to link the decisions to the objectives (see also page 6-4).



The most common mistake in specifying objectives is to select objectives that are too narrow, by concentrating on the most easily

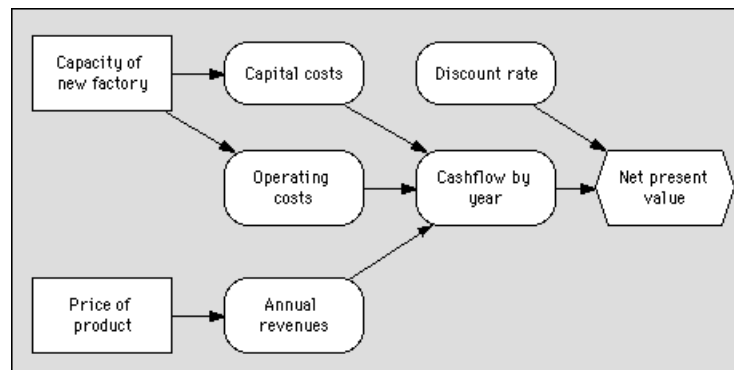
quantifiable objective—typically, near-term monetary costs—and to forget about the other, less tangible objectives. For example:

- When buying software you may want to consider the usability and reliability of different software packages, not just cost and performance.
- In pricing a product, you may want to consider the long-term effects of increased market share in developing new customers and markets and not just short-term revenues.
- In selecting a medical treatment, you may want to consider the quality of life if you survive the treatment, and not just the probability of survival.

For an excellent guide on how to identify and structure objectives, see *Value-Focused Thinking* by Ralph Keeney.

## Link the decisions to the objectives

The decisions and objectives are the starting and ending points of your model. Once you have identified them, you have reduced the diagram construction to the process of creating the links between the decisions and objectives, via intermediate variables. You may wish to work forward from the decisions, or backward from the objectives. Some people find it easiest to alternate, working inward from the left and the right until they can link everything up in the middle.



---

It helps to identify the decisions and objectives early during model construction, to maintain focus on what matters. There may be a bewildering variety of variables in the situation that may seem to be of potential relevance. But, you only need to worry about variables that influence how the decisions might affect the objectives. You can ignore any variable that has no effect on the objectives.

Focus on identifying the variables that make clear distinctions—variables whose interpretations won't change with time or viewer. Extra effort here will be repaid in model accuracy and cogency.

## **Move from the qualitative to the quantitative**

An influence diagram is a purely qualitative representation of a model. It shows the variables and their dependencies. It is usually best to draw in most or all of the first version of your model just as an influence diagram, or hierarchy of diagrams, before trying to quantify the values and relationships between the variables. In this way, you can concentrate on the essential qualitative issues of what variables to include, before having to worry about the details of how to quantify the relationships.

*“A theory should be as simple as possible, but no simpler.”  
Albert Einstein*

When the model is intended to reflect the views and knowledge of a group of people, it is especially valuable to start by drawing up influence diagrams as a group. A small group can sit around the computer screen; for a larger group, it is best if you have the means to project the image onto a large screen, so that the entire group can see and comment on the diagram as they create it. The ability to focus on the qualitative structure initially lets you involve early in the process participants who might not have the time or interest to be involved in the detailed quantitative analysis. With this approach, you can often obtain valuable insights and early buy-in to the modeling process from key people who would not otherwise be available.

Perhaps the most common mistake in modeling is to try to build a model that is too complicated or that is complicated in the wrong ways. Just because the situation you are modeling is complicated doesn't necessarily mean your model should be complicated. Every model is unavoidably a simplification of reality; otherwise it

would not be a model. The question is not whether your model should be a simplification, but rather how simple it should be. A large model requires more effort to build, takes longer to execute, is harder to test, and is more difficult to understand than a smaller model. And it may not even be more accurate.

### **Reuse and adapt existing models**

Building a new model from scratch can be a challenge. If you can find an existing model for a problem similar to the one you are now facing, it is usually much easier to start with the existing model and adapt it to the new application. In some cases, you may find parts or modules of existing models that you can extract and combine to address a new problem.

*“If I have seen further than [others] it is by standing upon the shoulders of Giants.”*  
*Sir Isaac Newton*

To find a suitable model to adapt, you can start by looking through the example models distributed with Analytica. If there is an Analytica users’ group in your own organization, it may collect a model library of classes of problems of interest to your organization.

### **Aim for clarity and insight**

The goal of building a model is to obtain clarity about the situation, about which decision options will best further your objectives, and why. If you are already clear about what decision to make, you don’t need to build a model, unless, perhaps, you are trying to clarify the situation and explain the recommended decisions for others. Either way, your goal is greater clarity. This goal is another reason to aim for simplicity. Large and complicated models are harder to understand and explain.

---

## 5.2 Testing and debugging a model

Even with Analytica, it is rare to create the first draft of a model without mistakes. For example, on your first try, definitions may not express what you really intended. It is important to test and evaluate your model to make sure it expresses what you have in mind. Analytica is designed specifically to make it as easy as possible to scrutinize model structures and dependencies, to explore model implications and behaviors, and to understand the reasons for them. Accordingly, it is relatively easy to debug models once you have identified potential problems.

### **Test as you build**

With Analytica, you can evaluate any variable once you have provided a definition for the variable and all the variables on which it depends, even if many other variables in the model remain to be defined. We recommend that you evaluate each variable as soon as you can, immediately after you have provided definitions for the relevant parts of the model. In this way, you'll discover problems as soon as possible after specifying the definitions that may have caused them. You can then try to identify the cause and fix the problem while the definitions are still fresh in your memory. Moreover, you'll be less likely to repeat the mistake in other parts of the model.

If you wait until you believe you have completed the model before testing it, it may contain several errors that interact in confusing ways. Then you'll have to search through much larger sections of the model to track them down. But if you have already tested the model components independently, you'll have already removed most of the errors, and it will usually be much easier to track down any that remain.

### **Test the model against reality**

The best way to check that your model is well-specified is to compare its predictions against past empirical observations. For example, if you're trying to predict future changes in the composition of acid rain, you should try to compare its "predictions" for past years for which you have empirical



observations. Or, if you're trying to forecast the future profitability of an existing enterprise, you should first calibrate your model for past years for which accounting data are available.

### **Test the model against other models**

Often you don't have the luxury of empirical measurements or data for the system of interest. In some cases, you're building a new model to replace an old model that is out-of-date, too limited, or not probabilistic. In these cases, it is usually wise to start by reimplementing a version of the old model, before updating and extending it. You can then compare the new model against the old one to check for discrepancies. Of course, differences may be due to errors in the new model or the old model. Once you have resolved any discrepancies, you can be confident that you are building on a foundation that you understand.

If the model is hard to test against reality in advance of using it, and if the consequences of mistakes could be catastrophic, you can borrow a technique that NASA uses widely for the space program. You can get two independent modelers (or two modelling teams) to each build their own model, then check the models against each other. It is important that the modelers be independent, and not discuss their work ahead of time, to reduce the chance that they will both make the same mistake. For a sponsor of models for critical applications in public or private policy, this multiple model approach can be very effective and insightful. The competition keeps the modelers on their toes. Comparing the models' structure and behavior often leads to valuable insights.

### **Test model behavior and sensitivities**

Many problems become immediately obvious when you look at a result, for example, if it has the wrong sign, the wrong order of magnitude, or the wrong dimensions, or if Analytica flags an evaluation error. Other problems, of course, are not immediately obvious, for example, if the value is wrong by only a few percentage points. For more thorough testing, it is often helpful to analyze the model behavior by specifying a list of alternative values for one or two key inputs (see Chapter 3, "Analyzing Model

---

Behavior”), and to perform sensitivity analysis (see Chapter 16, “Analyzing Uncertainty and Sensitivity”). If the model behaves in an unexpected way, this may be a sign of some mistake in the specification. For example, suppose that you are planning to borrow money to buy a new computer, and the net value increases with the interest rate on the loan; you might suspect a problem in the model.

### **Learn from unexpected behavior**

If analyzing the behavior or sensitivities of your model creates unexpected results, there are logically two possibilities:

- Your model contains an error, in that it does not correctly express what you intended.
- Your expectations about how the model should behave were wrong.

You should first check the model carefully to make sure it contains no errors, and does indeed express what you intended. Explore the model to try to figure out how it generates the unexpected results. If after thorough exploration, you can find no mistake, and the model persists in its unexpected behavior, do not despair! It may be that your intuitions were wrong in the first place. This discovery should be a cause for celebration rather than disappointment. If models always behaved exactly as expected there would be little reason to build them. The most valuable insights come from models that behave counterintuitively. When you understand how their behavior arises, you can deepen your understanding and improve your intuition, which is, after all, a fundamental goal of modeling.

### **Document the model as you build it**

Give your variables and modules meaningful titles, so that others—or you, when you revisit the model a year later—can more easily understand the model from looking at its influence diagrams. It’s better to call your variable *Net rental income* than *NR123*.

It’s also a good idea to document your model as you construct it by filling in the Description and Units attributes for each variable

and module. You may find that entering a line or three of description for each variable explaining clearly what the variable represents will help to keep you clear about the model. Entering units of measurement for each variable can help you avoid simple mistakes in model specification. Avoid the temptation to put documentation off until the end of the project, when you may run out of time, or may have forgotten key aspects.

Most models, once built, spend the majority of their lives being used and modified by people other than their original author. Clear and thorough documentation pays continuing dividends; a model is incomplete without it.

### **Have other people review your model**

It's often very helpful to have outside reviewers scrutinize your model. Experts with different views and experiences may have valuable comments and suggestions for improving it. One of the advantages of using Analytica over conventional modeling environments is that it's usually possible for an expert in the domain to review the model directly, without additional paper documentation. The reviewer can scrutinize the diagrams, the variables, their definitions, and the behavior of the model electronically. You can share models electronically on diskette, or over a network.

## **5.3 Expanding your model**

### **Extend the model by stages**

The best way to develop a model of appropriate size is to start with a very simple model, and then to extend it in stages in those ways that appear to be most important. With this approach, you'll have a usable model early on. Moreover, you can analyze the sensitivities of the simple model to find out where the key uncertainties and gaps are, and use this to set priorities for expanding the model. If, instead, you try to create a large model from the start, you run the risk of running out of time, or computer resources, before you have anything usable. And you

may end up putting much work into creating an elaborate module for an aspect of the problem that turns out to be of little importance.

### **Identify ways to improve the model**

There are many ways to expand a model:

- Add variables that you think will be important.
- Add objectives or criteria for evaluating outcomes.
- Expand the number of decision options specified for a decision variable, or the number of possible outcomes for a discrete chance variable.
- Expand a single decision into two or more sequential decisions, with the later decision being made after more information is revealed.
- For a dynamic model, expand the time horizon (say, from 10 years to 20 years) or reduce the time steps (say, from annual to quarterly time periods).
- Disaggregate a variable by adding a dimension (say, projecting sales and costs by each division of the company instead of only for the company as a whole).

Before plunging in to one of these approaches to expanding a model, it's best to list the alternatives explicitly and think carefully about which is most likely to improve the model the most for the least effort. Where possible, perform experiments or sensitivity analysis to figure out how much effect alternative kinds of expansion may have.

Changing the size or numbers of dimensions of tables is a difficult and time-consuming task in conventional modeling environments. Analytica makes it relatively easy, since you only need to change those definitions that directly depend on the dimension (for example, the input variables).

**Discover what parts are important to guide expansion**

A major advantage of starting with a simple model is that you use it to guide extensions in the ways that will be most valuable in improving the model's results. You can analyze the sensitivities of the simple model (for example, using importance analysis, as described in Section 16.5) to identify which sources of uncertainty contribute most to the uncertainty in the results. Typically, only a handful of variables contribute the lion's share of the overall uncertainty. You can then concentrate your future modeling efforts on those variables and avoid wasting your energy on variables whose influence is trivial.

Early intuitions about what aspects of a model are important are frequently wrong, and the results of the sensitivity analysis come as a surprise. Consequently, it's much safer to base model development on sensitivity analysis of simple models, than to rely on your intuitions about where to spend your efforts in model construction.

Once you have identified the most important variables in your simple model, there are several ways to reduce the uncertainty they contribute. You can refine the estimated probability distribution by consulting a better-informed expert, by analyzing more existing data, by collecting new data, or by developing a more elaborate model to calculate the variable based on other available information.

**Simplify where possible**

There's no reason that a model must grow successively more complex as you develop it. Sensitivity analysis may reveal that a variable or submodel is just not very important to the results. In this case, consider eliminating it. You may find that some dimensions of tables are unimportant—for example, that there's little difference in the performance of different divisions. If so, again, consider aggregating over the divisions and eliminating that dimension from your model.

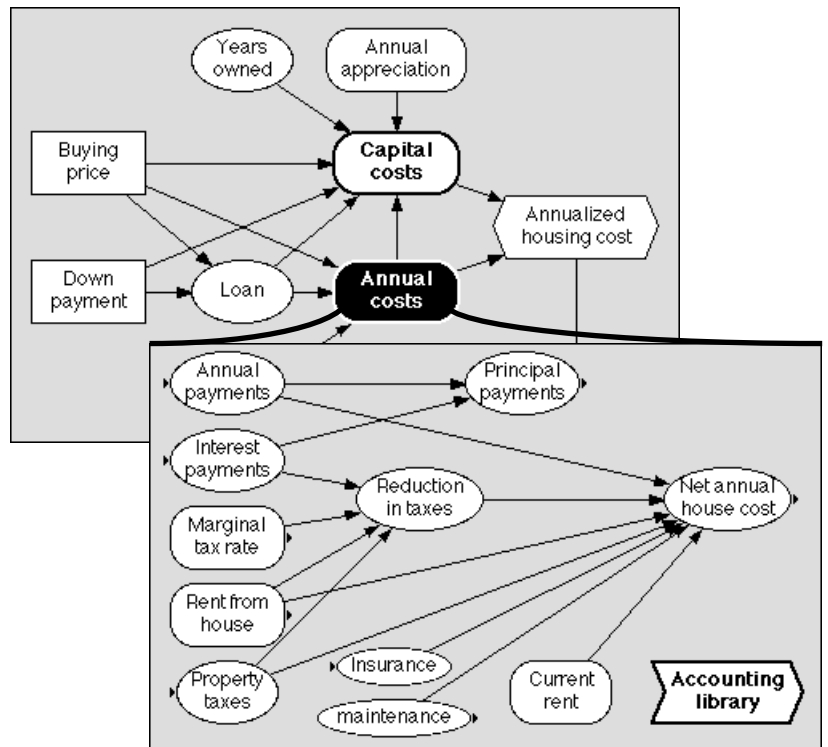
Simplifying a model has many benefits. It becomes easier to understand and explain, faster to run, and cheaper to maintain. These savings may afford you the opportunity to elaborate on some more significant aspect of the model.

# 6

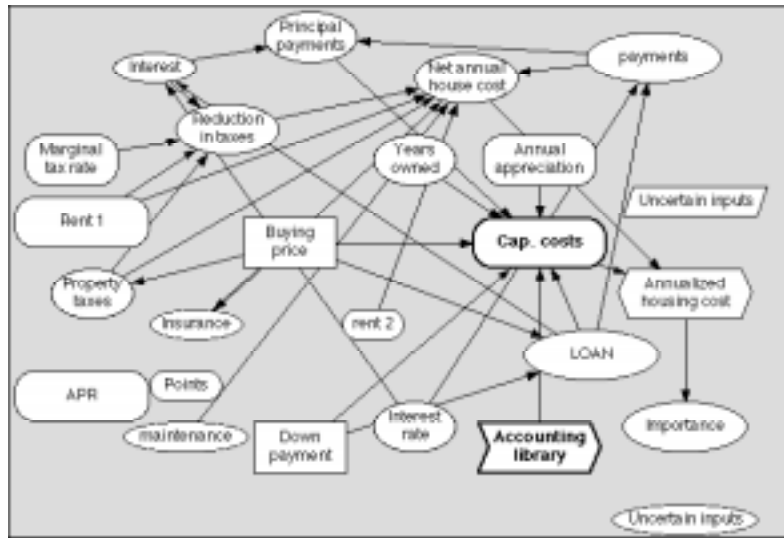
## Creating Lucid Influence Diagrams

This chapter presents guidelines for building influence diagrams in Analytica and explains how to customize your diagrams.

Hierarchical influence diagrams can provide an intuitive form to display the essential qualitative structure of a model with great clarity, uncluttered by the quantitative details.



It is also possible to create influence diagrams that are impenetrable spaghetti!



## 6.1 Guidelines for creating lucid and elegant diagrams

Where aesthetics are involved, rules cannot be hard and fast. You may want to adapt and modify these guidelines to suit your particular applications.

### Use clear, meaningful node titles

Aim to make each diagram stand by itself and be as comprehensible as possible. Each node title can contain up to 255 characters of any kind, including spaces. Use clear, concise language in titles, not private codes or names (as are often used for naming computer variables). Mixed case text (first letter uppercase

and remaining letters lowercase) is clearer than all letters uppercase.



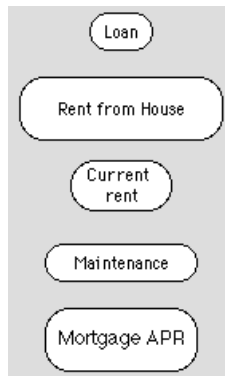
Poor object titles



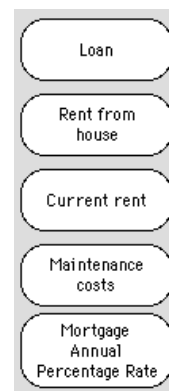
Good object titles

## Use consistent node sizes

Diagrams usually look best if most of the variable nodes are of the same size, rather than sized to fit their title text.



Inconsistent node sizes



Consistent node sizes

Node sizes will be uniform if you set the default minimum node size in the Diagram Style dialog box (see page 6-11) large enough so that it will fit the full title for almost all of the nodes. The



default minimum is used unless the text is too lengthy, in which case the node expands vertically to fit the text.

If you have nodes of several different sizes, you can make them more consistent by selecting **Adjust Size** (☞-T) from the **Diagram** menu. All of the selected nodes are resized to the default minimum node size, or the minimum size needed to enclose each node's title, whichever is larger.

You can also resize several nodes by the same amount simultaneously by following these steps:

1. Select the nodes to resize.
2. Resize one of the selected nodes by dragging one of its handles. All the other selected nodes are also resized.

### **Use small and large nodes sparingly**

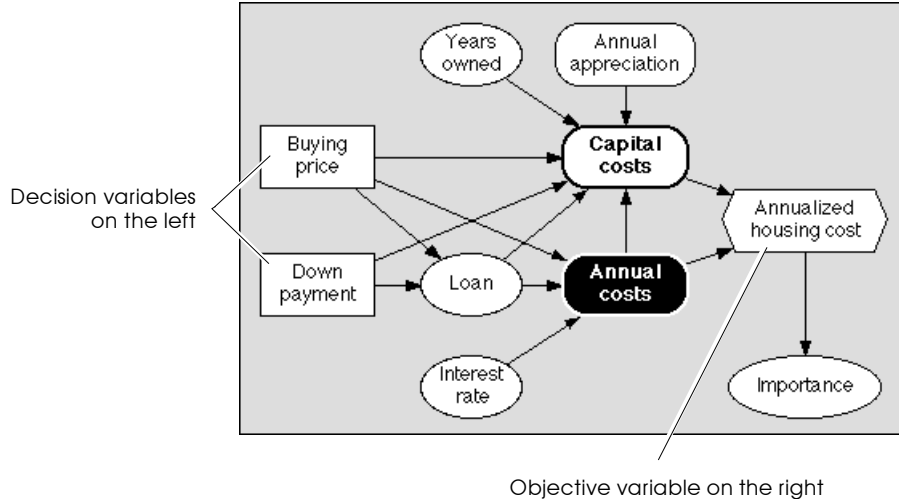
Sometimes it is more effective to make a few specialized nodes extra large or small. For example, start and end nodes, which may link to other models, often look best when they are very small. Conversely, you may want to make key input nodes containing large tables or model nodes containing the “guts” of a model unusually large to convey their importance.

### **Arrange nodes from left to right (or top to bottom)**

People like to read diagrams, like text, from left to right, or top to bottom.<sup>1</sup> Try to put the decision node(s) on the left or top and the objective node(s) on the right or bottom of the diagram, with all of the other variables or modules arranged between them.

You may want to allow a few arrows to go counter to the general flow in order to reduce crossing arrows, or overlaps. In dynamic models, there may be feedback loops (depicted with dashed arrows), which may appropriately go counter to the general flow.

1. For applications in Arabic, Hebrew, or other languages written from right to left, you may want to reverse this convention.



### Tolerate spaghetti at first...

It is often hard to figure out a clear diagram arrangement in advance. It is usually easiest to start a new model using the largest Diagram window you can get. Click the zoom box to have the diagram fill your screen. You may want to create key decisions and other input nodes near the left or top of the window, and objectives or output nodes near the right or bottom of the window. Aside from that, create nodes wherever you like, without worrying too much about clarity.

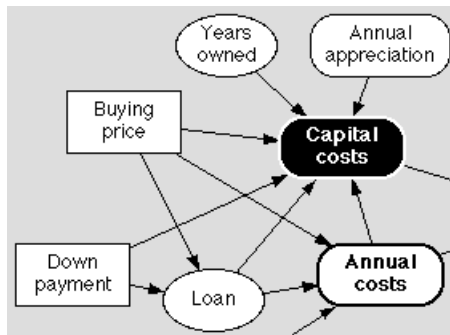
### ...reorganize later

When you start linking nodes, the diagram may start to look tangled. This is the time to start reorganizing the diagram to create some clarity. Try to move linked nodes together into a module. Develop vertical or horizontal lines of linked nodes. Accentuate symmetries, if you see them. Gradually, order will emerge.

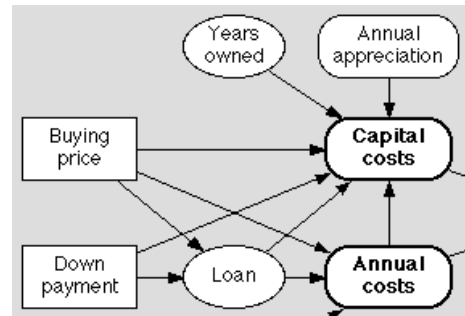
### Align nodes horizontally or vertically

It usually looks best to align nodes with their centers on the same horizontal or vertical lines, so that many arrows are exactly horizontal or vertical. The square grid of 9x9 points underlying each diagram makes this easy. When **Resize Centered** is selected in

the **Diagram** menu (the default), each node is centered on a grid point.



Poor alignment



Good alignment

If nodes are not centered on a grid point, recenter them by following these steps:

1. Select all nodes in the diagram with the **Select All** (⌘-A) command from the **Edit** menu.
2. Select **Align to Grid** from the **Diagram** menu.

## Hide less important arrows

Sometimes nodes are so interrelated that it is hard or impossible to arrange a diagram to avoid arrows crossing each other or crossing nodes. It may be helpful to hide some arrows that show less important linkages. For example, indexes are often connected to many other variables; therefore, hiding the arrows from indexes can greatly simplify a diagram.

You can hide all of the arrows linking indexes, functions, or modules, or the dashed feedback arrows in dynamic models, using the **Set Diagram Style** command from the **Diagram** menu (see page 6-11). You can also hide the input or output arrows from each node individually, using the **Set Node Style** command (see page 6-12).

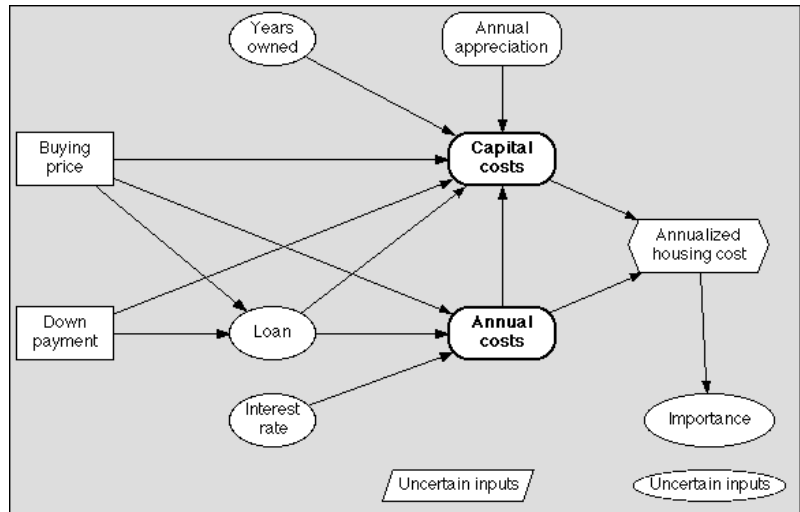
## Keep diagrams compact

Screen space is valuable. To save space, keep nodes close together, leaving enough space between them for the arrows to be visible.

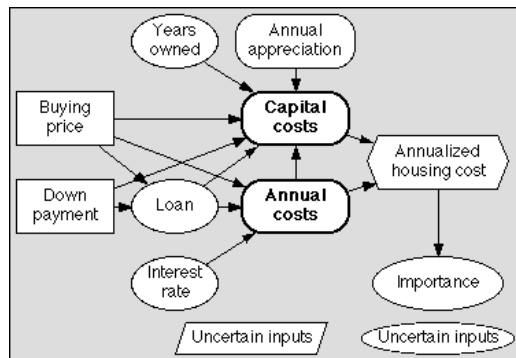
When first creating a diagram, use plenty of space. Your diagram window can be as large as your monitor screen. Using this space, find a clear arrangement, one that minimizes arrow crossing and avoids node overlaps.

After you have a clear arrangement, you can usually make the diagram more compact by moving the nodes closer together and moving the entire diagram closer to the upper left corner of the window. You can then reduce the window size to fit the diagram by dragging the resize box.

A spread-out diagram



A compact diagram

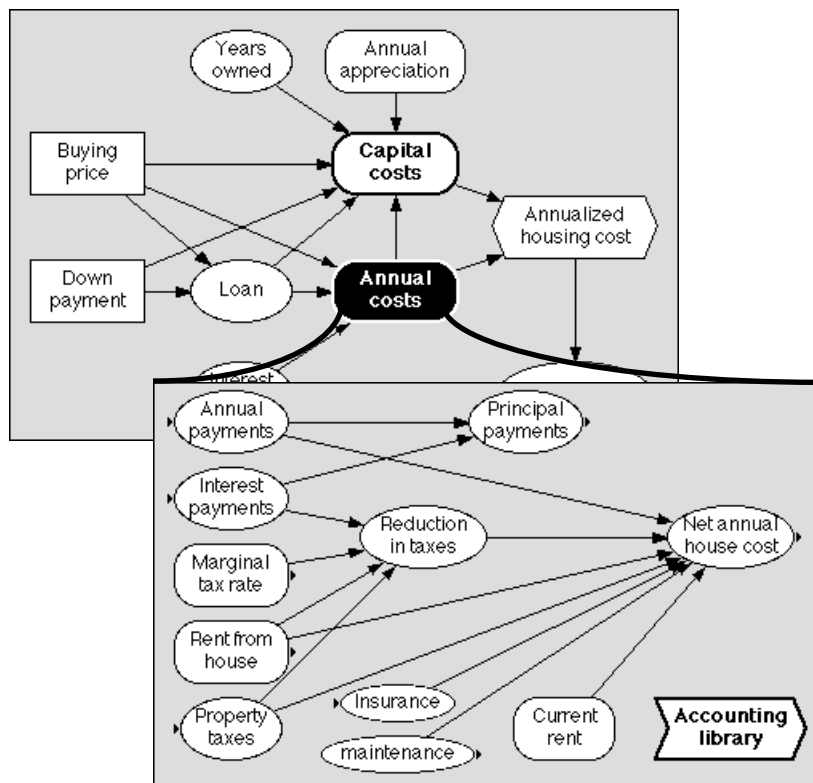


## 6.2 Organizing a module hierarchy

In addition to arranging the nodes in a single diagram well, you can also improve the clarity of your models by using module hierarchies effectively.

### Group related nodes in the same diagram

When assigning nodes to diagrams, the goal is to put groups of nodes with many links among them in the same diagram, and to separate them from other groups with which they have few or no links. For example, the diagram below shows that a group of nodes related to annual housing costs have been organized into the *Annual costs* module within the larger model.



Sometimes you have a good idea of how to group nodes before you create them. In such cases, it is easy to create the modules first, and then create and link the nodes in groups in each module.

In other cases, it may not be obvious what groupings will work best. It is then often best to create all the nodes in a single large diagram. After drawing all the arrows, you may have a confusing spaghetti diagram. At this point, try to move the nodes around to identify groups containing 5 to 15 nodes, with many links within each group and fewer links between groups. When you arrive at a satisfactory grouping, create a module node for each group, and move the group of variables into its own module.

### Use 5 to 15 nodes per diagram

In creating a hierarchy of diagrams of a model that contains 100 variables, you could create a single module with 100 nodes, 10 modules with an average of 11 nodes each, 20 modules with 6 nodes each, or 50 modules with 3 nodes each.<sup>2</sup>

A module containing more than 15 nodes is often hard to decipher, unless there are very strong regularities in the structure. On the other hand, if the modules are small, averaging fewer than 5 nodes, you need so many modules that it is easy for users to get lost.

The range of 5 to 15 nodes per diagram is a good general goal. But don't feel too constrained by it if a few diagrams must be much smaller or larger than this range.

Contrast the module hierarchy in the illustration on page 6-8 with the model on page 6-2. The relationships among objects are much easier to see and understand in the model with 10 nodes in the top-level module and 12 nodes in the embedded module (page 6-8) than in the model with 25 top-level nodes (page 6-2).

## 6.3 Color in influence diagrams

Color can greatly improve the clarity and appeal of diagrams. The diagram's background and its nodes are all lightly colored by default. You can change the colors to meet your special needs.

2. Note that each module also creates a new node, so the total number of nodes is the number of variables plus the number of modules.

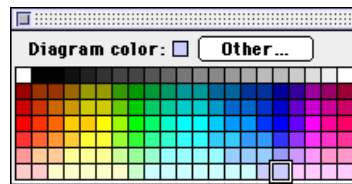
**Use colors judiciously** Selecting garish, uncoordinated colors can take attention away from the diagram. Light colors work best because the black arrows and text are easier to read over them. Analytica's default colors provide a light neutral color for the background and a slightly stronger color for the nodes.

**Background color** Light background colors work best so that the black arrows show up clearly.

**Node colors** If you wish to change the color of nodes, it is best to have all similar nodes be the same color. It generally looks messy to have nodes in many different colors.

## 6.4 Changing background or node colors

To change the color of the diagram background, or one or more nodes, select the Edit Tool and bring the diagram window to the front. Select **Show Color Palette** from the **Diagram** menu.



Select the node or nodes, or click in the diagram background to select the background for changing color. The current color displays in the single square at the top of the color palette. Click on a color square to select the new color.

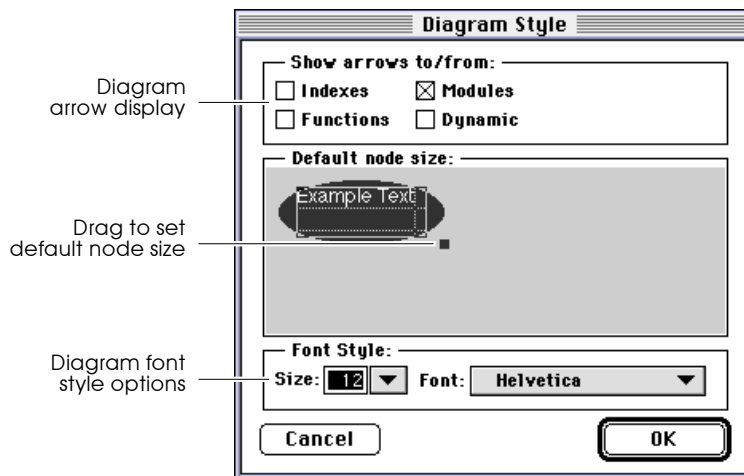
For more color selections, click on the **Other** button to display a color wheel with the colors available on your monitor.

In the color wheel, select a color by clicking with the mouse pointer at the desired color.

## 6.5 Diagram Style dialog box

Use the Diagram Style dialog box to control various aspects of the diagram display: the default font size and typeface for the node labels, whether arrows are displayed for specified node classes, and the default node size.

To display the Diagram Style dialog box, select **Set Diagram Style...** from the **Diagram** menu.



**Show arrows to/from** Use the options in this box to control various arrow displays.

### Indexes

Turns on or off the display of arrows into and out of index variables.

### Functions

Turns on or off the display of arrows into and out of functions.

### Modules

Turns on or off the display of arrows into and out of modules.



### Dynamic

Shows and hides dynamic arrows (for variables defined using the `Dynamic()` function; see page 17-2).

### Default node size

Drag the handle in this box to set the default node size. When you create a new variable or select the **Adjust Size** command from the **Diagram** menu, the node is made this size. When you change the title of a node, its size is adjusted to this size if the new title fits within it.

### Font Style

Use the options in this box to set a default typeface and font (size) for all the nodes in the model.

## 6.6 Node Style dialog box

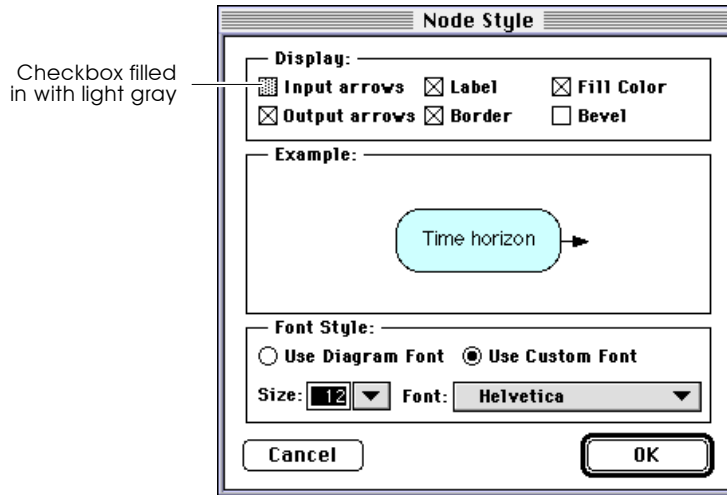
Use the Node Style dialog box to control the display of one or more nodes in a diagram.

You can specify the typeface and font (size), and whether to display the incoming arrows, outgoing arrows, the node outline, or the node label. The options for each node override the defaults specified for the entire diagram in the Diagram Style dialog box.

### Changing the node style


To change the node style:

1. Select one or more nodes.
2. Choose **Set Node Style...** from the **Diagram** menu.



<b>Display</b>	Use the options in this box to control various display options:
<b>Input Arrows</b>	Display arrows coming into a node.
<b>Output Arrows</b>	Display arrows going out of a node.
<b>Label</b>	Display the node label (title or identifier).
<b>Border</b>	Display the node border.
<b>Fill Color</b>	Display the node color. If unchecked, the node will appear transparent.
<b>Bevel</b>	Display the border beveling (3D button effect).

---

**Note:**  A checkbox filled in with light gray indicates that this option is not the same for all selected nodes. If you leave it unchanged (gray), each node keeps its current setting for this option. If you change this option (on or off), all nodes are changed to the new setting.

---

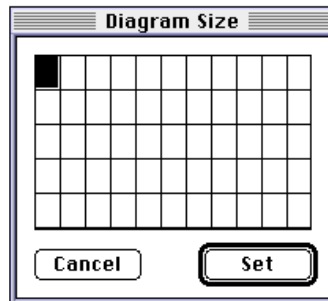
**Font Style** Use the options in this box to change the typeface and font (size) from the defaults (see page 6-12) to a custom style for the selected node(s).

## 6.7 Changing the size of the diagram

The diagram is preset to display and print in the orientation determined by the setting in the Page Setup dialog box. On your monitor, the size is shown by the extent of the colored background. You can change the size of the diagram in whole-page increments.

To change the size of the diagram:


1. Bring the diagram window to the front.
2. Select **Set Diagram Size...** from the **Diagram** menu.



3. Specify the size diagram you want.

Each rectangle in the grid represents a page. The size of the page is determined by the paper size and the Reduce or Enlarge percentage setting in Page Setup. To increase or decrease the size by whole-page increments, click the rectangle that you want to have as the bottom-right boundary of the diagram.

---

**Note:**  If you are decreasing the diagram size, you cannot remove pages that contain nodes.


---

#### 4. Click on **Set**.

When you save a model and later open it, the diagram size is reset to the number of pages holding the nodes.

## 6.8 Taking screenshots of diagrams

This section contains some tips for taking good screenshots of influence diagrams and other Analytica windows for use in hardcopy documents.

**Use Browse mode** When making screen captures of a Diagram window, be sure that the Browse tool () is selected, rather than the Edit or Arrow tool. The diagram is clearer in Browse mode, without the background grid visible.

**Switch off cross-hatching** By default, the nodes of undefined variables show a cross-hatched pattern around the title. To get rid of this pattern, deselect the **Show undefined** option in the Preferences dialog box (see “Preferences dialog box” on page 4-19).

**Diagram colors** Use white for the background if you plan to print the diagram on a black and white printer at less than 600 dpi (dots per inch). A light gray works well on a printed version if you have a 600 dpi or better printer.

**Use a common level of reduction** When scaling down screenshots of windows, use a consistent reduction value. If your page setup precision bitmap alignment option is on, use a multiple of 25%; if the precision bitmap alignment option is off, use a multiple of 24%. Other reductions

can create interference in printing and result in distorted screenshots.

**Fit windows to tables** When showing a table or Edit Table, click on the window's zoom box so that the window is an exact fit to the table.

---

# 7

# Formatting Graphs and Tables

---

This chapter describes how to control the display of results in graphs and tables.

## 7.1 Graph Setup dialog box

Use the Graph Setup dialog box to select the graphing tool and control graphing options.

Display the Graph Setup dialog box in one of two ways:

- Select **Graph Setup** from the **Result** menu.
- Double-click on a graph in the Result window.

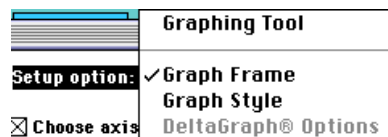
To set defaults for all new graphs, open the Graph Setup dialog box when no graph is the active window.

To establish settings for the graph of results of a specific variable, open the Graph Setup dialog box when that graph is the active window.

The settings are saved when you save the model.

### Setup option popup menu

Several options for viewing and changing settings in the Graph Setup dialog box are accessible using its **Setup option** popup menu.



**Buttons** Set Default

Accepts all Graph Setup settings for the current and all future graphs, and closes the dialog box.

**Apply**

Applies all Graph Setup settings to the current graph, and closes the dialog box.

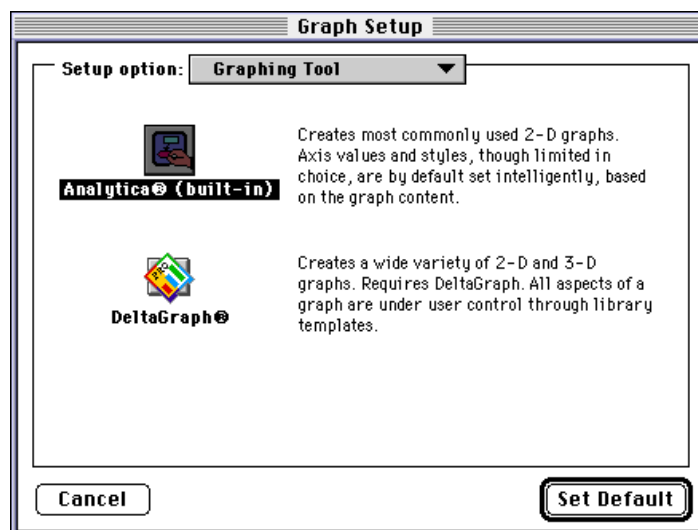
**Cancel**

Leaves the Graph Setup settings unchanged, and closes the dialog box.

When you first open the Graph Setup dialog box for a model, the Graph Frame setup option displays.

## 7.2 Graphing Tool setup option

Use the Graphing Tool option to switch between Analytica and DeltaGraph (if it is installed on your computer) for graphing results.

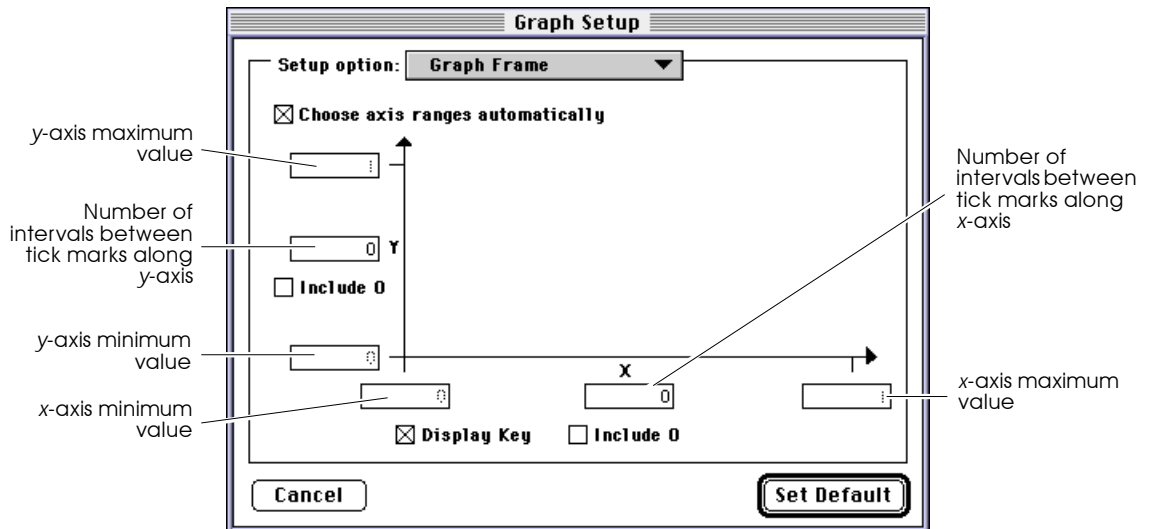


**Analytica® (built-in)** By default, Analytica's graphing tool is selected. If you previously had selected DeltaGraph, select this option to choose Analytica's graphing tool.

**DeltaGraph®** Select this option to use DeltaGraph. DeltaGraph will launch when you select this icon and will remain open until you quit the Analytica session. In the next Analytica session, DeltaGraph will launch when you first evaluate a variable to which the selection applies and choose the graph view.

## 7.3 Graph Frame setup option

To change the graph frame in an Analytica graph, select the **Graph Frame** setup option from the popup menu.



### Value entry boxes    Number of intervals between tick marks

If 0, Analytica chooses the number of intervals. If you enter a number,  $n$ , Analytica uses either  $n$  or  $n+1$ , depending on the minimum and maximum values.



### **Minimum/maximum value**

After un-checking the Choose axis ranges automatically checkbox, you can enter the desired value.

### **Checkboxes** Choose axis ranges automatically

Controls whether the ranges on the axes are set automatically. You must un-check this box before you can edit the minimum and maximum fields for each axis. For bar graphs, you can change only the *y*-axis values.

You cannot uncheck this box to set defaults for all new graphs; you must uncheck it for *each* graph.

### **Display Key**

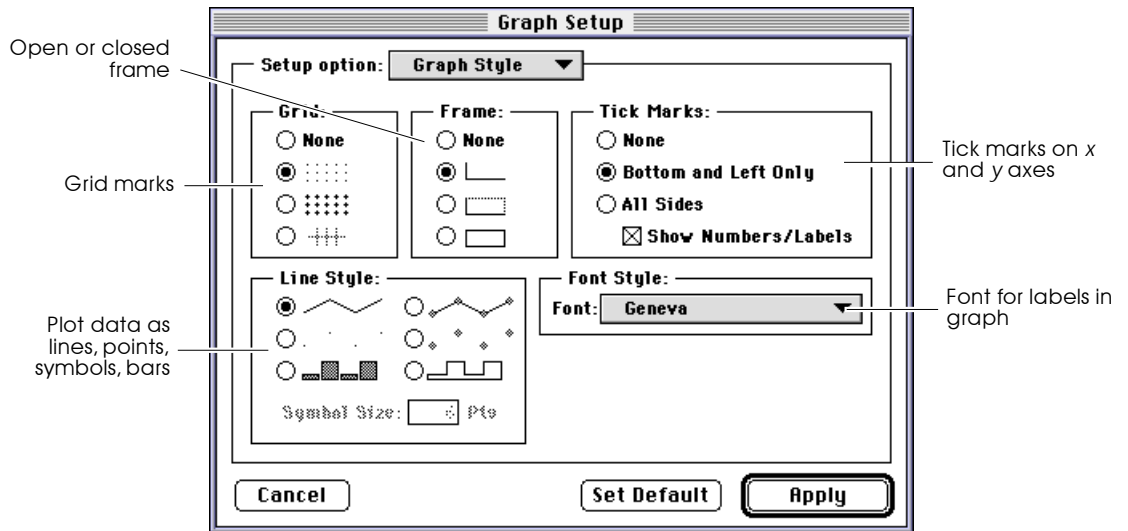
Shows the key (for a result of two or more dimensions).

### **Include 0**

Includes 0 (the origin) on the given axis.

## 7.4 Graph Style setup option

To change the graph style in an Analytica graph, select the **Graph Style** setup option from the popup menu.



**Grid** Controls whether a background grid displays, and if it is comprised of dots or lines.

**Frame** Controls whether the graph displays the axes alone, or with a frame around the graph.

**Tick Marks** Controls how the tick marks appear along the axes.

**None** Display no tick marks.

**Bottom and Left Only** Display tick marks along the bottom and left hand axes.

**All Sides** Display tick marks all around the frame.

**Show Numbers/Labels**

Display numbers or labels along the axes.

**Line Style**

Controls the style of the graph.

**Line graph**

Different line styles and colors are used for each key value.

**Line and data markers**

Different line styles, colors, and symbols are used for each key value. You can size the symbols.

**Data markers (dots)**

Useful with a large number of data points.

**Data markers only**

Different symbols are used for each key value; you can size the symbols.

**Bar chart**

Bars are of equal width and are center labelled on the horizontal axis. (Default for Probability Mass Function of a Protable.)

**Histogram**

A rectangle between two points  $x_1, y_1$  and  $x_2, y_2$  is created as  $x_1, y_1$  to  $x_1, y_2$  to  $x_2, y_2$ . The first rectangle is created as  $0, y_1$  to  $x_1, y_1$ .

**Symbol Size**

Enter the desired symbol size in points.

Minimum size: 4

Default size: 6

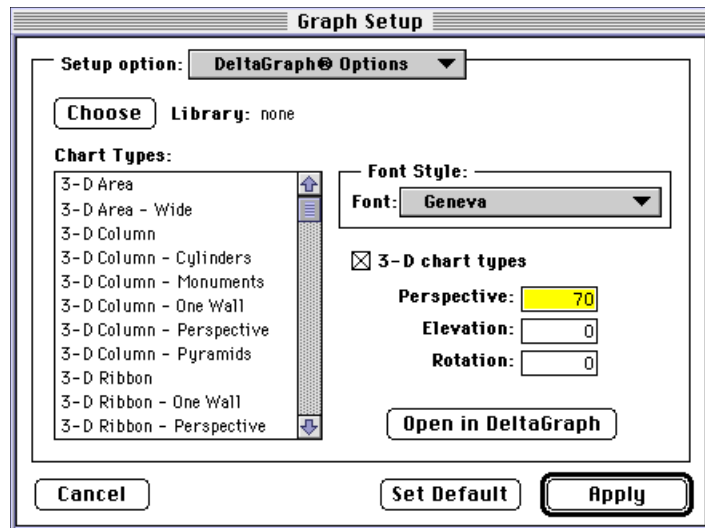
Maximum size: 36

**Font Style**

Sets the typeface for the graph. (Font size is determined by the window size and is adjusted when the window is resized.)

## 7.5 DeltaGraph options setup option

This option is available only if the DeltaGraph option was selected as the graphing tool (see Section 7.2). Use it for selecting a library, chart type, font style, and 3-D options.



### Choose

Click to choose a DeltaGraph library. The name of the currently selected library is shown next to the button. Only one library may be open at a time.

### Chart Types

Select a chart type template from the library.

### Font Style

Enter the font name.

### 3-D Chart types

Check this box to list 3-dimensional chart types. If not checked, other chart types are listed.

#### Perspective

Enter a number (0% to 100%) for the apparent distance between the front and back planes of the 3-D graph. A value of 40% or less will create a very deep graph and a value of 100% will create a very shallow graph.

#### Elevation

Enter the view angle ( $-90^\circ$  to  $+90^\circ$ ) of the 3-D graph from above the horizontal plane. A positive angle will be a view of the graph from above. An angle of  $0^\circ$  will be a view level with the horizontal plane. A negative angle will be a view of the graph from below.

#### Rotation

Enter the rotation ( $0^\circ$  to  $360^\circ$ ) of the 3-D graph around the z-axis. An angle of  $0^\circ$  will be a view from the front (looks directly into the  $xz$  plane). An angle of  $90^\circ$  will be a view from the right (looks directly into the  $yz$  plane).

### Open in DeltaGraph

This button is enabled when the Graph Setup dialog box was opened with a DeltaGraph-generated graph as the active window.

Click to open the graph within DeltaGraph, to create or modify the custom library chart template for the graph. Keyboard shortcut: hold down the option key while double-clicking on a DeltaGraph-generated graph.

After making changes in DeltaGraph, switch back to Analytica in one of the following ways:

- Click in an Analytica window
- Select “Analytica” from the application menu (rightmost icon in the menu bar).


To use a newly created custom library chart template, open the Graph Setup dialog box and select the new chart name from the chart list.

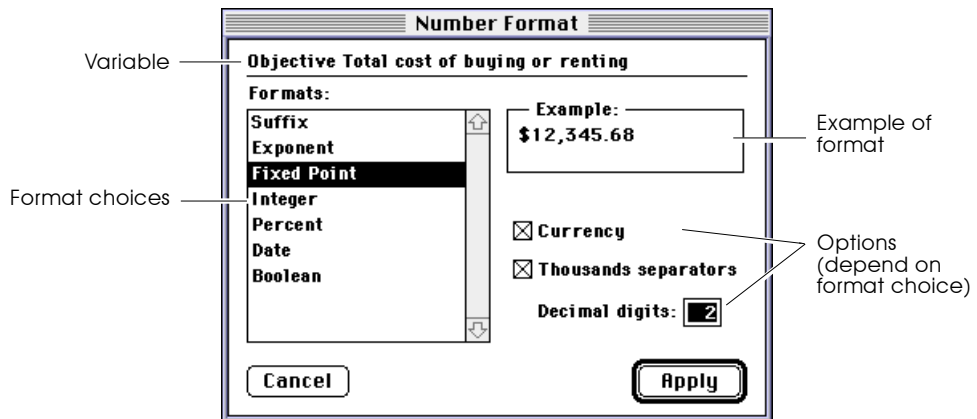
## 7.6 Number Format dialog box

Number formats can be specified for a table's contents, its row and column indexes, and for the y axis on a graph.

The number format for a variable affects the display of all of its values everywhere they appear. For example, if you set the number format for an Index variable in one Result window, the same number format is used if the Index variable appears in another Result window.

To set the number format:

1. Open a Result window.
2. If the Result window is a table, select a row, column, or cell.
3. Choose **Number Format** from the **Result** menu or -B to display the Number Format dialog box.



The top line shows the variable to which the number format will be applied.


## Formats

Choose from the following number formats:

Format	Description	Example
Suffix	the default (see the following table)	12.35K
Exponent	scientific exponential	1.235e04
Fixed Point	fixed decimal point	12345.68
Integer	fixed point with no decimals	12346
Percent	percentage	1234568%
Date	text date	12 Jan 93
Boolean	true or false	True

The suffix characters are:

Power of 10	Suffix	Prefix	Power of 10	Suffix	Prefix
3	K	Kilo	-2	%	percent
6	M	Mega or Million	-3	m	milli
9	G	Giga	-6	u	micro (mu)
12	T	Tera or Trillion	-9	n	nano
15	Q	Quad	-12	p	pico
			-15	f	femto

**Note:**  If fixed point is selected, a number larger than  $10^9$  displays in exponent format.

For fixed point, integer, and percent, the currency symbol, thousands separators, and decimal point are determined by the local geographic region that your Macintosh system software supports. You can change these formats by installing different localized versions of Macintosh system software on your computer, or by using the Numbers control panel.

**Options** The options in the Number Format dialog box depend on the format selected.

The maximum number of digits or decimal digits is six; the maximum number precision is six digits. The Suffix format shows a minimum of four significant digits.

### Currency

The currency symbol, as shown in your Macintosh Numbers control panel.

### Decimal digits

If set to 1 or more, the decimal symbol is as shown in your Macintosh Numbers control panel. Numbers are padded with zeros to fill out the specified number of digits.

### Number of digits

For Exponent format, numbers are padded with zeros. For Suffix format, fewer digits can be displayed.

### Date formats

These formats show a number as a date, computed as the number of days since January 1, 1904 (33,969 is January 1, 1997). Set the long and short formats in your Macintosh Date & Time control panel.

### Thousands separators

The symbol as shown in your Macintosh Numbers control panel.

---

**Note:**  Suffix and exponent formats always use period (.) for the decimal point.

---





---


# 8



## Creating and Editing Definitions

---

This chapter introduces the tools for creating and editing powerful mathematical models by giving each variable a formula that defines how to compute its value in its *definition*. The definition of a variable can be a simple number, text, a probability distribution, or a more complicated expression. It can also be a list or table of numbers or other expressions. Subsequent chapters present more details about using mathematical expressions, arrays, and probability distributions.

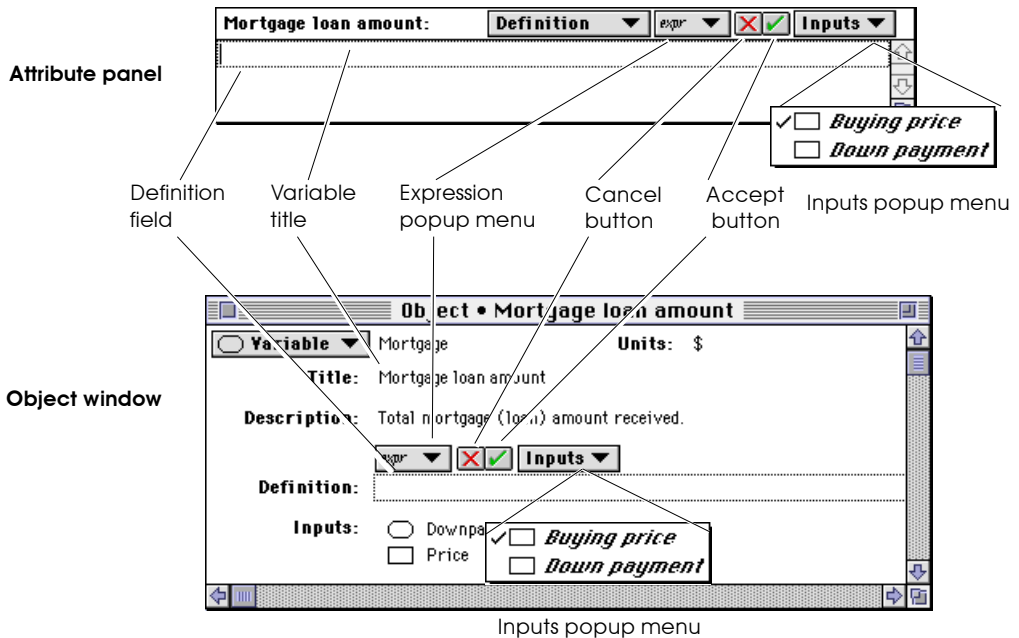
### 8.1 Creating or editing a definition

To create or edit the definition of a variable, first be sure that the Edit tool () is selected. Select the variable and do any of the following:

- Enter ⌘-E.
- Click on () in the tool palette.
- Select **Edit Definition** from the **Definition** menu.
- Double-click on the variable to open its Object window; then click in the definition field.
- Click on the Key icon () to open the Attribute panel of the diagram; select **Definition** from the Attribute popup menu; then click in the definition field.

If the inputs to the variable were specified by drawing arrows in the diagram, the definition initially looks like the illustration below. The definition field is blank and a popup menu listing the

inputs to the variable appears above the definition field. If the variable has no inputs, the **Inputs** popup menu does not appear.



To edit a definition that is a simple number, text, or other expression:

1. Select the definition.
2. Edit it by typing, by deleting, or by using the standard Macintosh text editing operators, that is, Copy ( $\text{⌘-C}$ ), Cut ( $\text{⌘-X}$ ), and Paste ( $\text{⌘-V}$ ).

See Chapter 10, “Using Expressions,” for the syntax of numbers, operators, simple expressions, and mathematical functions.

You can change the definition to one of several commonly used expressions with the Expression popup menu (see Section 8.3).

**Identifiers** To refer to the value of another variable, use its identifier. To place a variable's identifier at the insertion point in the definition, do any of the following:

- If the variable is an input, select it from the Inputs popup menu.
- Type in the variable's identifier. To see all nodes in the active diagram labelled with their identifiers, select **Show By Identifier** from the **Object** menu (⌘-Y).
- Select **Paste Identifier** from the **Definition** menu and use the Find button or identifier menu items (see Section 8.4).

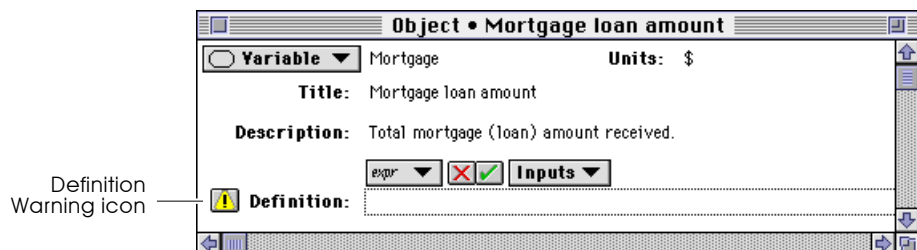
**Functions** You can paste functions at the insertion point by doing either of the following:

- Select **Paste Identifier** from the **Definition** menu to open the Object Finder (see Section 8.4).
- Select the function from its library in the **Definition** menu (see Section 8.5).

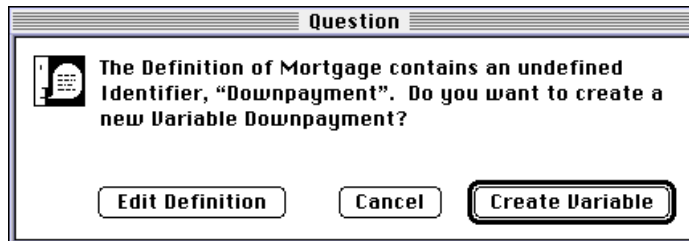
**Syntax check** After entering or editing a definition, press *enter* or click on the accept button (☑) to perform a syntax check of the revised definition and accept the changes.

Click on the cancel button (☒) to cancel your changes.

The definition Warning icon (⚠) appears next to the definition if it is not syntactically correct. Click on the icon to see a message about what may be wrong.



A definition's syntax check may reveal syntax errors. (See “Syntax error” on page E-2.) For example, if a definition contains text that is not an identifier, the following dialog box appears:



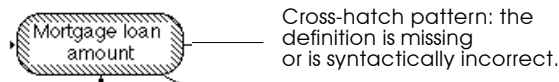
## 8.2 How a valid definition may change the diagram

After you give a variable a valid definition, the influence diagram containing that variable might change.

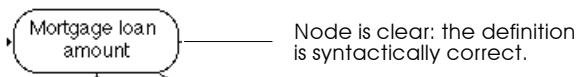
### Cross-hatching disappears

If the “Show Undefined” preference is selected (see Section 4.10), a node whose definition is missing or syntactically incorrect displays with a cross-hatch pattern.

For example:



After the definition is checked to be syntactically correct, the variable's node in the influence diagram is clear.



**Arrow updating** As part of the syntax check, the influence arrows going into the active variable (its inputs) are reconciled with the definition.

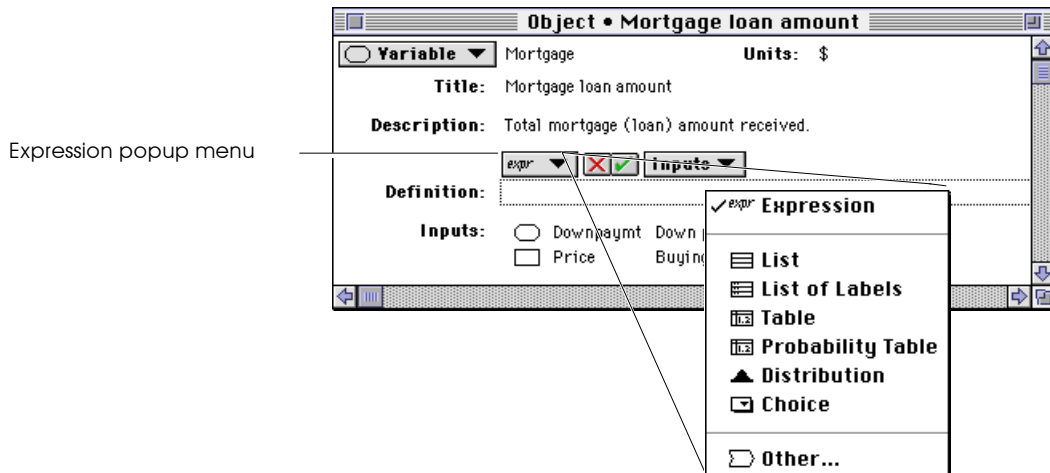
- An arrow is drawn if the identifier of another variable is included in the definition.
- An arrow is removed if the identifier of an input is omitted from the definition.

To avoid removing influence arrows while editing a definition, do not click on the check mark or press *enter* to leave the definition. Instead:

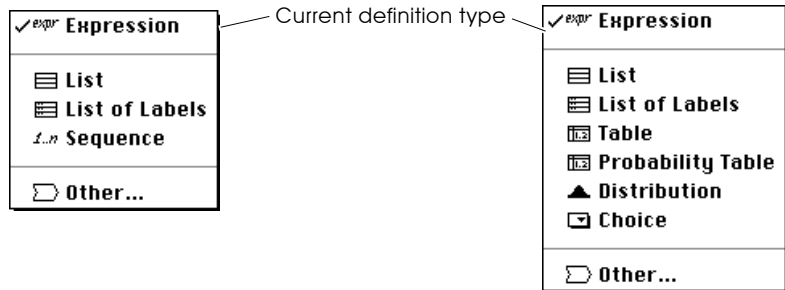
- In the Object window, click on the close box.
- In the attribute panel, select a different attribute or select a different variable in the diagram.

## 8.3 The Expression popup menu

Press on *expr* to see the Expression popup menu. The Expression popup menu shows the type of the definition, which is an empty expression in the following figure.



Use this popup menu to change the definition to one of several common kinds of expressions. The entries in this menu depend on the class of the node being defined.



**Expression** Shows the definition as a mathematical expression, even if it was defined using the other expression types in this popup menu. See Chapter 10, “Using Expressions.”

**List** Creates an ordered set of expressions or numbers. See Section 11.3, “Creating an index”.

**List of Labels** Creates an ordered set of text labels. See Section 11.3, “Creating an index”.

**Sequence** Creates a list of numerical values. See “Sequence(Start, End, Stepsize)” on page 12-5.

**Table** Creates an array of numbers or expressions. See Chapter 11, “Modeling with Arrays and Tables.”

**Probability table** Creates an array defining probabilities (numbers or expressions) across the domain of a discrete (chance) variable. See Section 15.1, “Using a probability table.”

- Distribution** Creates an uncertain definition by selecting a function from the Distribution system library. See Section 13.2, “Defining a variable as a distribution.”
- Choice** Creates a popup menu for choosing one or all elements from a list. See Section 9.2, “Creating a popup menu.”
- Other** Opens the Object Finder dialog box, which is described in the next section. Changes the definition to the function or variable that you select from the Object Finder.

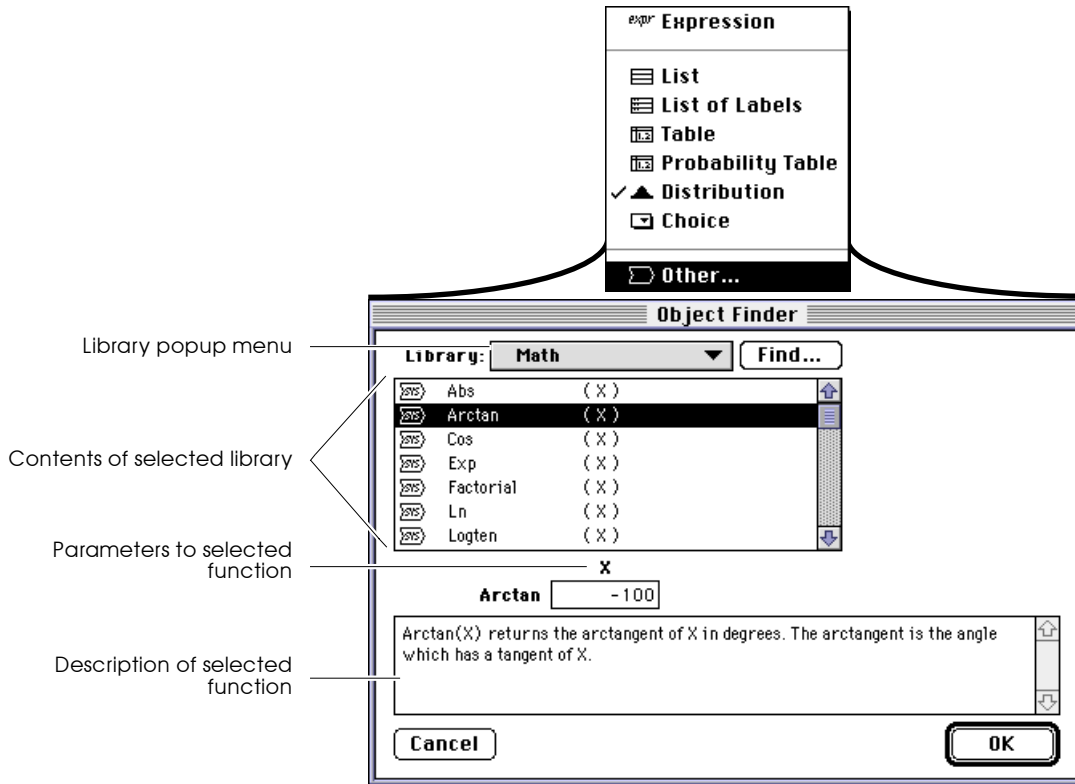
## 8.4 Object Finder dialog box

Use the Object Finder dialog box to browse system functions, your own library functions, and all of a model’s identifiers and place any of these objects into a definition.

Open the Object Finder in either of the following ways:

- To insert the desired function or identifier at the insertion point in the definition, select **Paste Identifier** from the **Definition** menu.
- To replace the entire definition with the desired function or identifier, select **Other** from the Expression popup menu.



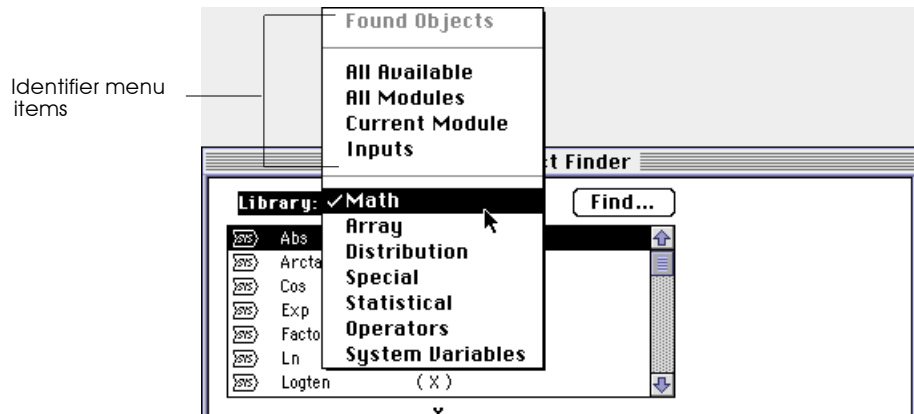


Use the **Library** popup menu to select a group of identifiers or a library.

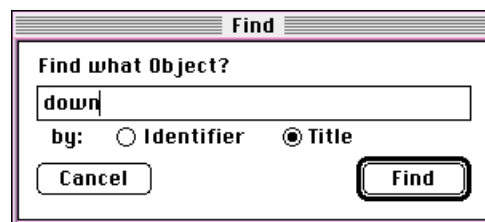
- For identifiers, scroll to select:

<b>Found Objects</b>	Displays identifiers of objects found with the Find dialog. (see below)
<b>All Available</b>	Displays all library functions and identifiers.
<b>All Modules</b>	Displays identifiers in all modules.
<b>Current Module</b>	Displays identifiers in the current module.
<b>Inputs</b>	Displays identifiers of the inputs to the selected node.

- For a library, the contents of the selected library are listed below the popup menu, showing the parameters if they are functions.

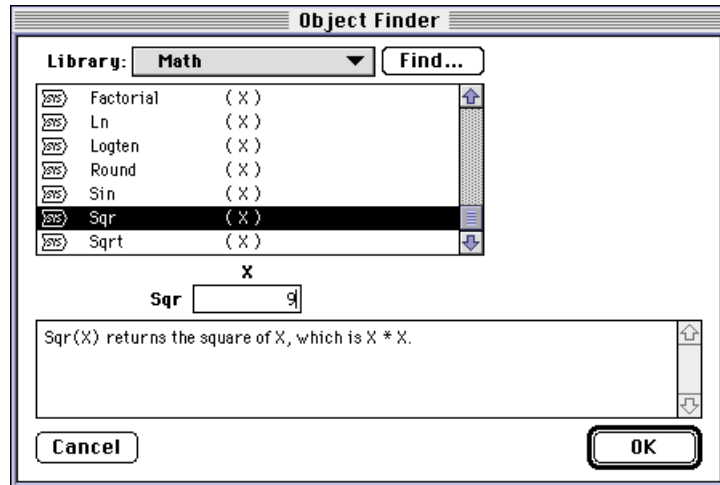


Use the **Find** button to search on the identifiers or titles of all variables, modules, and functions.



Matching objects are listed in the Found Objects library.

To use a function, identifier, or system expression in a definition, select it. For a function, enter the required parameters in the parameter fields.



Click on **OK** to place the function, identifier, or expression in the definition.

 **Definition:** Sqr( 9 )

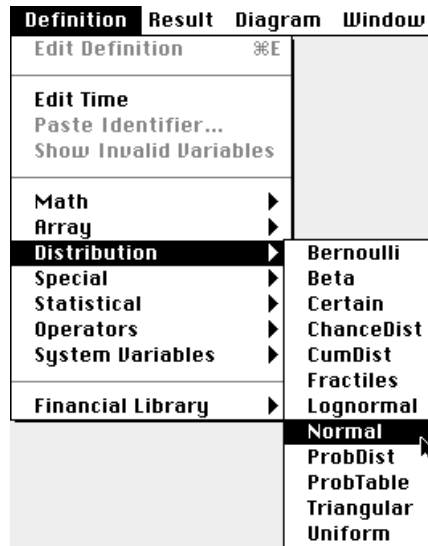
## 8.5 Pasting from a library in the Definition menu

Use the **Definition** menu to quickly paste a function or system expression into a definition, when you do not need the description of the function or expression.

While editing a definition:

1. Have the cursor at the point you want to insert a function or expression.

- From the **Definition** menu, select the library and then the function or expression.



- Release the mouse and the function or expression is pasted into the definition.



- Replace all parameters with input identifiers or expressions. Each parameter is enclosed in << >>. To replace it, select both << >> and its contents, then type or use the Inputs button or the **Paste Identifier** command to open the Object Finder dialog box.

## 8.6 Checking the validity of a variable's values

You can create an automatic check on the validity of the value of a variable using its Check attribute. For example, to check that the value of Percent\_damage is between 0 and 100, you give it a check of:

```
Percent_damage >= 0 AND Percent_damage <= 100
```

When the variable is evaluated, and if the Check attribute fails (evaluates to *False*), Analytica will give a warning and the opportunity to edit the definition.

There are two steps to using value checking:

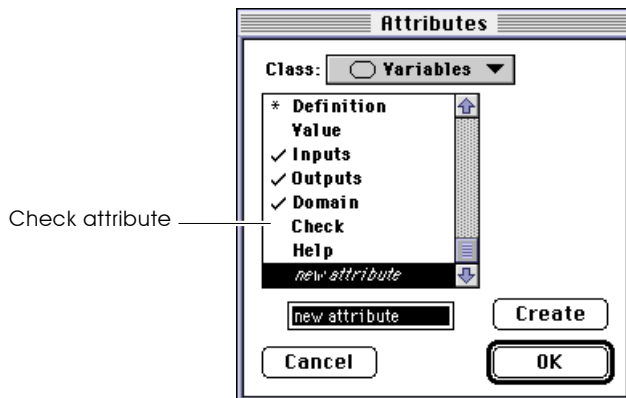
1. Display the Check attribute.
2. Define checks for variables.

### Displaying the Check attribute

If you want to use checking, first set the Check attribute to be displayed in the Object window and Attribute view, since it is hidden by default.

To show the Check attribute:

1. Select **Attributes** from the **Object** menu to open the Attributes dialog box. See “Managing attributes”, Section 19.4.



2. Scroll down the Attribute list and find Check.
3. Click on Check once to select it, and a second time to add a check mark next to it. The check mark indicates that the attribute is displayed in the Object window and in the Attribute popup menu.
4. Click on the **OK** button.

Now the Check attribute appears in Object windows and in the Attribute popup menu in the Attribute panel below the diagram.

## Defining the check

You can set the check attribute for any variable. First open the Object window for the variable, or show its Check attribute in the Attribute view. Enter an expression in the Check attribute field to constrain the value of a variable. The expression should refer to this variable by identifier or *Self*, and must be a Boolean (that is, evaluate to *True* or *False*). For example, to constrain the value for the lifetime of a car (*Lifetime*) to be greater than 0 and less than 12, define the check as:

**Check:** (Lifetime > 0) And (Lifetime < 12)

or

**Check:** (Self > 0) And (Self < 12)

If the check expression refers to another variable, a dependency is created between the variable being checked and the variable included in the check expression. If the definition does not already refer to the other variable, an arrow will be drawn between the two variables.

## Triggering a check

Analytica performs the check the first time it evaluates the checked variable. Analytica evaluates a variable the first time you ask to see its result, or the result of another variable that depends on it. Analytica also performs the check on an input node immediately after you edit the input value (see Section 9.1, “Using input nodes”).

**If a check fails** If a check fails (the check evaluates to *False*), Analytica gives you the option of editing the variable's definition, cancelling, or continuing. If you continue, the check will not be performed again unless you change the definition of the variable or a variable it depends on.

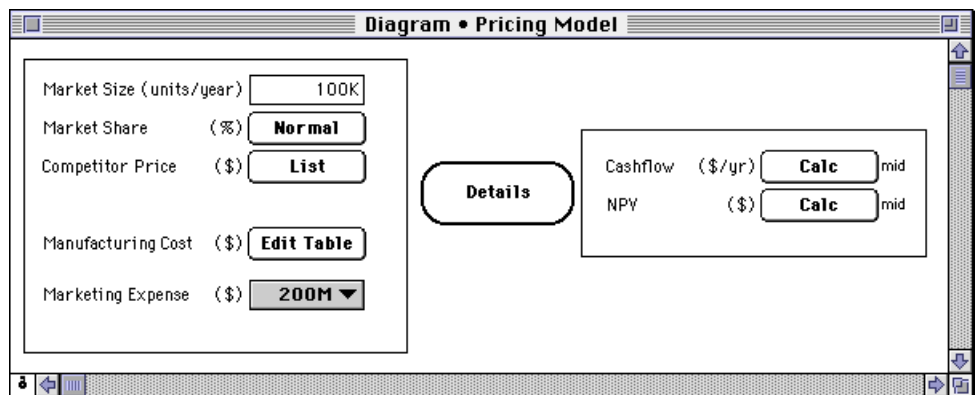
**Disabling value checking** You can disable all value checking by unchecking the **Check value bounds** checkbox in the Preferences dialog box (see page 4-19). This checkbox is checked by default.



# Creating Models to be Used by Others

You can use input and output nodes to create a simple user interface for other people who will use your Analytica model. Input nodes allow the user to see and change the values of variables directly from Diagram windows. Similarly, with output nodes you can display selected output numbers in a diagram and open tables or graphs with a single click. Users of your model can then easily view and modify input variables, and view the results, without navigating the details of the model, unless they wish to.

The diagram below contains input nodes on the left side and output nodes on the right side. The details of how the model computes the outputs from the inputs are available inside the “Details” module, for anyone who is interested.





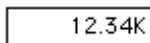
## 9.1 Using input nodes

An *input node* lets you, or your end user, see and easily change the value of a variable directly in the diagram, without opening an attribute view or Object window (see Section 1.3, “Browsing with Input and Output Nodes”). In Browse mode you can change only the values and definitions of input nodes.

An input node is an alias of a variable that you want to treat as an input to the model (see Section 4.7, “Using an alias node”).

The type of definition of the original variable determines the appearance of the input node (see Section 8.3, “The Expression popup menu”). If you want your users to be able to change the type of definition, instruct them on how to open an attribute view or Object window and use the Expression popup menu.

### Input field



A single number or text value (scalar) displays as an input field. You can have Analytica check if the input value is acceptable by using the check attribute (see Section 8.6, “Checking the validity of a variable’s values”); the check is performed on input of a new value.

### Input popup menu



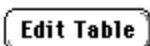
A choice displays as an input popup menu. To create an input menu for an input node, see Section 9.2, “Creating a popup menu”.

### List



A list or list of labels displays as a **List** button (see Section 11.3, “Creating an index”).

### Edit table



An edit table displays as an **Edit Table** button (see “An Edit table is a window that appears similar to a Result table. Unlike a Result table, you can select or change the indexes of the array and enter or edit the value of each element. If you select a variable defined as an Edit table and click on the edit definition button (), you will see its Edit table.”, page 11-4).


 A button with the word "Normal" inside, enclosed in a rounded rectangular border.

### Probability distribution

A probability distribution displays a button with the name of the distribution (see Chapter 13, “Expressing Uncertainty”).

### Creating an input node

To create an input node from a variable:

1. Select the variable.
2. Select **Make Input Node** from the **Object** menu. The input node will appear in the same diagram next to the selected node.
3. Move the input node to the location you want.
4. Adjust the size of the node.

To make several input nodes at once, select the variables and then choose **Make Input Node**.

## 9.2 Creating a popup menu

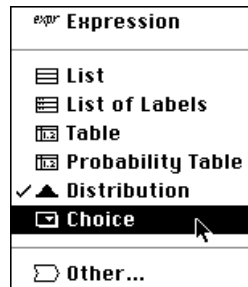
For the classes of nodes that may be used for parametric analysis, such as decision and chance, the Expression popup menu includes the Choice option. The **Choice** option provides a way to offer the user a choice of one or all values from a list.

### Creating a menu from a list

If the original variable is already defined as a list of numbers or labels, create a popup menu to select from the list as follows:

1. Show the definition of the variable as a list, either in the attribute view or the object window.

2. Press the Expression popup menu and select the **Choice** option.



The definition field of the original variable now displays as a popup menu, and in browse mode, the input node displays as a popup menu. The original definition (list of numbers or labels) is now available as the *domain* of the variable—the possible outcomes. In the expression view the popup menu displays as the `Choice()` function (see page 12-20).

**Note:** 

To define *Var1* as a popup menu of another variable *Var2*, that is defined as a list, define *Var1* as `Choice(Var2, 1)` in expression view (see “`Choice(I, n)`” on page 12-20).

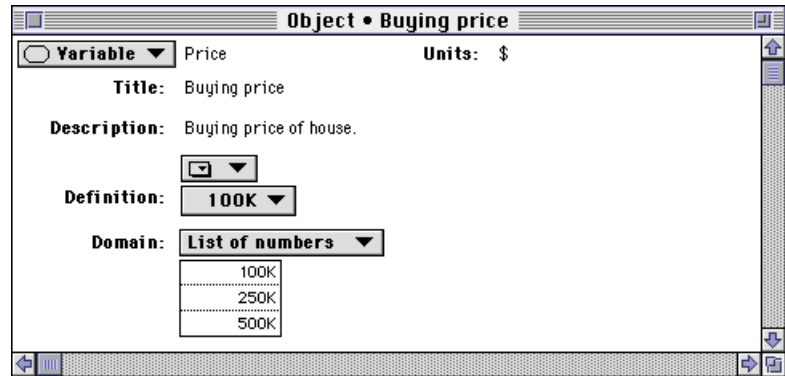
## Creating a new definition


If a variable has no previous definition, when you select **Choice** from the Expression popup menu, a domain (possible outcomes) of List of Labels is created, with one element in the list.

To change the domain to List of Numbers, press the Domain popup menu and select **List of Numbers**.

Edit the list of values as you would edit a list of labels or list of numbers (see “Editing a list” on page 11-15). When you press

*enter*; the definition field becomes a popup menu of the domain values.



**Note:**  The values in the domain are evaluated deterministically.

**Calc** mid

## 9.3 Using output nodes

An *output node* gives you, or your end user, rapid access to a selected result in the model. You can use output nodes to focus attention on particular outputs of interest.

An output node displays a result value in the view style—table or graph, the indexes displayed, and the uncertainty view—last selected for display and saved with the model. It also shows the uncertainty view icon (see Section 2.4, “Uncertainty view options”).

61.73 mid If the result is a single value (mid value or mean), it displays directly in the output field.

**Result** mid

If the result is an array, the output node displays a Result button. Click on the button to display the table or graph.

After you display the table or graph, you can use the result tool palette to change the view.

If the value of an output has not yet been computed, the **Calc** button appears in the node. Click on the **Calc** button to compute and display the value.

## Creating an output node

To create an output node from a variable:

1. In a diagram window, select the node of the variable from which you wish to create an output node.
2. Select **Make Output Node** from the **Object** menu. The output node will appear in the diagram next to the selected node.
3. Move the output node to the location you want.
4. Adjust the size of the node.

The view style of the output result—table or graph—will be the format you last set for it (see Chapter 7, “Formatting Graphs and Tables”).

## 9.4 Changing display style

The title and units of an input or output node are obtained from the original node. To edit them, edit the title and units of the original node (see Section 4.8, “Editing an attribute of a node”). If you edit the title or units of the original node, the input or output node's title or units changes to match the original.

By default, an input or output node shows its original node's title (label) in the original font, with no node outline or arrows. The node takes its color from its original node when the node is created. Later changes to the original node color do not change the color of the input or output node.

To change the appearance of an input or output node alone, use the **Set Node Style...** and **Show Color Palette** options from the **Diagram** menu (see Sections 6.6 and 6.4). When you use these options to change the appearance of an input or output node, its original node does not change. Similarly, using these options to

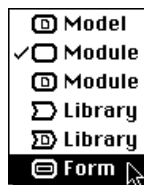
change the appearance of an original node does not affect its previously created input or output node.

## 9.5 Using form modules

It is often helpful to group input and output nodes into a single diagram for easy access by model users. The **form module** makes it easy for you to create input and output nodes in the form by drawing arrows between the form and variables.

To create a form:



1. Make sure you are in a diagram window with the Edit tool selected.
2. Drag the module icon off the node palette and position it in the diagram.
3. Type in a title for the module, e.g. Inputs.
4. Open the attribute view at the bottom of the diagram window.
5. From the attribute popup menu, select **Class**. A popup menu of available classes displays.



6. Select **Form** from the popup menu of classes.

### Creating input and output nodes in a form module

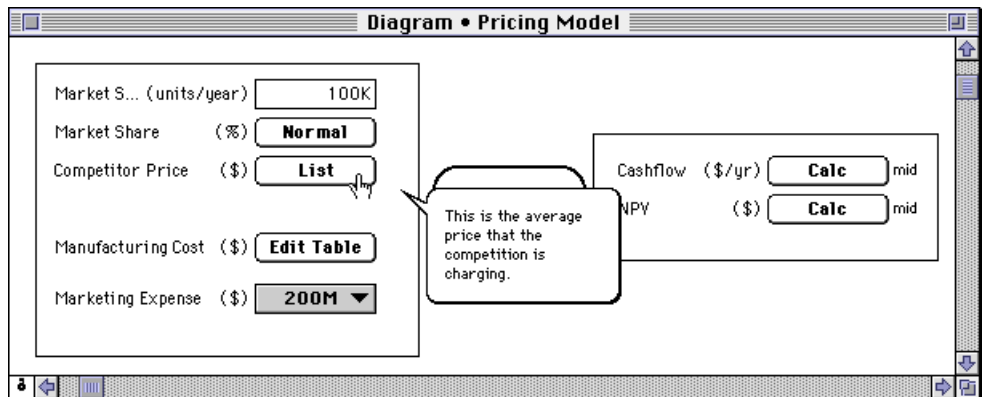
An input or output node is an alias to another variable in the model. Creating an input or output node is similar to creating an alias (see Section 4.6, “Creating alias nodes”). To create a set of input and/or output nodes in the form module:

1. Adjust the diagram(s) on your screen so the form node and the source variables for the input or output nodes are all visible (they can be in the same or different diagram windows).
2. In the tool palette, click on the arrow button ().
3. For input nodes, draw an arrow from the form node to each variable. Analytica creates an input node for each variable inside the form module.
4. For output nodes, select the variables and draw arrows from the variables to the form node. Analytica creates an output node for each selected variable inside the form module.
5. When you have finished creating input and output nodes, double-click on the form node to open its diagram window.
6. In the tool palette, click on the edit button ().
7. Rearrange and resize the input and output nodes for clarity. It is usually clearest to put the input nodes down the left side and the output nodes down the right side.

A form module is like any other module, except when you draw arrows to or from the form module. So you can also create nodes that are not inputs or outputs and modules inside a form. If you have too many nodes to fit comfortably in a single diagram, you can create additional modules (which need not be forms) to enclose related groups of inputs and outputs.

## 9.6 Using balloon help

Balloon help provides a way for model users to learn more about a model by simply moving the cursor around a diagram. When balloon help is on, a balloon containing text about a node appears when you position the cursor over the node. *You do not need to click on the node.* This help text can give the model user details to explain the nature and role of the node.



By default, balloon help displays the text in a variable's description attribute. If you want balloon help text to be different from the description, you can use the Help attribute. (see Section 19.4, “Managing attributes”). When the Help attribute displays, Balloon Help uses its text rather than the description text.

## 9.7 Adding icons to nodes

You can add an icon to any node in a diagram. The Icon window contains an enlarged space that you can use for creating or editing an icon.

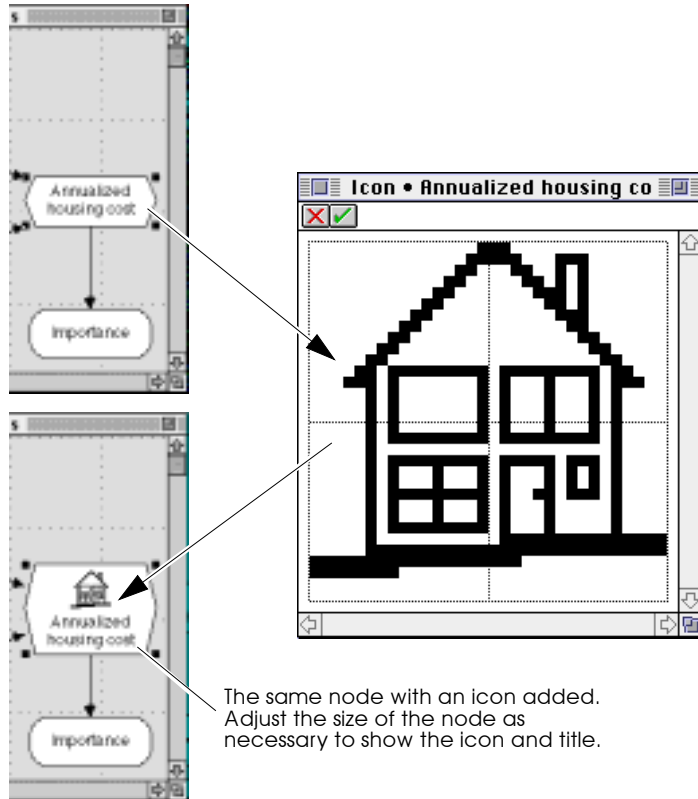
### Opening the Icon window

To add an icon:

1. Make sure that the Edit tool is selected.





2. Select the node that you wish to illustrate.
3. Choose **Edit Icon** from the **Diagram** menu to open the Icon window.



## Drawing or editing an icon

You can draw or edit the icon one pixel at a time using mouse clicks, or you can draw lines by holding down the mouse button as you drag the cursor.

- To make a dark pixel light or a light pixel dark, click on the pixel.

- To set the node's icon, click on the  button.
- To restore the original icon in the window (or to clear the window if there was no previous icon), click on the  button.

You can copy and paste an icon from one place in a model to another using the standard Macintosh **Copy** (⌘-C) and **Paste** (⌘-V) commands.

## 9.8 Adding graphics, frames, and text to a diagram

### Adding graphics


You can add a graphic image created in another application to any node or to the diagram background. Both color bitmaps and PICT graphics can be pasted in.

To paste in a graphic:

1. Copy (⌘-C) the graphic to the clipboard from within a graphics application.
2. Make sure that the Edit tool is selected in Analytica.
3. Select the node or the diagram window where you want the graphic to appear.
4. Paste (⌘-V) the graphic from the clipboard.

When you paste a graphic into the diagram window, a special node of class picture is created. Variable, module, and function nodes can be placed on top of picture nodes.

---

**Note:** 

When you save the model, graphics are saved as resources.

---

To remove a graphic, select it and press *delete*, or choose **Clear** from the **Edit** menu.

**Adding a frame** You can create a rectangular frame for nodes in a diagram in either of the following ways:

- Paste a graphic into the diagram window to create a picture node, then delete the graphic. This leaves a blank picture node. Use the Node Style dialog box (see Section 6.6) to display the border of the node. Other nodes can be placed on top of this node.
- Create a decision node and leave the title blank. Give it a definition of 0 (or any number) to remove the cross-hatch pattern. Use the Node Style dialog box (see Section 6.6) to hide the label and fill color. Create this frame first, then create the nodes to be framed and place them in the frame. If you create a framing decision node after you create the nodes to be framed, the nodes will be “under” the framing decision node; they will be visible, but you will not be able to select them.

**Adding text** You can add a text to a diagram in the following way:

1. Copy (⌘-C) the text to the clipboard from any source.
2. Make sure that the Edit tool is selected in Analytica.
3. Select the diagram window where you want the text to appear.
4. Paste (⌘-V) the text from the clipboard.

When you paste text into the diagram window, a special node of class text is created. Use the Node Style Dialog box (see Section 6.6) to change the font and size.

---

# 10

## Using Expressions

---

This chapter describes the building blocks for creating and editing expressions to define variables: numbers, operators and mathematical functions.

### 10.1 Numbers


The following formats are all valid for entering numbers:

Number Format	Examples
Integers	2, 10, 1234
Decimals	32.5, .0002, 0.000012345
Suffix	250K, 10.5M, 10.5m, 22%
Exponential form	53E11, 1E20, 4.5632E-25

- The signed integer after the E is an exponent that denotes a power of ten. For example:  
 $5E4 = 5 \times 10^4 = 50,000$   
 $4.3E-3 = 4.3 \times 10^{-3} = 0.0043$
- A character suffix denoting a power of ten is a convenient way to express very large or small numbers. For example:  
 $50K \rightarrow 50,000$   
 $1.5m \rightarrow 0.0015$

The character suffixes are the same as used in the default output number format (see page 7-10).

Power of 10	Suffix	Prefix	Power of 10	Suffix	Prefix
3	K	Kilo	-2	%	percent
6	M	Mega or Million	-3	m	milli
9	B	Billion	-6	u	micro (mu)
9	G	Giga	-9	n	nano
12	T	Tera or Trillion	-12	p	pico
15	Q	Quad	-15	f	femto

Note: 

The character suffixes m ( $10^{-3}$ ) and M ( $10^6$ ) are distinct. This is the only situation in which the case of a letter makes any difference for input to Analytica. You can use, for example, k and K interchangeably.

### Range

Analytica on a 68000 version Macintosh can represent zero and numbers with absolute values between about  $10^{-37}$  and  $10^{37}$ . On a Power Macintosh, the range extends from between about  $10^{-320}$  and  $10^{308}$ .

### Numbers out of range

When a calculation results in a number whose absolute value is less than the smallest number that can be represented, Analytica rounds the number to 0 (zero), without warning. For example,

$$1/10^{1000} \rightarrow 0$$

### INF (infinity)

When a calculation results in a number whose absolute value is greater than the largest that can be represented, Analytica displays it as INF or -INF, for positive or negative infinity. For example,

$$10^{1000} \rightarrow \text{INF}$$

$$-10^{1000} \rightarrow -\text{INF}$$

$$1/0 \rightarrow \text{INF}$$

You can enter INF as a value in an expression. Analytica can perform some computations with INF, such as:

$$\text{INF} + 10 \rightarrow \text{INF}$$

$$\text{INF}/0 \rightarrow \text{INF}$$

$$10 - \text{INF} \rightarrow -\text{INF}$$

Other computations with INF, such as difference and ratio, give results that are ill-defined and return NAN (Not A Number) codes:

$$\text{INF} - \text{INF} \rightarrow \text{NAN}(000)$$

$$\text{INF}/\text{INF} \rightarrow \text{NAN}(000)$$

**Precision** The maximum internal precision of numbers is 7 significant digits on a 68000 version Macintosh and 15 significant digits for a Power Macintosh. Some calculations, especially those that involve small differences between numbers, may result in less precision than the maximum.

## 10.2 Text values

You can specify a text value by enclosing text in single quotes, for example:

```
'A', 'A25', 'A longish string - with punct.'
```

A text value can contain any character, including comma, space, new line, except single quote('). You can enter a text value directly as the value of a variable, or in an expression, including as an element of a list (see “Creating an index,” Section 11.3 and “List vs. List of Labels,” Section 11.4) or edit table (see “Creating an array with an Edit table,” Section 11.6). Analytica displays text values in results without the enclosing quotes.

## 10.3 Boolean or logical values

There are two *Boolean* or *logical* values — *True* and *False*. You can specify a Boolean value in an expression as *False* or *True*, or, equivalently, as the numbers, 0 or 1; for example:

False or True → True

1 And 0 → False

Analytica treats every nonzero number as True; for example:

2 And True → True

Analytica displays Boolean results as 0 or 1, by default. To display them as False or True, change the format of the definition or result to Boolean (see “Number Format dialog box,” Section 7.6).

## 10.4 Operators

An *operator* is a symbol, such as a plus sign (+), that represents a computational operation or action such as addition or comparison. Analytica includes the following sets of standard operators.

### Arithmetic operators

The arithmetic operators apply to numbers and produce numbers

Operator	Meaning	Examples
+	plus	3+2 → 5 'a'+5 → 'a5'
-	minus	3-2 → 1
*	multiplied by	3*2 → 6
/, ÷	divided by	3/2 (= $\frac{3}{2}$ ) → 1.5
^	to the power of	3^2 (= 3 <sup>2</sup> ) → 9
	root	4^.5 (= 4 <sup><math>\frac{1}{2}</math></sup> ) → 2

## Comparison operators

The comparison operators apply to numbers and text values and produce Boolean values. Applied to text values, they use the standard ASCII ordering of characters .

Operator	Meaning	Examples (1 = true, 0 = false)
<	less than	2 < 2 → 0 'A' < 'B' → 1
<=, ≤	less than or equal to	2 <= 2 → 1 'ab' <= 'ab' → 1
=	equal to	100 = 101 → 0 'AB' = 'ab' → 0
>=, ≥	greater than or equal to	100 >= 1 → 1 'ab' >= 'cd' → 0
>	greater than	1 > 2 → 0 'A' > 'a' → 1
<>, ≠	not equal to	1 <> 2 → 1 'A' <> 'B' → 1

## Logical operators

The logical operators apply to Boolean values and produce Boolean values.

Operator	Meaning	Examples (1 = true, 0 = false)
<i>b1</i> AND <i>b2</i>	true if both <i>b1</i> and <i>b2</i> are true, otherwise false	1 AND 2 < 2 → 0
<i>b1</i> OR <i>b2</i>	true if <i>b1</i> or <i>b2</i> or both are true, otherwise false	0 OR 1 < 2 → 1
NOT <i>b</i>	true if <i>b</i> is false, otherwise false	NOT (2 < 3) → 0



**If  $B$  Then  $U$  Else  $V$**  This is a conditional expression. If the Boolean expression  $b$  is true, it returns the value of  $u$ . If  $b$  is false, it returns the value of  $v$ . For example:

If  $2 > 4$  Then 100 Else 0  $\rightarrow$  0

In Analytica, the Else part of the expression is required.

An additional conditional operator, `If all ... Then ... Else` exists for arrays; see “Conditional operators” on page 11-26.

## Operator binding precedence

A precedence hierarchy resolves potential ambiguity when evaluating operators and expressions. The hierarchy of precedence for operators, from most tightly bound to least tightly bound is:

functions, not  
 $\wedge$   
 - (unary)  
 $*$ ,  $/$   
 $+$ ,  $-$   
 $<$ ,  $>$ ,  $\leq$ ,  $\geq$ ,  $=$ ,  $<>$   
 and, or  
 If...Then...Else

Within each level of this hierarchy, the operators bind from left to right (left associative).

### Examples:

The following arithmetic expression:

$1 / 2 * 3 - 3 ^ 2 + 4$

is interpreted as:

$((1 / 2) * 3) - (3 ^ 2) + 4$

The following logical (Boolean) expression:

If  $a$  and  $b > c$  or  $d + e < f ^ g$  Then  $x$  Else  $y + z$

is interpreted as:

If  $((a$  and  $(b > c))$  or  $((d + e) < (f ^ g))$ ) Then  $x$   
 Else  $(y + z)$

## 10.5 Math functions

These functions can be accessed under the **Definition** menu **Math** command, or in the Object Finder dialog box, **Math** library.

**Abs ( X )** Returns the absolute value of  $X$ .

$\text{Abs}(180) \rightarrow 180$

$\text{Abs}(-210) \rightarrow 210$

**Arctan ( X )** Returns the arctangent of  $X$  in degrees.

$\text{Arctan}(0) \rightarrow 0$

$\text{Arctan}(1) \rightarrow 45$

Remembering the trigonometric identity,

$\text{Tan}(X) = \text{Sin}(X) / \text{Cos}(X)$ , you get:

$\text{Arctan}(\text{Sin}(45) / \text{Cos}(45)) \rightarrow 45$

**Cos ( X )** Returns the cosine of  $X$ ,  $X$  assumed in degrees.

$\text{Cos}(180) \rightarrow -1$

$\text{Cos}(-210) \rightarrow -0.866$

**Exp ( X )** Returns the exponential of  $X$ , that is,  $e^X$ .  $X$  must not be greater than 85.

$\text{Exp}(5) \rightarrow 148.4$

$\text{Exp}(-4) \rightarrow 0.01832$

**Factorial ( X )** Returns the factorial of  $X$ , which must be between 0 and 33.

$\text{Factorial}(5) \rightarrow 120$

$\text{Factorial}(0) \rightarrow 1$

If  $X$  is not an integer,  $X$  is rounded to the nearest integer before taking the factorial.

**Ln ( X )** Returns the natural logarithm of  $X$ , which must be positive.

$\text{Ln}(150) \rightarrow 5.011$

$\text{Ln}(\text{Exp}(5)) \rightarrow 5$

**Logten ( X )** Returns the logarithm to the base 10 of  $X$ , which must be positive.

$\text{Logten}(180) \rightarrow 2.255$

$\text{Logten}(10 \wedge 30) \rightarrow 30$

**Round ( X )** Returns the value of  $X$  rounded to the nearest integer.

Round(1.8) → 2

Round(-2.8) → -3

Round(1.499) → 1

Round(-2.499) → -2

**Sin ( X )** Returns the sine of  $X$ ,  $X$  assumed in degrees.

Sin(30) → 0.5

Sin(-45) → -0.7071

**Sqr ( X )** Returns the square of  $X$ .

Sqr(5) → 25

Sqr(-4) → 16

**Sqrt ( X )** Returns the square root of  $X$ , which must be positive or zero.

Sqrt(25) → 5

---

# 11

## Modeling with Arrays and Tables

---

The value of a variable may be a *scalar* — a single number, text, or Boolean — or it may be an *array* — a collection of values, viewable as a table with one or more dimensions. The ease and flexibility with which you can create, operate with, and display multidimensional arrays is the source of much of the power of Analytica for creating and managing substantial models. An array's dimensions are identified using *index variables*. You can extend a dimension by adding elements to its index, or add a dimension to an array variable, and the change in dimensions will automatically carry through the rest of the model.

There are some subtleties to the effective use of arrays. Your prior experience with spreadsheets or programming languages may mislead you about how best to use arrays in Analytica. So, if you plan to use arrays in your models, we suggest that you first read Section 11.1, “Introduction to Arrays” and Section 11.2, “Operations on arrays”. The remainder of this chapter provides the details on how to create index variables, how to use Edit tables to create array values, and how the arithmetic, comparison, logical, and conditional operators work with arrays. Chapter 12, “Array Function Reference,” describes the special functions that create and operate on arrays.

### 11.1 Introduction to Arrays

#### What is an array?

An array is a collection of values that you can view as a table or graph. An array has one or more dimensions, which may appear as the row headers or column headers of a table. For example, the


value of variable *Fuel price per gallon* is a one-dimensional array with two values, \$1.50 for the Econocar (which uses regular gasoline) and \$1.70 for the Supercar (which uses premium gasoline):

Car type	
Econocar	\$1.50
Supercar	\$1.70

*Maintenance cost per year* is defined as a two-dimensional array, which varies by *Car type* and by *Year*:

Car type	Year	1	2	3	4	5
Econocar		300	300	500	1000	1400
Supercar		700	700	700	800	900

The Econocar is cheap to maintain initially, but gets more expensive than the Supercar after 3 years, as its components start to wear out and need replacing.

**Note:**  You can swap the rows (*Car type*) and columns (*Year*) by using the row or column popup menus (see “Index selection area” on page 2-3).

## What is an index?

Each dimension of an array is identified by an index variable. The index variable holds the possible values, either a list of numbers or a list of labels. In the examples above, *Car type* is a list of labels, ‘Econocar’ and ‘Supercar’. *Year* is a list of numbers.

Car type:
Econocar
Supercar

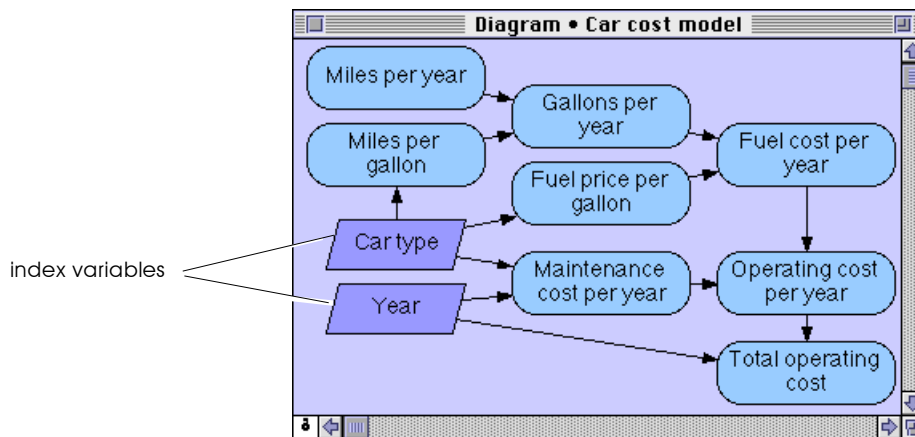
Year:
1
2
3
4
5

To create an index, see Section 11.3, “Creating an index”.


Before creating an array, it is usually best to create the indexes for the array’s dimensions. An index may be used in multiple arrays. When building a model that will use several multidimensional arrays, a key task is to define the indexes.

## Index variables in a diagram

Below is a diagram of a *Car cost model*, which includes the variables described above. The two index variables are shown as parallelogram nodes on the diagram.




*Fuel price per gallon* is the destination of an arrow from *Car type* because it is defined as an array indexed by *Car type*. Similarly, *Maintenance cost per year* has arrows from *Car type* and from *Year*, because it is indexed by both.

Note: 

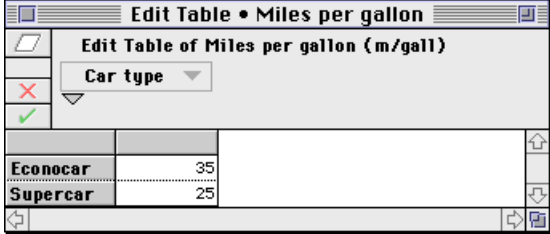
By default, Analytica does not show arrows to and from index variables. You can display these arrows, as in this example, by selecting the option in the Diagram Style dialog from the **Diagram** menu (see Section 6.5, “Diagram Style dialog box”).

## Viewing an array as an Edit table

An *Edit table* is a window that appears similar to a Result table. Unlike a Result table, you can select or change the indexes of the array and enter or edit the value of each element. If you select a variable defined as an Edit table and click on the edit definition button () , you will see its Edit table.

Example (continued from above):

*Miles per gallon*:



Edit Table of Miles per gallon (m/gall)	
Car type	
Econocar	35
Supercar	25

To create or edit an array with an Edit table, see Section 11.6, “Creating an array with an Edit table”.

## Two sources of array value

When you evaluate a variable and its Result window shows an array value, there are two possible sources. A variable will have an array value if:

- it is defined as an array using an Edit table, or
- it is defined as an expression calculated from one or more other array-valued variables.

## 11.2 Operations on arrays

Arithmetic operations and simple functions generalize straightforwardly when they are applied to arrays, according to the dimensions of the arrays. This section gives some simple examples.

### Operation on a scalar and an array

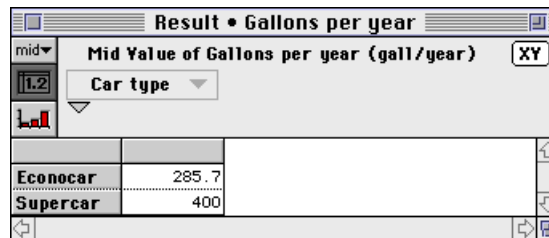
An operation applied to a scalar and an array results in an array of the same shape, applying the scalar operation to each element in the array.

**Example (continued):**

*Miles\_per\_year*: 10K (a scalar)

*Gallons per year*: *Miles\_per\_year* / *Miles\_per\_gallon*

The result of an operation (division in this case) combining a scalar and an array is a result array with the same index(es) as the original array:



Result • Gallons per year	
Car type	Mid Value of Gallons per year (gall/year)
Econocar	285.7
Supercar	400

### Operation on two arrays with the same indexes

An arithmetic operator applied to two arrays with the same indexes creates another array with the same indexes. Analytica applies the operator to pairs of corresponding elements.

**Example (continued):**

*Fuel cost per year*:

*Fuel\_price\_per\_gall* + *Gallons\_per\_year*

Both *Fuel price per gallon* and *Gallons per year* are arrays with the same index, *Car type*. The result is an array also indexed by *Car*



*type*, containing the value obtained by multiplying the corresponding elements of each array:

Result • Fuel cost per year	
Mid Value of Fuel cost per year (\$/year) XY	
Car type	
Econocar	\$429
Supercar	\$680

## Operation on a one- and two-dimensional array

An arithmetic operator applied to a one-dimensional array and a two-dimensional array, that have one index in common, creates another two-dimensional array with the same two indexes.

### Example (continued):


*Op\_cost\_per\_year*:

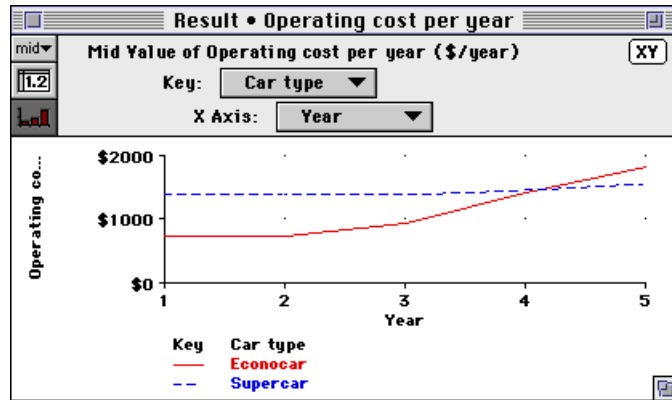
$$\text{Fuel\_cost\_per\_year} + \text{Maintenance\_per\_year}$$

*Operating cost per year* is the sum of a one dimensional variable indexed by *Car type* and a two-dimensional variable indexed by *Car type* and *Year*. The result is a two-dimensional array indexed by both indexes:

Result • Operating cost per year					
Mid Value of Operating cost per year (\$/year) XY					
Car type					
Year					
	1	2	3	4	5
Econocar	\$729	\$729	\$929	\$1429	\$1829
Supercar	\$1380	\$1380	\$1380	\$1480	\$1580

Each *Car type* (row) in the result uses the fuel cost and maintenance cost for the corresponding *Car type*. Each *Year* (column) uses the same annual fuel cost, which does not change by year, and the corresponding maintenance cost, which does change by year.

Changing the above table to a graph, using the graph button () shows:



The graph shows how the operating costs of the Econocar are less than the costs of Supercar in the first 3 years and grow to be larger in the 5th year, crossing over just after the 4th year.

## Summing over an index variable


The `Sum()` function sums an array over one index, giving a result without that index.

**Example (continued):**

*Total operating cost:* `Sum(Op_cost_per_year, Year)`

This operation sums *Operating cost per year* over the *Year* dimension, producing a result indexed only by the *Car type* dimension:

Result • Total operating cost	
Mid Value of Total operating cost (\$)	
Car type	
Econocar	\$5643
Supercar	\$7200

**Note:**  The expression does not need to mention any other possible indexes, such as *Car type*.

Because the `Sum()` function eliminates one index of an array, it is called an array-reducing function. Analytica includes several array-reducing functions (see Section 12.4).

## Operation on arrays with different dimensions

An arithmetic operator applied to two one-dimensional arrays with different indexes creates a two-dimensional array with both indexes.

### Example (continued):

*Miles per year* is redefined as a list (see “Creating a list” on page 11-10):

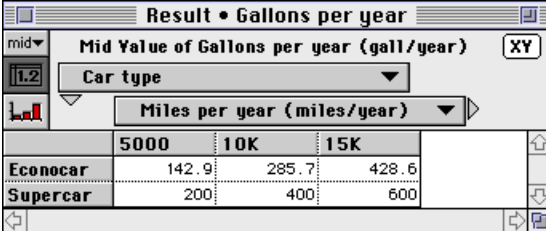
*Miles\_per\_year*: [5000, 10K, 15K]

Being defined as a list means that it is like a one-dimensional array indexed by itself. It is eligible to be an index of arrays that are computed from it.

The definitions of *Miles\_per\_gallon*, an array indexed by *Car type*, and *Gallons per year*, a ratio, remain unchanged.

*Gallons per year*: *Miles\_per\_year* / *Miles\_per\_gallon*

The result of *Gallons per year* is now an array indexed by both *Miles per year* and *Car type* (compare page 11-5):




	Miles per year (miles/year)		
	5000	10K	15K
Econocar	142.9	285.7	428.6
Supercar	200	400	600

Each value in the table is computed from the *Miles per year* for the column divided by the *Miles per gallon* for each *Car type* (row). For

example, 5000 miles per year divided by Supercar's 25 miles per gallon gives 200 gallons per year.

The list value for *Miles per Year* propagates as a new dimension to all its dependent variables. Recomputing the result for *Operating cost per year* gives a three-dimensional table; it is now indexed by *Miles per Year*:

	1	2	3	4	5
5000	\$514	\$514	\$714	\$1214	\$1614
10K	\$729	\$729	\$929	\$1429	\$1829
15K	\$943	\$943	\$1143	\$1643	\$2043

Here is the result for the other *Car type*, displayed by clicking on the diagonal arrow (  ):

	1	2	3	4	5
5000	\$1040	\$1040	\$1040	\$1140	\$1240
10K	\$1380	\$1380	\$1380	\$1480	\$1580
15K	\$1720	\$1720	\$1720	\$1820	\$1920



## General rule for operations on arrays

We can summarize and generalize the behavior of an operation on two arrays with the following rule: An operation on two arrays yields an array whose indexes are the union of the indexes of the two arrays. In this way, Analytica combines arrays without requiring explicit iteration over each index. We call this feature of generalized operations for multidimensional values **intelligent arrays™**.

## 11.3 Creating an index

Analytica includes a specific class of variable node — the *index variable* — to identify dimensions of arrays. Other variables, such as decision nodes are often also used to identify dimensions of arrays. Actually, any variable defined as a list (one-dimensional array) can serve as an index to an array. For clarity in your model diagram, use the index variable whenever possible. (The terms index and index variable are used interchangeably here.)

Create an index variable in the following way:

1. Select the Edit tool () and have the Diagram window active.
2. Drag the parallelogram shape () from the node palette to the diagram.
3. Give the new index a descriptive title.
4. Define the index as a list, list of labels, or sequence (see below).

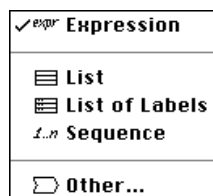
**Creating a list** To define a variable as a list, first select the variable and open one of the following:

- The variable's Object window.
- The Attribute panel of the diagram (see page 1-14).

In the Attribute panel, select **Definition** from the Attribute popup menu (see page 1-15) as the attribute to display.

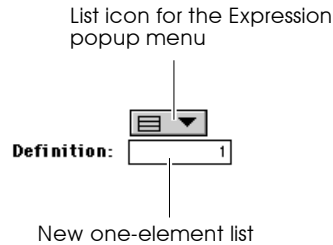
To create a list:

1. Press the Expression popup menu above the definition field and select **List** (for numbers) or **List of Labels** (for text).

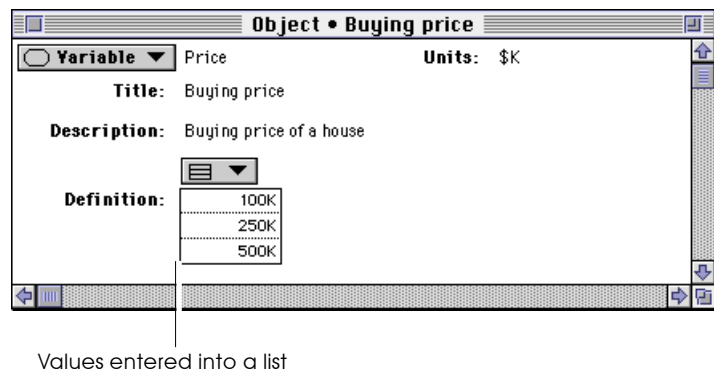


(If the variable already has a definition, Analytica confirms that you wish to replace it. Click on **OK** to replace the definition with a one-element list.)

A one-element list is displayed in the definition field.



2. Select the element by clicking on it.
3. Type in a number or expression (for **List**) or text (for **List of Labels**).
4. Press *return* and type in the next value.
5. Repeat step 4. until you have entered all the values you want.



**Autofilling a list** Analytica gives the first cell of a list the default value 1 or the value of the variable's previous definition. When you press *return*, Analytica gives the second cell the value of the first cell plus 1.

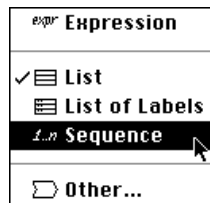
After you have entered at least two values, Analytica gives each new cell a value that is incremented by the difference between the last two values.

### Autofilling a list of labels

Analytica gives the first cell of a list of labels the default text value `item 1`. Analytica gives each subsequent cell receives a value the same as the previous cell.

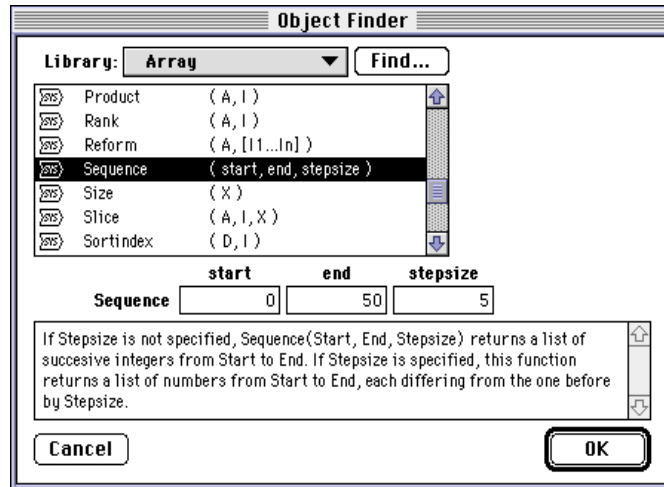
### Creating a list with the Sequence option

For the classes of nodes that are often defined as lists, such as index and decision variables, the Expression popup menu includes the **Sequence** option.




The **Sequence** option provides a quick way to define a list of equally spaced numbers.

When you select **Sequence**, the Object Finder opens, showing the `Sequence()` function (see page 12-5).



After entering the *Start*, *End*, and *Stepsize* values, click **OK**; the definition field shows the **Sequence** button with its parameters.

Definition:  **Sequence** ( 0, 50, 5 )

**Note:**  To edit the Sequence, click on the **Sequence** button.




## 11.4 List vs. List of Labels

You can display a list or list of labels in two ways: “List View” or “Expression View”. The “List View” displays by default; the Expression popup menu shows the list or list of labels icon.

**List View:**

1
2
3
4
5

The “Expression View” displays when you select  in the Expression popup menu.

**Expression View:** [ 1 , 2 , 3 , 4 , 5 ]

### List (of numbers)

In a list of numbers (usually called simply a list), each value is a number or an expression that evaluates to a number. For example, the sequence of five integers above is a list.

### List of labels

In a list of labels every value is text. For example, the set of states below is a list of labels; in the expression view, each label is contained in single quotation marks.

**List view:**

Alabama
Alaska
Arizona
Arkansas

**Expression view:** [ 'Alabama' , 'Alaska' ,  
'Arizona' , 'Arkansas' ]

To include a single quote (apostrophe) as part of the text in a label, you must use the “smart” quote (') created with the keyboard combination *option-shift-]*.

## Mixing numbers and text

A list can include a mix of cells containing text and numbers. In both views the text is contained in single quotation marks. For example:

List view:

1
'Alabama'
2
'Alaska'


Expression view: [1, 'Alabama', 2, 'Alaska']

If you attempt to mix numbers and text in a list of labels, all the values will be treated as text. For example:

List view:

1
Alabama
2
Alaska

Expression view: ['1', 'Alabama', '2', 'Alaska']

Note: 

A list cell can contain any valid expression, including one that refers to other variables or one that evaluates to an array.

## 11.5 Editing a list

You can edit a list by changing, adding or deleting *cells* (list items).

### Inserting cells

To add a cell at the end of the list, select the last cell and press *return* or the down arrow key.

To insert a cell anywhere other than at the end of the list, select a cell and press *option-return* or choose **Insert Rows** (⌘-I) from the **Edit** menu. The value in the selected cell is duplicated in the new cell.

To insert several contiguous cells in the middle of the list, select the number of cells you want to insert and choose **Insert Rows** (⌘-I) from the **Edit** menu. The value of the last selected cell is duplicated in the new cells.

### Deleting cells

To delete one or more cells, select them and do one of the following:

- Choose **Delete Rows** (⌘-K) from the **Edit** menu.
- Press *delete*.

---

#### Note:

If you add or delete a cell in a list that is an index of an edit table, the corresponding elements of the table are also added or removed (see “Editing a table” on page 11-20).

---

### Navigating a list

Use the up and down arrow keys to move the cursor up and down the list.

## 11.6 Creating an array with an Edit table

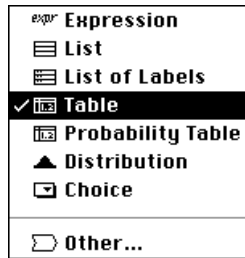
To define a variable as an array (table), first select the variable and open one of the following:

- The variable’s Object window.
- The Attribute panel of the Diagram window (see page 1-14).

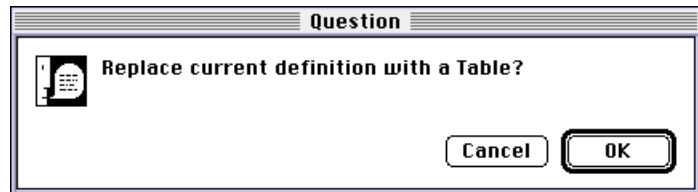
In the Attribute panel, select **Definition** from the Attribute popup menu (see page 1-15) as the attribute to display.

To create a table:

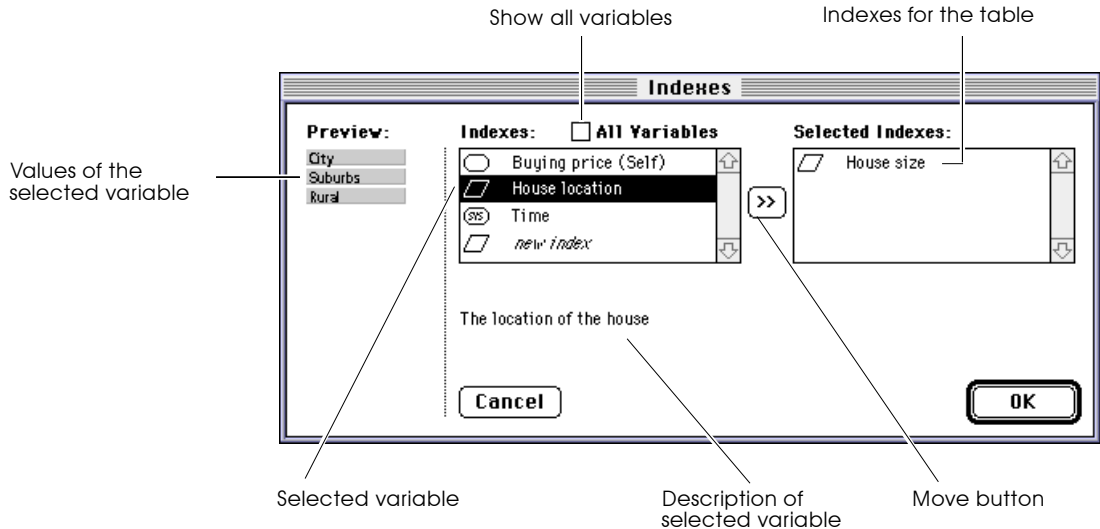
1. Press the Expression popup menu above the definition field and select **Table**.




If the variable already has a definition, you are asked to confirm that you wish to replace it.



2. The Indexes dialog box displays for selecting the table's indexes (dimensions).



3. Select a variable from the Indexes list and click on the move button () , or double-click on the variable, to select it as an index of the table. Repeat for each index you want.
4. Click on **OK** to create the table and open the Edit Table window for editing the table's values (see Section 11.7).

**Indexes dialog box** The Indexes dialog box contains (see figure above):

**Preview** A list of the values of the selected index variable. If the selected variable is not a list, it says “Can’t use as index.”

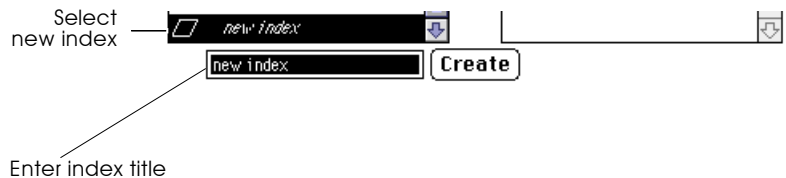
**All variables checkbox** If checked, the Indexes list includes all variables in the model. If not checked, it lists only variables of the class Index and Decision, plus the variable being defined (*Self*) and *Time*. If you select this variable (*Self*) as an index, the variable itself holds the alternative index values.

<b>Selected indexes</b>	A list of all indexes already selected for this variable.
<b>new index</b>	Select to create a new index.

## Creating a new index

You can create an index variable in the course of creating a table, in the following way:

1. Select **new index** from the variables list in the Indexes dialog box.
2. Enter a title for the index.



3. Click on the **Create** button.
4. To make the new index an index of the table, click on the  button.

Enter the values of the Index in the Edit Table window (see the following section).

## Removing an index

To remove an index from a table:

1. Select the index from the Selected Indexes list.
2. Click on the  button.

Removing an index will leave the first table (slice) along that index as the value of the array.

## System index variables *Run* and *Time*

Analytica includes two system index variables: *Run* and *Time*. You can generally treat these index variables like any other index variable.

*Run* is the index for the array of sample values for probabilistic simulation. You can examine the array with the Sample uncertainty mode (see “Sample” on page 2-13) or the `Sample()` function (see page 16-7).

*Time* is the index for dynamic simulation. It is the only index permitted for cyclically dependent modeling (see Chapter 17, “Modeling Changes over Time”).

## 11.7 Editing a table

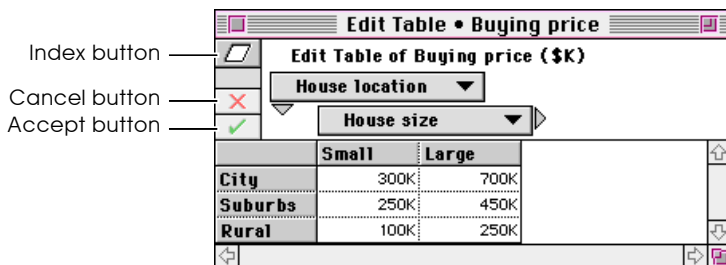
To open the Edit Table window, click on the **Edit Table** button in either:

- The Object window (shown above)
- The Attribute panel of the diagram (see page 1-14)

In the Attribute panel, select **Definition** from the Attribute popup menu (see page 1-15).

### The Edit Table window


The Edit Table window appears similar to the Result window Table view (see page 2-5). The difference is that you can add indexes and edit (change) the values in cells in an Edit Table window.



<b>Selecting cells</b>	<b>Select a single cell</b>	Click on the cell once.
	<b>Select multiple cells</b>	Drag the mouse from one cell to another to select a rectangular region.

**Editing a cell** Enter an expression into a cell in the same manner as you would in a definition field. Press *return* to accept the value and to select the next cell.

---

**Note:**  Edit tables are not designed for defining complex expressions in each cell. Rather than define a cell as a complex expression, create a new variable, and define it as the complex expression. Then enter the new variable's identifier in the edit table cell.

---

**Adding and deleting cells** To add cells by adding rows or columns to a table:

1. Select the row (or column) before which you wish to insert a new one.
2. Choose **Insert Rows** (or **Insert Columns**) from the **Edit** menu.
  - To insert a row (or column) after the selected row or column, press *option-return* (or *option-tab*).
  - To add a new row (or column) after the last one, select the last one and press *return*, or the down arrow key (or tab, or the right arrow key).

The added cells contain zeros.


To delete cells by deleting rows (or columns) in a table:

1. Select the row (or column) you wish to delete.
2. Choose **Delete Rows** (or **Delete Columns**) from the **Edit** menu.

You can only add or delete an element of an index in this way if the index was defined as a list or list of labels. You cannot modify



an index defined as a `Sequence()` or other expression from an edit table.

**Note:** 



When you change an index in this way, you will also affect any other arrays using this index.

---


## Copying and pasting cells

You can copy a cell or a range (two dimensional rectangular region) of cells from a table.

To copy a cell or region:


1. Select the cell or region.
2. Choose **Copy** from the **Edit** menu (-C).
3. Paste the item(s) into another cell or region by selecting **Paste** from the **Edit** menu (-V). The region you paste into must either be a single cell at the top left corner of the destination region, or it must have the same size as the copied region.


## Adding or removing indexes

Click on the Index button () to add more indexes (increasing the number of dimensions) or to remove indexes (decreasing the number of dimensions). The Indexes dialog box appears (see page 11-18).

Any new index of size  $n$  copies the current table with its current values  $n$  times along the new dimension.

## Saving the table

Click on the Accept button () to perform a syntax check and store changes you have made.

Click on the Cancel button () to discard changes made since opening the window, or since the last time you clicked on the Accept button.

If you close an Edit Table window without clicking on the Accept or Cancel buttons, the changes are accepted.

## 11.8 Calculating with arrays

### Conventions for array examples

Most of the examples in this and the next chapter show variables defined as tables (arrays) or evaluating to arrays. Indexes and arrays in these examples are represented as follows:

- An index or list and its values

*IndexName:*

<i>value1</i>	<i>value2</i>	<i>valueN</i>
---------------	---------------	---------------

- An expression that evaluates to a scalar or an array

*expression* → *result*

- A one-dimensional array

*Index\_a* ►

	<b>a</b>	<b>b</b>	<b>c</b>
	<i>value</i>	<i>value</i>	<i>value</i>

- A two-dimensional array

*Index\_b* ▼, *Index\_a* ►

	<b>a</b>	<b>b</b>	<b>c</b>
<i>x</i>	<i>value</i>	<i>value</i>	<i>value</i>
<i>y</i>	<i>value</i>	<i>value</i>	<i>value</i>
<i>z</i>	<i>value</i>	<i>value</i>	<i>value</i>

- A three-dimensional array

*Index\_a* ▼, *Index\_b* ►, *Index\_c* displayed value ⇅

	<b>a</b>	<b>b</b>	<b>c</b>
<i>x</i>	<i>value</i>	<i>value</i>	<i>value</i>
<i>y</i>	<i>value</i>	<i>value</i>	<i>value</i>
<i>z</i>	<i>value</i>	<i>value</i>	<i>value</i>

**Math functions** Any of the math functions that take a single parameter (see Section 10.5) can be applied to an array, resulting in an array of the same shape. Each element of the resulting array is calculated by applying the function to the corresponding element of the input array.

This example takes the square root of every value in a one-dimensional array.

`Sqrt([1, 2, 3, 4, 5])` →

	1	2	3	4	5
	1.000000	1.414214	1.732051	2.000000	2.236068

**Arithmetic operations** An arithmetic operator (+, -, \*, /, ^) applied to two arrays results in an array indexed by every index variable in the two input arrays. Several examples illustrate this:

- An arithmetic operator applied to a scalar and an array results in an array of the same shape, applying the scalar operation to each element in the array:

X:

2	3	4
---	---	---

`10 * X ^ 2` →

X ►

	2	3	4
	40	90	160

- An arithmetic operator applied to two arrays that are both indexed by the same variable creates another array indexed by the same variable with the operator applied to pairs of corresponding elements:

`X + Sqr(X)` →

X ►

	2	3	4
	6	12	20

- An arithmetic operator applied to two arrays with different index variables (or with no index variables) creates an array indexed by every index variable in the two arrays, with the operator applied to all pairs of elements:

*Inflation:*

1.05	1.10	1.15
------	------	------

*Price:*

5K	10K	15K
----	-----	-----

*Inflation \* Price* →

*Price* ▼, *Inflation* ►

	<b>1.05</b>	<b>1.10</b>	<b>1.15</b>
<b>5000</b>	5250	5500	5750
<b>10K</b>	10.5K	11K	11.5K
<b>15K</b>	15.75K	16.5K	17.25K

## 11.9 Comparison and logical operations

The comparison operators (>, >=, =, <=, < and so on) and the logical operators (And and Or) combine array values in the same way as the arithmetic operators. (See section 10.4 for the full list of operators.) The only difference from the arithmetic operators is that both comparison and logical operators return arrays of Boolean values, and the logical operators treat their operands as Boolean. Each cell contains either 1 (*True*) or 0 (*False*) (see Section 10.3, “Boolean or logical values”).

For example:

*Vw\_price:*

8250	10K	15K
------	-----	-----

*Honda\_price:*

12.5K	15K	20K
-------	-----	-----

Honda\_price > Vw\_price →  
 Vw\_price ▼, Honda\_price ►

	12.5K	15K	20K
8250	1	1	1
10K	1	1	1
15K	0	0	1

Honda\_price = Vw\_price →  
 Vw\_price ▼, Honda\_price ►

	12.5K	15K	20K
8250	0	0	0
10K	0	0	0
15K	0	1	0

Honda\_price > VW\_price OR Honda\_price = Vw\_price →  
 Vw\_price ▼, Honda\_price ►

	12.5K	15K	20K
8250	1	1	1
10K	1	1	1
15K	0	1	1

## 11.10 Conditional operators

The conditional operators are :

If  $B$  then  $U$  else  $V$  and Ifall  $B$  then  $U$  else  $V$ .

**If  $B$  Then  $U$  Else  $V$**  Returns an array whose cells contain values of  $U$  or  $V$ , depending on the value of  $B$ .

- If  $B$  has the value *True* (or any nonzero number) or is an array containing only *True* (or nonzero) values, it returns the value of  $U$  and does not evaluate  $V$ .
- If  $B$  has the value *False* (or 0) or is an array containing only *False* (or 0) values, it returns the value of  $V$  and does not evaluate  $U$ .
- If  $B$  is an array which contains at least one *True* (nonzero) value and at least one *False* (0) value, it evaluates both  $U$  and  $V$ . It returns an array indexed by the union of the indexes of  $B$ ,

$U$ , and  $V$ , containing elements from  $U$  or  $V$  according to the corresponding elements of  $B$ . In this case, `If` and `Ifall` give the same result.

### Examples:

$N$ :

1	2	3
---	---	---

If  $N > 0$  Then 'Yes' Else 'No' → 'Yes'

If  $N = 2$  Then 'Yes' Else 'No' →

$N$  ►

	<b>1</b>	<b>2</b>	<b>3</b>
	'No'	'Yes'	'No'

If  $N < 0$  Then 'Yes' Else 'No' → 'No'

## Avoiding Evaluation

You may want to avoid evaluation of  $U$  for elements of  $B$  that give undefined results. For example:

$Myarray$ :

$In2$  ►

	<b>21</b>	<b>22</b>	<b>23</b>
	-10	0	10

If  $Myarray > 0$  Then  $\text{Ln}(Myarray)$  Else 0 gives a warning message on evaluating  $\text{Ln}(-10)$ . Ignoring the message gives

$In2$  ►

	<b>21</b>	<b>22</b>	<b>23</b>
	0	0	2.303

To avoid evaluation of  $U$  for the elements that are false, evaluate `If...Then...Else` on each element of  $Myarray$  using a `For...Do` loop (see page 12-31).

```
For Temp := In2 Do
  If Myarray[In2=Temp] > 0
    Then Ln(Myarray[In2=Temp])
    Else 0
```

**Ifall  $B$  Then  $U$  Else  $V$**  Ifall is similar to If, except that it always evaluates both  $U$  and  $V$ , and returns a value whose dimensions are always the same — the union of the indexes of  $B$ ,  $U$ , and  $V$ . If and Ifall give the same result when  $B$  is an array whose values are partially true.

### When to use:

When  $B$  is an array, the number and identity of the dimensions of the result of evaluating an If expression can vary according to the values in  $B$ . Use Ifall instead of If to ensure that the dimensions of the result are always the same.

### Examples:

Ifall  $N > 0$  Then 'Yes' Else 'No' →

$N$  ►

	1	2	3
	'Yes'	'Yes'	'Yes'

Ifall  $N = 2$  Then 'Yes' Else 'No' →

$N$  ►

	1	2	3
	'No'	'Yes'	'No'

Ifall  $N = 'A'$  Then 'Yes' Else 'No' →

$N$  ►

	1	2	3
	'No'	'No'	'No'

---

# 12

## Array Function Reference

---

Analytica performs operations on arrays without your needing to explicitly identify or iterate over the dimensions of each array. You can use an array variable in the definition of other variables as if it were a single value. For example, if variable  $C$  is defined as  $(A + 2*B)$ , the result will be calculated whether  $A$  and  $B$  are scalars or arrays, and whether their dimensions have the same or different indexes. As a result, you can define an array and its dimensions once, then use the array in calculating other variables without concern for the size or dimensionality of the array.

### 12.1 Overview

This chapter describes Analytica's functions for creating and manipulating arrays. It is organized by the type of function:

- functions that create lists (Section 12.2).
- functions that create arrays (Section 12.3).
- functions that reduce an array to another array with one fewer dimension (Section 12.4).
- functions that return an array with the same number of dimensions as the input array (Section 12.5).
- functions that select part of an array (Section 12.6).
- functions that interpolate values (Section 12.7).
- other array functions (Section 12.8).
- matrix functions (Section 12.9).
- control functions (Section 12.10).



The commonly used array functions are included in the Array Library; the interpolation, matrix, and less commonly used functions are in the Special Library.

Function	See Page	Array Library	Special Library
Area()	12-12	✓	
Argmax()	12-13		✓
Array()	12-7	✓	
Average()	12-13	✓	
Choice()	12-20	✓	
Concat()	12-26	✓	
Cubicinterp()	12-24		✓
Cumproduct()	12-17	✓	
Cumulate()	12-17	✓	
Decompose()	12-28		✓
Determinant()	12-29		✓
Determtable()	15-12	✓	
For...Do	12-31		✓
Ident[I=U]	12-21		✓
Integrate()	12-18	✓	
Invert()	12-30		✓
Linearinterp()	12-24		✓
Max()	12-14	✓	
Min()	12-14	✓	
Normalize()	12-18	✓	
Product()	12-14	✓	
Rank()	12-19	✓	
Sequence()	12-5	✓	
Size()	12-27	✓	
Slice()	12-21	✓	
Sortindex()	12-27	✓	
Stepinterp()	12-25		✓
Subindex()	12-15		✓
Subscript()	12-22	✓	
Subset()	12-28	✓	
Sum()	12-16	✓	
Table()	12-10	✓	
Transpose()	12-30		✓
Uncumulate()	12-19	✓	
Using...Do	12-33		✓

**Examples:** The examples in this chapter refer to the following variables:

*Car\_type:*

VW	Honda	BMW
----	-------	-----

*Years:*

1985	1986	1987	1988
------	------	------	------

*Mpg:*

26	30	35
----	----	----

*Time:*

0	1	2	3	4
---	---	---	---	---

*Cost: Mpg ▼, Car\_type ►*

	VW	Honda	BMW
26	2185	2810	3435
30	1705	2330	2955
35	1585	2210	2835

*Car\_prices: Car\_type ▼, Years ►*

	1985	1986	1987	1988
VW	8000	9000	9500	10K
Honda	12K	13K	14K	14.5K
BMW	18K	20K	21K	22K

*Cost\_in\_time: Mpg ▼, Time ►, Car\_type = VW ↓*

	0	1	2	3	4
26	2185	2294	2409	2529	2656
30	2810	2951	3098	3253	3416
35	3435	3607	3787	3976	4175

*Cost\_in\_time: Mpg ▼, Time ►, Car\_type = Honda ↓*

	0	1	2	3	4
26	2385	2314	2529	2649	2856
30	2910	3041	3238	3343	3526
35	3535	3847	3897	4166	4365

*Cost\_in\_time: Mpg* ▼, *Time* ►, *Car\_type = BMW* ↓

	0	1	2	3	4
26	3185	3294	3409	3529	3656
30	3810	3951	4098	4253	4416
35	4435	4607	4787	4976	5175

## Relating indexes to values

The dimensions (indexes) of a multidimensional array are ordered from *outermost* to *innermost* as they appear in the function viewed as an expression. The outermost is the dimension that changes most slowly; the innermost is the dimension that changes most rapidly. For example, the following expression defines a three-dimensional array indexed by *Index\_a*, *Index\_b*, and *Index\_c* that contains the eight values in the value list. Each value's subscript shows its position in the resulting array:

```
Table(Index_1, Index_2, Index_3)
(value_ace, value_acf, value_ade, value_adf, value_bce,
value_bcf, value_bde, value_bdf)
```

*Index\_1*:

a	b
---	---

*Index\_2*:

c	d
---	---

*Index\_3*:

e	f
---	---

*Index\_1* ▼, *Index\_2* ►, *Index\_3 = e* ↓

	c	d
a	value_ace	value_ade
b	value_bce	value_bde

*Index\_1* ▼, *Index\_2* ►, *Index\_3 = f* ↓

	c	d
a	value_acf	value_adf
b	value_bcf	value_bdf

*Index\_1* is the outermost dimension, whose index value changes least rapidly in the value list. In other words, the value of *Index\_1* stays *a* for the first half of the value list while the values of the other indexes change, then the value of *Index\_1* changes to *b*.

*Index\_3* is the innermost dimension, whose index value changes with every subsequent value in the value list. *Index\_2* is the intermediate dimension, changing more rapidly than *Index\_1*, but less rapidly than *Index\_3*.

## 12.2 Functions that create lists

Use the **List** option in the Expression popup menu to define a variable as a list of numbers or text values (labels) (see “Creating a list” on page 11-10). You can also create a list within a variable definition using the functions described below.

**[ *u1, u2, u3, ... um* ]** The list of expressions, separated by commas and surrounded by brackets, creates a list, whose values are *u1, u2, u3, ... um*.

Using square brackets to specify a list directly as an expression is equivalent to using the **List** or **List of Labels** options in the Expression popup menu, as described on page 11-10, according to the type of values.

**Examples:**            [ 8000 , 12K , 15K ]  
                              [ 'VW' , 'Honda' , 'BMW' ]

**Sequence ( *Start, End, Stepsize* )** Creates a list of numbers increasing or decreasing from *Start* to *End* by increments (or decrements) of *Stepsize*. *Stepsize* is optional and must be a positive number; if omitted, Analytica uses increments of 1. *Start, End,* and *Stepsize* must be deterministic scalar numbers, not arrays.

Using this function is equivalent to using the **Sequence** option in the Expression popup menu, as described on page 11-12.

**Library:**            Array

**Examples:**

If *End* is greater than *Start*, the sequence is increasing:

Sequence(1, 5) →

List view:

1
2
3
4
5

Expression view: [1, 2, 3, 4, 5]

If *Start* is greater than *End*, the sequence is decreasing:

Sequence(5, 1) → [5, 4, 3, 2, 1]

If *Start* and *End* are not integers, and if *Stepsize* is not specified, Analytica rounds them first:

Sequence(1.2, 4.8) → [1, 2, 3, 4, 5]

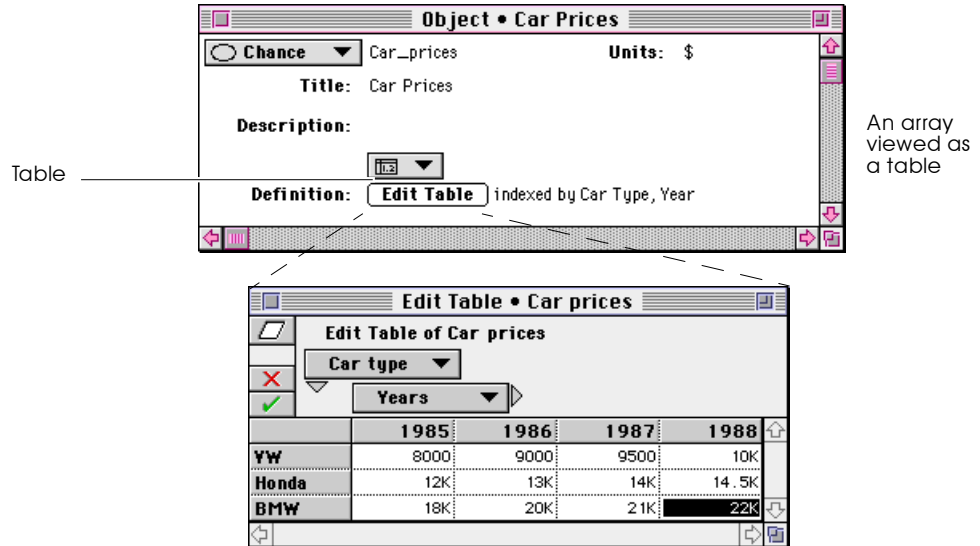
If *Stepsize* is specified, Analytica can create non-integer values from *Start* to *End* with the *Stepsize*:

Sequence(0.5, 2.5, 0.5) → [0.5, 1, 1.5, 2, 2.5]

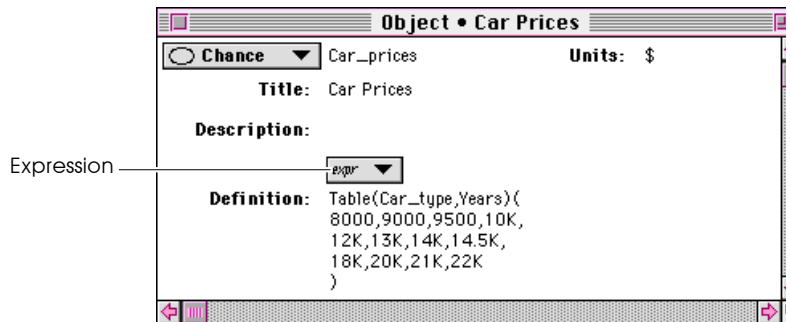
## 12.3 Functions that create arrays

Use the **Table** option in the Expression popup menu to define a variable as an array (see Section 11.6, “Creating an array with an Edit table”).

For more flexibility and control, you can define a variable as an array by entering the `Array()` or `Table()` function as an expression.



An array viewed as an expression appears in the `Table()` function syntax:



**Array(*I1*, *I2*, ... *In*, *A*)** Assigns a set of indexes, *I1*, *I2*, ... *In*, as the indexes of the array *A*, with *I1* as the index of the outermost dimension (changing least rapidly), *I2* as the second outermost, and so on. (See “Relating indexes to values” on page 12-4.) *A* must have at least *n* dimensions. The elements of *A* are listed in square brackets as the last parameter, or *A* is a previously defined array.

Use `Array()` to specify an array directly as an expression. `Array()` is similar to `Table()` (see page 12-10); in addition, it

permits defining an array with repeated values (see Example 3), and changing indexes of a previously defined array (see Example 4).

**Library:** Array

**Example 1:** \*

Definition viewed as an expression:

```
Array(Car_type, [32, 34, 18])
```

Definition viewed as a table:

*Car\_type* ►

	<b>VW</b>	<b>Honda</b>	<b>BMW</b>
	32	34	18

**Example 2:** \*

If an array has multiple dimensions, then the elements are listed in nested brackets, following the structure of the array as an array of arrays (of arrays..., and so on, according to the number of dimensions):

Definition viewed as an expression:

```
Array(Car_type, Years, [[8K, 9K, 9.5K, 10K],  
[12K, 13K, 14K, 14.5K], [18K, 20K, 21K, 22K]])
```

Definition viewed as a table:

*Car\_type* ▼, *Years* ►

	<b>1985</b>	<b>1986</b>	<b>1987</b>	<b>1988</b>
<b>VW</b>	8000	9000	9500	10K
<b>Honda</b>	12K	13K	14K	14.5K
<b>BMW</b>	18K	20K	21K	22K

The size of each array in square brackets must match the size of the corresponding index. In this case, there is an array of three elements (for the three car types), and each element is an array of

\* Example variables are defined on page 12-3.

four elements (for the four years). An error message displays if these sizes don't match. See also `Size()` on page 12-27.

### Example 3: \*

If an element is a scalar where an array is expected, `Array()` expands it to create an array with the scalar value repeated across a dimension.

Definition viewed as an expression:

```
Array(Car_type, Years, [[8K, 9K, 9.5K, 10K], 13K,
[18K, 20K, 21K, 22K]])
```

Definition viewed as a table:

*Car\_type* ▼, *Years* ►

	1985	1986	1987	1988
VW	8000	9000	9500	10K
Honda	13K	13K	13K	13K
BMW	18K	20K	21K	22K

### Example 4: \*

Use `Array()` to change an index of a previously defined array.

*Car\_model*:

Jetta	Accord	320
-------	--------	-----

*Table\_a*: `Table(Car_type) (32, 34, 18)`

*Table\_b*: `Array(Car_model, Table_a) →`

*Car\_model* ►

	Jetta	Accord	320
	32	34	18

\* Example variables are defined on page 12-3.



**Table ( *l1, l2, ... ln* )** ( *u1, u2, u3, ... um* )  
Creates an *n*-dimensional array of *m* elements, indexed by the indexes *l1, l2, ... ln*. In the set of indexes, *l1* is the index of the outermost dimension, varying the least rapidly.

The second set of parameters, *u1, u2 ... um*, specifies the values in the array. The number of values, *m*, must equal the product of the sizes of all of the dimensions.

Each *u* is an expression that evaluates to a number, text value or probability distribution. It can also evaluate to an array, causing the dimensions of the entire table to increase. *u* cannot be a literal list.

Both sets of parameters are enclosed in parentheses; the separating commas are optional except if the table values are negative.

Use `Table()` to specify an array directly as an expression. `Table()` is similar to `Array()` (see page 12-7); `Table()` requires *m* numeric or text values.

A definition created as a table from the Expressions popup menu uses `Table()` in expression view.

**Library:** Array

#### Example 1: \*

Definition viewed as an expression:

```
Table(Car_type) (32, 34, 18)
```

Definition viewed as a table:

*Car\_type* ►

	VW	Honda	BMW
	32	34	18

#### Example 2: \*

Definition viewed as an expression:

```
Table(Car_type, Years)
(8K, 9K, 9.5K, 10K, 12K, 13K, 14K, 14.5K, 18K, 20K,
21K, 22K)
```

\* Example variables are defined on page 12-3.

Definition viewed as a table:

*Car\_type* ▼, *Years* ►

	1985	1986	1987	1988
<b>VW</b>	8000	9000	9500	10K
<b>Honda</b>	12K	13K	14K	14.5K
<b>BMW</b>	18K	20K	21K	22K

**Example 3:** \*

A table created with blank (zero) cells appears in expression view without the second set of parameters.

Definition viewed as a table:

*Car\_type* ▼, *Years* ►

	1985	1986	1987	1988
<b>VW</b>	0	0	0	0
<b>Honda</b>	0	0	0	0
<b>BMW</b>	0	0	0	0

Definition viewed as an expression:

Table(*Car\_type*, *Years*)

## 12.4 Array-reducing functions

An *array-reducing function* operates across a dimension of an array and returns a result that has one dimension less than the number of dimensions of its input array. When applied to an array of  $n$  dimensions, a reducing function produces an array that contains  $n-1$  dimensions.

The function `Sum(X, I)` illustrates some properties of reducing functions.

\* Example variables are defined on page 12-3.

**Examples: \***

Sum(Car\_prices, Car\_type) →  
 Years ►

	1985	1986	1987	1988
	38K	42K	44.5K	46.5K

Sum(Car\_prices, Years) →  
 Car\_type ►

	VW	Honda	BMW
	36.5K	53.5K	81K

Sum(Sum(Car\_prices, Years), Car\_type) → 171K

The second parameter, *l*, specifying the dimension over which to sum, is optional. But, if the array, *X*, has more than one dimension, Analytica may not sum over the dimension you expect. For this reason, it is safer *always* to specify the dimension index explicitly in Sum() or any other array-reducing function.

If the index variable, *l*, is not a dimension of the array, *X*, Sum(*X*, *l*) returns *X* unreduced. In this way, Analytica will evaluate the model properly even if the number of dimensions changes.

**Area ( *R*, *l*, *X1*, *X2* )** Returns the area (sum of trapezoids) under array *R* across index *l* between *X1* and *X2*. *l* must contain increasing numbers. *X1* and *X2* are optional; if not specified, the area is calculated across all of *l*.

If *X1* or *X2* fall outside the range of values in *l*, the first value (for *X1*) or last value (for *X2*) are used. Area() computes the total integral across *l*, returning a value with one less dimension than *R*. Compare Area() to Integrate() (see page 12-18).

**Library:** Array

\* Example variables are defined on page 12-3.

**Example: \***

Area(Cost\_in\_time, Time, 0, 5000) →  
 Car\_type ▼, Mpg ►

	26	30	35
VW	9652	12.42K	15.18K
Honda	10.11K	12.84K	15.86K
BMW	13.65K	16.42K	19.18K

**Argmax ( R, I )** Returns the corresponding value in index *I* for which *R* is the maximum. If more than one value equals the maximum, returns the index of the last occurrence.

**Library:** Special

**Example: \***

Argmax(Car\_prices, Car\_type) →  
 Years ►

	1985	1986	1987	1988
BMW	BMW	BMW	BMW	BMW

To obtain the corresponding value in index *I* for which *A* is the minimum use Argmax(-A, I).

Argmax(-Car\_prices, Car\_type) →

	1985	1986	1987	1988
VW	VW	VW	VW	VW

**Average ( X, I )** Returns the mean value of all of the elements of array *X*, averaged over index *I*.

**Library:** Array

**Examples: \***

Average(Mpg) → 30.33

Average(Car\_prices, Car\_type) →  
 Years ►

	1985	1986	1987	1988
	12.67K	14K	14.83K	15.5K

\* Example variables are defined on page 12-3.

**Max ( X, I)** Returns the highest valued element of *X* along index *I*.

**Library:** Array

**Examples:** \*

Max(Years) → 1988

Max(Car\_prices, Years) →  
Car\_type ►

	VW	Honda	BMW
	10K	14.5K	22K

To obtain the maximum of two numbers, first turn them into an array:

Max([10, 5]) → 10

**Min ( X, I)** Returns the lowest valued element of *X* along index *I*.

**Library:** Array

**Examples:** \*

Min(Years) → 1985

Min(Car\_prices, Years) →  
Car\_type ►

	VW	Honda	BMW
	8000	12K	18K

To obtain the minimum of two numbers, first turn them into an array:

Min([10, 5] ) → 5

**Product ( X, I)** Returns the product of all of the elements of *X*, along the dimension indexed by *I*.

**Library:** Array

\* Example variables are defined on page 12-3.

**Examples: \***

Product(Mpg) → 27.3K

Product(Cost, Mpg) →  
Car\_type ►

	VW	Honda	BMW
	5.905G	14.47G	28.78G

**Subindex ( A, U, I )** Returns the value of *I* corresponding to value *U* in array *A*. If more than one value corresponds, returns the index value of the last occurrence. For the values that do not correspond, returns undefined (shows as blank).

Argmax() uses Subindex(A, Max(A, I), I) to return the index value corresponding to the maximum value in *A*. See Argmax() on page 12-13.

**Library:** Special

**Examples: \***

Subindex(Car\_prices, 12K, Car\_type) →  
Years ►

	1985	1986	1987	1988
Honda				

Subindex(Car\_prices, 12K, Years) →  
Car\_type ►

	VW	Honda	BMW
		1985	

If *U* is an array of values, an array of index values is returned.

Subindex(Car\_prices, [12K, 21K], Car\_type) →  
Subindex ▼, Years ►

	1985	1986	1987	1988
12K	Honda			
21K			BMW	

\* Example variables are defined on page 12-3.

**Sum ( X, I )** Returns the sum of array *X* over the dimension indexed by variable *I*.

**Library:** Array

**Examples:** \*

Sum(Mpg) → 91

Sum(Car\_prices, Years) →  
Car\_type ►

	VW	Honda	BMW
	36.5K	53.5K	81K

## 12.5 Transforming functions

A *transforming function* operates across a dimension of an array and returns a result that has the same dimensions as its input array.

The function `Cumulate(X, I)` illustrates some properties of transforming functions.

**Example:** \*

Cumulate(Car\_prices, Years) →

Car\_type ▼, Years ►

	1985	1986	1987	1988
VW	8000	17K	26.5K	36.5K
Honda	12K	25K	39K	53.5K
BMW	18K	38K	59K	81K

The second parameter, *I*, specifying the dimension over which to cumulate, is optional. But, if the array, *X*, has more than one dimension, Analytica may not cumulate over the dimension you expect. For this reason, it is safer *always* to specify the dimension index explicitly in any transforming function.

\* Example variables are defined on page 12-3.

If the index variable,  $l$ , is not a dimension of the array,  $X$ , `Cumulate(X, l)` returns  $x$  unreduced. In this way, Analytica will evaluate the model properly even after the number of dimensions change.

**Cumproduct (  $X, l$  )** Returns an array with each element being the product of all of the elements of  $X$  along dimension  $l$  up to, and including, the corresponding element of  $X$ .

**Library:** Array

**Example: \***

`Cumproduct(Cost_in_time, Time) →`  
*Mpg ▼, Time ►, Car\_type = VW ↑*

	0	1	2	3	4
26	2185	5.012M	12.08G	30.54T	81.11Q
30	2810	8.292M	25.69G	83.57T	285.5Q
35	3435	12.39M	46.92G	186.6T	778.9Q

**Cumulate (  $X, l$  )** Returns an array with each element being the sum of all of the elements of  $X$  along dimension  $l$  up to, and including, the corresponding element of  $X$ .

**Library:** Array

**Example: \***

`Cumulate(Cost_in_time, Time) →`  
*Mpg ▼, Time ►, Car\_type = VW ↑*

	0	1	2	3	4
26	2185	4479	6888	9417	12.07K
30	2810	5761	8859	12.11K	15.53K
35	3435	7042	10.83K	14.8K	18.98K

\* Example variables are defined on page 12-3.



**Integrate ( *R*, *I* )** Returns the result of applying the trapezoidal rule of integration of array *R* over index *I*. `Integrate()` computes the cumulative integral across *I*, returning a value with the same number of dimensions as *R*. Compare `Integrate()` to `Area()` (see page 12-12).

An alternative syntax is `Integrate( R1, R2, I )`, which returns the integral of array *R1* over array *R2*. If *R2* has one dimension, its index must also be an index of *R1* and *I* is optional. If *R2* has more than one dimension, then *I* is required and must be an index of both *R1* and *R2*.

**Library:** Array

**Example:** \*

`Integrate(Cost_in_time, Time) →`  
*Mpg* ▼, *Time* ►, *Car\_type* = VW ↓

	0	1	2	3	4
26	0	2240	4591	7060	9652
30	0	2880	5905	9080	12.42K
35	0	3521	7218	11.1K	15.18K

**Normalize ( *R*, *I* )** Returns an array that is normalized array *R*, so the area across index *I* is 1.

`Normalize()` does not force the values along index *I* to sum to 1; to make the values sum to 1, divide *R* by `Sum( R, I )`.

An alternative syntax is `Normalize( R1, R2, I )`, which returns the normalized array of array *R1* over array *R2*. If *R2* has one dimension, its index must also be an index of *R1* and *I* need not be stated. If *R2* has more than one dimension, then *I* is required and must be an index of *R1* and *R2*.

**Library:** Array

\* Example variables are defined on page 12-3.

**Example: \***

Normalize(Cost\_in\_Time, Time) →  
 Mpg ▼, Time ►, Car\_type = VW ↓

	0	1	2	3	4
26	0.2264	0.2377	0.2496	0.2620	0.2752
30	0.2263	0.2377	0.2495	0.2620	0.2752
35	0.2264	0.2377	0.2496	0.2620	0.2751

**Rank ( X, I )** Returns an array of the rank values of *X* across index *I*. The lowest value in *X* has a rank value of 1, the next-lowest has a rank value of 2, and so on. *I* is optional if *X* is one-dimensional. If *I* is omitted when *X* is more than one-dimensional, the innermost dimension is ranked.

If two values are equal, they receive the same rank and the next-higher value receives a rank 2 higher.

**Library:** Array

**Examples: \***

Rank(Mpg) →  
 Mpg ►

	26	30	35
	1	2	3

Rank(Car\_prices, Car\_type) →  
 Car\_type ▼, Years ►

	1985	1986	1987	1988
VW	1	1	1	1
Honda	2	2	2	2
BMW	3	3	3	3

**Uncumulate ( X, I )** Returns an array whose first element (along *I*) is 0, and each other element is the difference between the corresponding element of *X* and the previous element of *X*.

Uncumulate() is similar to a discrete differential operator.

**Library:** Array

\* Example variables are defined on page 12-3.

**Example: \***

Uncumulate(Cost\_in\_time, Time) →  
 Mpg ▼, Time ►, Car\_type = VW ↓

	0	1	2	3	4
26	0	109	115	120	127
30	0	141	147	155	163
35	0	172	180	189	199

## 12.6 Functions that select part of an array

Analytica includes four functions useful for selecting an element or slice of an array.

With `Choice()`, you can select an element from a list.

With `Slice()`, you can select the *n*th element or “plane” of an array. With `Ident[I=U]` and `Subscript()`, you can select the element or “plane” of an array whose index matches a given value.

All these functions return a result that has one dimension less than the number of dimensions of its input array.

**Choice ( *l*, *n* )** Appears as a popup menu in the definition field, allowing selection of the *n*th item from *l* (see “Creating a popup menu” on page 9-3). `Choice()` must appear at the topmost level of a definition. It cannot be used inside another expression.

**Library:** Array

**Examples: \***

`Choice(Years, 2)` → 1986

If *n*=0, all values of *l* are returned:

`Choice(Years, 0)` →

*Years* ►

	1985	1986	1987	1988
--	------	------	------	------

\* Example variables are defined on page 12-3.

***Ident* [I = U]** Returns a specific element or slice of an array, where *Ident* is the identifier of an array variable, *I* is an index variable, and *U* is one or more elements of index *I* that corresponds to the desired array element. `Ident[I = U]` is equivalent to `Subscript(U1, I, U2)` when *Ident* is a variable identifier that evaluates to *U*.

**Library:** Special

**Examples:** \*

`Car_prices[Car_type = 'VW']` →  
**Years** ►

	1985	1986	1987	1988
	8000	9000	9500	10K

`Car_prices[Car_type = ['VW', 'Honda']]` →  
**Years** ►

	1985	1986	1987	1988
VW	8000	9000	9500	10K
Honda	12K	13K	14K	14.5K

You can specify more than one index when each index is given a single value.

`Car_prices[Car_type = 'Honda', Years = 1986]` → 13K

***Slice* (U, I, N)** Returns the element or cross-section of array *U*, for which index *I* has position *N*. *I* must be an index of *U*, and *N* must be an integer or array of integers between 1 and the length of *I*.

If *N* is an integer, the result of `slice()` is an array indexed by all indexes of *U* except *I*. If *N* is an array, the result of `slice()` is also indexed by the indexes of *N*.

If *U* is a scalar, `slice(U, I, N)` returns *U*.

**Library:** Array

\* Example variables are defined on page 12-3.

**Examples: \***

Here, *Analytica* returns the values in *Cost* corresponding to the first element in *Car\_type*, that is, the values of *VW*:

`Slice(Cost, Car_type, 1) →`  
*Mpg* ►

	26	30	35
	2185	1705	1585

Here, *N* is an array of positions:

`Slice(Cost, Car_type, [1, 2]) →`  
*Mpg* ►

	26	30	35
1	2185	1705	1585
2	2810	2330	2210

**Subscript (*U1*, *I*, *U2*)** Returns the element or slice of array *U1*, for which index *I* has value *U2*. *I* must be an index of *U1*, and *U2* must be value(s) of *I*.

If *U2* is a single value, the result of `Subscript()` is an array indexed by all indexes of *U1* except *I*. If *U2* is an array, the result of `Subscript()` is also indexed by the indexes of *U2*.

If *U1* is a single value, `Subscript(U1, I, U2)` returns *U1*.

`Subscript(U1, I, U2)` is equivalent to `Ident[I = U]` when *Ident* is a variable identifier that evaluates to *U1*. `Subscript()` allows *U1* to be an arbitrary expression.

**Library:** Array

**Examples: \***

To see the values in *Cost* corresponding to `Mpg = 26`:

`Subscript(Cost, Mpg, 26) →`  
*Car\_type* ►

	VW	Honda	BMW
	2185	2810	3435

\* Example variables are defined on page 12-3.

Here  $U2$  is an array of values:

`Subscript(Cost, Car_type, ['VW', 'Honda'])` →  
 $Car\_type \blacktriangledown, Mpg \blacktriangleright$

	26	30	35
VW	2185	1705	1585
Honda	2810	2330	2210

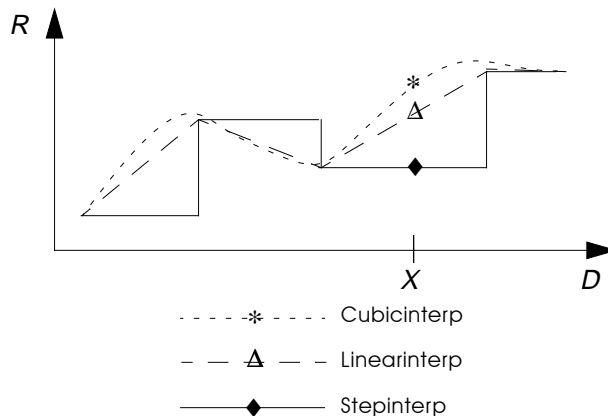
Example of an arbitrary expression as the first parameter:

`Subscript(Cost/12, Mpg, 26)` →  
 $Car\_type \blacktriangleright$

	VW	Honda	BMW
	182.1	234.2	286.2

## 12.7 Interpolation functions

Analytica includes three functions that interpolate across arrays. The graph below is a simple comparison of the three.



The first two examples use the following variables:

$Index\_a$ :

a	b	c
---	---	---

$Index\_b$ :

1	2	3
---	---	---

Array\_a:  
Index\_a ▼, Index\_b ►

	1	2	3
a	7	-3	1
b	-4	-1	6
c	5	0	-2

**Cubicinterp (  $D, R, X$  )** Returns the natural cubic spline interpolated values of  $R$  along  $D$ , interpolating for values of  $X$ .  $D$  must be a one dimensional array of increasing numbers, whose index is an index of  $R$ .

For each value of  $X$ , `Cubicinterp()` finds the nearest values from  $D$ , and using a natural cubic spline between the corresponding values of  $R$ , computes the interpolated value. If  $X$  is less than the minimum value in  $D$ , it returns the first value in  $R$ ; if  $X$  is greater than the maximum value in  $D$ , it returns the last value for  $R$ .

**Library:** Special

**Example:**

`Cubicinterp(Index_b, Array_a, 1.5) →`  
Index\_a ►

	a	b	c
	0.6875	-2.875	2.219

**Linearinterp (  $D, R, X$  )** Returns linearly interpolated values of  $X$ , given  $R$  representing an arbitrary piecewise linear function.  $D$  must be a one-dimensional array of increasing numbers, whose index is an index of  $R$ .  $R$  is an array of the corresponding output values for the function (not necessarily increasing and may be more than one dimension).  $X$  may be probabilistic and/or an array.

For each value of  $X$ , `Linearinterp()` finds the nearest two values from  $D$  and interpolates linearly between the corresponding values from  $R$ . If  $X$  is less than the minimum value in  $D$ , it returns the first value in  $R$ . If  $X$  is greater than the maximum value in  $D$ , it returns the last value in  $R$ .

**Library:** Special

**Example:**

Linearinterp(Index\_b, Array\_a, 1.5) →  
 Index\_a ►

	a	b	c
	2	-2.5	2.5

**Stepinterp (D,A,X)** Returns the element or slice of array *A* for which *D* has the smallest value that is greater than or equal to *X*. *D* must be a one-dimensional array of increasing numbers whose index is an index of *A*. If *X* is greater than all values of *D*, returns the element for which *D* has the largest value.

If *X* is a single value, the result of Stepinterp() is an array indexed by all indexes of *A* except *D*'s index. If *X* is an array, the result of Stepinterp() is also indexed by the indexes of *X*.

Stepinterp() is similar to Subscript() (see page 12-22); however, Subscript() selects based on the index value being equal to *X*, while Stepinterp() selects based on the array value being greater than or equal to *X*.

Stepinterp() can be used to perform table lookup.

**Library:** Special

**Examples: \***

To see the values in *Cost* corresponding to *Mpg* >= 33:

Stepinterp(MPG, Cost, 33) →  
 Car\_type ►

	VW	Honda	BMW
	1585	2210	2835

Here *X* is an array of values:

Stepinterp(MPG, Cost, [28,33]) →

	VW	Honda	BMW
<b>28</b>	1705	2330	2955
<b>33</b>	1585	2210	2935



## 12.8 Other array functions

**Concat ( A1, A2, I, J, K )** Appends array A2 to array A1. I and J are indexes of A1 and A2, respectively. K is the index of the resulting dimension, and usually consists of the list created by concatenating I and J.

A1 and A2 must have the same number of dimensions. If they are one-dimensional, the parameters I, J, and K are optional. If not specified, the resulting array is unindexed.

If A1 and A2 are multidimensional, they must have the same nonconcatenated indexes.

**Library:** Array

### Examples:

In addition to the variables on page 12-3, these examples use the following:

*More\_years:*

1989	1990	1991
------	------	------

*All\_years:*

1985	1986	1987	1988	1989	1990	1991
------	------	------	------	------	------	------

*More\_prices:* Car\_type ▼, More\_years ►

	1989	1990	1991
VW	11K	12K	12.5K
Honda	15K	15.5K	16.5K
BMW	23.5K	25K	27K

Concat(Years, More\_years) →

*Concat* ►

	1985	1986	1987	1988	1989	1990	1991
--	------	------	------	------	------	------	------

*Sequence2:* Sequence(1,7)

Concat(Years, More\_years, Years, More\_years, Sequence2) →

*Sequence2* ►

	1	2	3	4	5	6	7
	1985	1986	1987	1988	1989	1990	1991

Concat(Car\_prices, More\_prices, Years, More\_years,  
All\_years) →  
All\_years ▼, Car\_type ►

	VW	Honda	BMW
1985	8000	12K	18K
1986	9000	13K	20K
1987	9500	14K	21K
1988	10K	14.5K	22K
1989	11K	15K	23.5K
1990	12K	15.5K	25K
1991	12.5K	16.5K	27K

**Size (U)** Returns the number of array elements of *U*.

**Library:** Array

**Examples:** \*

Size(Years) → 4

Size(Car\_prices) → 12

Size(10) → 1

**Sortindex (D)** Returns a list containing the elements of *D*'s index, sorted according to *D*'s values (from smallest to largest). *D* must be a one-dimensional array, with either all numeric values or all text values.

**Library:** Array

**Examples:** \*

Maint\_costs:

Car\_type ►

	VW	Honda	BMW
	1950	1800	2210

SortIndex (Maint\_costs) →

SortIndex ►

	Honda	VW	BMW
--	-------	----	-----

\* Example variables are defined on page 12-3.

To sort a list, use both `Sortindex()` and `Subscript()` (see page 12-22).

Define `Index_new` as an index node:

```
Index_new: Sortindex(Maint_costs)
```

```
Subscript(Maint_costs, Car_type, Index_new) →
```

	Honda	VW	BMW
	1800	1950	2210

**Subset (*D*)** Returns a list containing all the elements of *D*'s index for which *D*'s values are true (that is, non-zero). *D* must be a one-dimensional array. If all elements of *D* are false (zero), then `Subset(D)` is undefined.

**When to use:** Use `Subset()` to create a new index that is a subset of an existing index.

**Library:** Array

**Example:** \*

```
Subset(Years < 1987) → [1985, 1986]
```

## 12.9 Matrix functions

Matrix functions perform matrix operations. In Analytica, a *matrix* is defined as a two-dimensional array of numbers with indexes of equal length.

**Decompose (*C,I,J*)** Returns the Cholesky decomposition (square root) matrix of matrix *C* along dimensions *I* and *J*. Matrix *C* must be symmetric and positive-definite. (Positive-definite means that  $v * C * v > 0$ , for all vectors *v*.)

Cholesky decomposition computes a lower diagonal matrix *L* such that  $L * L' = C$ , where *L'* is the transpose of *L*.

**Library:** Special

\* Example variables are defined on page 12-3.

**Example:***MatrixS:**l* ▼, *m* ►

	1	2	3	4	5
1	6	2	6	3	1
2	2	4	3	1	3
3	6	3	9	3	4
4	3	1	3	8	4
5	1	3	4	4	7

Decompose(MatrixS, l, m) →

*l* ▼, *m* ►

	1	2	3	4	5
1	2.4495	0	0	0	0
2	0.8165	1.8257	0	0	0
3	2.4495	0.5477	1.6432	0	0
4	1.2247	0	0	2.5495	0
5	0.4082	1.4606	1.3389	1.3728	1.0113

**Determinant ( C, l, J)** Returns the determinant of matrix *C* along dimensions *l* and *J*.**Library:** Special**Example:***MatrixA:**j* ▼, *i* ►

	1	2	3
a	4	1	2
b	2	5	3
c	3	2	7

Determinant(MatrixA, i, j) → 89

**Invert ( *C, I, J* )** Returns the inversion of matrix *C* along dimensions *I* and *J*.

**Library:** Special

**Example** (set number format to fixed point, 3 decimal digits):

Invert(MatrixA, i, j) →  
*j* ▼, *i* ►

	1	2	3
a	0.326	-0.034	-0.079
b	-0.056	0.247	-0.090
c	-0.124	-0.056	0.202

**Transpose ( *C, I, J* )** Returns the transpose of matrix *C* along dimensions *I* and *J*.

**Library:** Special

**Example:**

Transpose(MatrixA, i, j) →  
*j* ▼, *i* ►

	1	2	3
a	4	2	3
b	1	5	2
c	2	3	7

## Dot product of two matrices

The dot product of *MatrixA* and *MatrixB* is equal to

Sum(MatrixA \* MatrixB, i)

**Example:**

*MatrixA* is defined as above.

*MatrixB*:

*k* ▼, *i* ►

	1	2	3
l	3	2	1
m	2	5	3
n	4	1	2

Sum(MatrixA \* MatrixB, i) →

$k \blacktriangledown, j \blacktriangleright$

	a	b	c
l	16	19	20
m	19	38	37
n	21	19	28

## 12.10 Control functions

The two functions in this section can be used to control specialized evaluation of arrays.

**For Temp:= I Do Expr** For each successive value of index *I*, assigns that value to variable *Temp*, and evaluates expression *Expr*. *Expr* may refer to *I* and/or *Temp*. *Temp* is a local or temporary variable that can be referred to only within the expression *Expr*.

The result of the `FOR` control function is an array indexed by *I* containing the results of evaluating *Expr*. *I* must be an index variable, or be defined as a list or `Sequence()`.

If you make appropriate use of the intelligent array features described earlier in this and preceding chapters, you will rarely need to use `FOR` structure (unlike in conventional computer languages, which require extensive use of `For` loops and related control structures for handling arrays). Use `FOR` in two specialized cases:

- To apply an Analytica function that requires a one or two dimensional array input to a higher dimensioned array: the interpolation functions, `Linearinterp()`, `Stepinterp()`, `Cubicinterp()`, whose first parameter must be a one-dimensional array, and the matrix functions, `Decompose()`, `Invert()`, and `Transpose()`, whose parameters must be two-dimensional matrices.
- To reduce the memory needed for calculations with very large arrays.

**Library:** Special

**Examples:**

*X*: A two-dimensional array indexed by *J* and *K*

*Y*: A function over *X*, a two-dimensional array indexed by *J* and *K*

*X\_in*: A list of values in *X* for which you want to perform linear interpolation to find the corresponding values from *Y*.

You cannot use

```
Linearinterp(X, Y, X_in)
```

because the first parameter to `Linearinterp()` must be one-dimensional (see page 12-24).

Instead, use

```
For L:=K Do Linear_interp(X[K=L],Y[K=L],X_in)
```

This will evaluate correctly, provided that *X* has only two dimensions, one of which is *J*, and the other is an index in common with *Y*.

*Array\_a*: A two dimensional array indexed by *M* and *N*

*Array\_b*: A two dimensional array indexed by *N* and *P*

The generalized matrix inner product is equal to (see page 12-30)

```
Sum(MatrixA * MatrixB, N)
```

During the calculation, Analytica needs memory to compute *MatrixA* \* *MatrixB*, an array indexed by *M*, *N*, and *P*. If these indexes have sizes 100, 200, and 300 respectively, then *MatrixA* \* *MatrixB* contains 6,000,000 numbers, requiring over 60 megabytes of memory at 10 bytes per number.

To reduce the memory required, define the inner product as

```
For L:=M Do Sum(Array_a[M=L]*Array_b,N)
```


Each element `Array[M=L]*Array_b` has dimensions *N* and *P*, needing only  $200 \times 300 \times 10 = 600$  kilobytes of memory at a time.

**Using  $Temp := X$  Do  $Expr$**  Assigns the value of  $X$  to a temporary variable,  $Temp$ , and then evaluates  $Expr$ , which is an expression referring to  $Temp$ .

$Temp$  is a local or temporary variable that can be referred to only within the expression  $Expr$ . Use `Using ... do` whenever a complex subexpression appears more than once in an expression. The `Using` expression is more compact to write, clearer to read, and more efficient to evaluate, since  $Expr$  is evaluated only once.

If you have two or more common subexpressions you can use two or more nested `Using ... do` expressions.

---

**Note:**  The temporary variables from a `Using ... do` control expression are accessible only within the variable that defines them. If your model has the same subexpression in the definitions of multiple variables, consider defining a new permanent variable with this expression, and using that variable instead of the subexpression in each definition.

---

An alternate syntax is

```
Using Temp:= X in I do Expr
```

where  $I$  is an index of  $X$ .

**Library:** Special

**Examples:**

Instead of defining a variable as:

```
Sum(Array_a*Array_b,N)/(1+ Sum(Array_a*Array_b,N))
```

define it as:

```
Using t:=Sum(Array_a*Array*b,N) Do t/(1+t)
```

To compute a correlation between  $Xdata$  and  $Ydata$ , instead of:

```
Sum((Xdata-Sum(Xdata,Data_index)/Nopts)*(Ydata-
Sum(Ydata,Data_index)/Nopts),Data_index)/
Sqrt(Sum((Xdata-Sum(Xdata,Data_index)/
Nopts)^2,Data_index) * Sum((Ydata-
Sum(Ydata,Data_index)/Nopts)^2,Data_index))
```

define the correlation as:



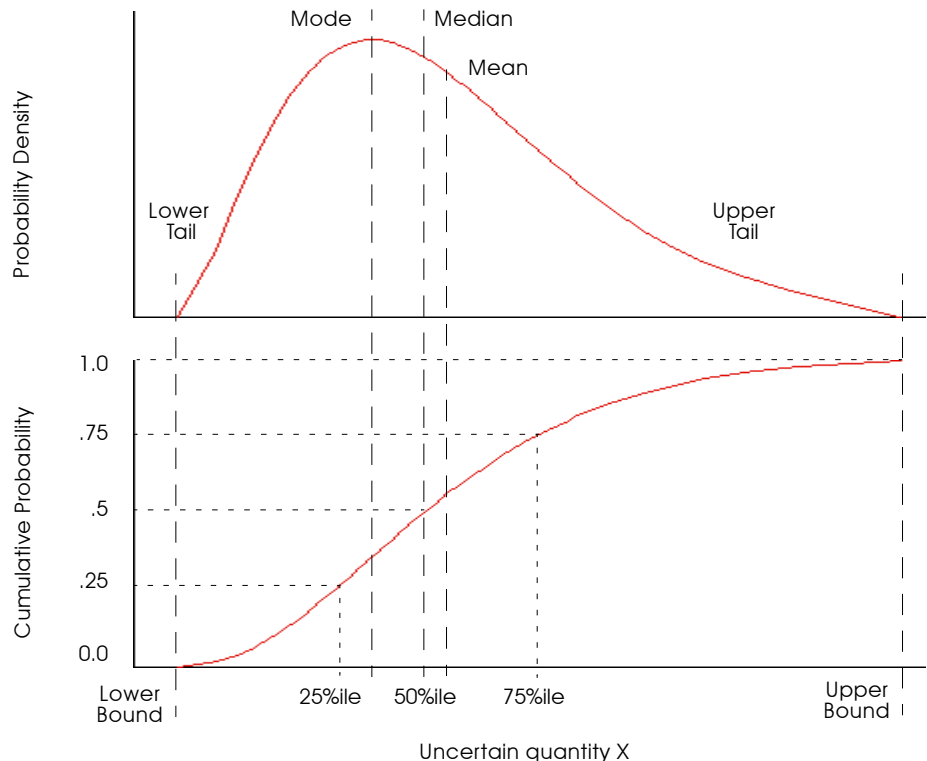
```
Using mx:=Sum(Xdata,Data_index)/Nopts Do
  Using my:=Sum(Ydata,Data_index)/Nopts Do
    Using dx:=Xdata-mx Do
      Using dy:=Ydata-my
        Do Sum(dx*dy,Data_index)
      /Sqrt(dx^2,Data_index)*Sum(dy^2,Data_index)
```

The latter expression is faster to execute and clearer to read.

# 13

## Expressing Uncertainty

Analytica makes it easy to model and analyze uncertainties even if you have minimal background in probability and statistics. The graphs below review several key concepts from probability and statistics that will help you understand the probabilistic modeling facilities in Analytica. This chapter assumes that you have encountered most of these concepts before, but possibly in the distant past. If you need more information, see the Glossary or refer to an introductory text on probability and statistics.



## 13.1 Choosing an appropriate distribution

With Analytica you can express uncertainty about any variable using a probability distribution. You may base the distribution on available relevant data, on the judgment of a knowledgeable individual, or on some combination of data and judgment.

Answer the following questions about the uncertain quantity to select the most appropriate kind of distribution:

- Is it discrete or continuous?
- If continuous, is it bounded?
- Does it have one mode or more than one?
- Is it symmetric or skewed?
- Should you use a standard or a custom distribution?

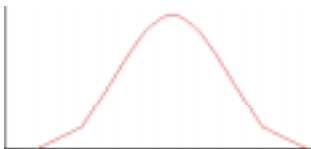
We will discuss how to answer each of these in turn.

### Is the quantity discrete or continuous?

When trying to express uncertainty about a quantity, the first technical question is whether the quantity is discrete or continuous.



A *discrete* quantity has a finite number of possible values, for example, the gender of a person or the country of a person's birth. *Logical* or *Boolean* variables are a type of discrete variable with only two values, true or false, sometimes coded as yes or no, present or absent, or 1 or 0; for example, whether a person was born before January 1, 1950, or whether a person has ever resided in California.

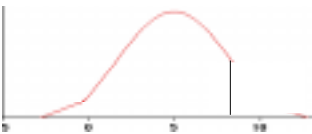


A *continuous* quantity can be represented by a real number, and has infinitely many possible values between any two values in its domain. Examples are the quantity of an air pollutant released during a given period of time, the distance in miles of a residence from a source of air pollution, and the volume of air breathed by a specified individual during one year.

For a large discrete quantity, such as the number of humans residing within 50 miles of Disneyland on December 25, 1980, it is often convenient to treat it as continuous. Even though you know that the number of live people must be an integer, you may want to represent uncertainty about the number with a continuous probability distribution.

Conversely, it is often convenient to treat continuous quantities as discrete, by partitioning the set of possible values into a small finite set of partitions. For example, instead of modeling human age by a continuous quantity between 0 and 120, it is often convenient to partition people into infants (age < 2 years), children (3 to 12), teenagers (13 to 19), young adults (20 to 40), middle-aged (41 to 65), and seniors (over 65 years). This process is termed *discretizing*. It is often convenient to discretize continuous quantities before assessing probability distributions.

### Does the quantity have bounds?



If the quantity is continuous, it is useful to know if it is bounded before choosing a distribution — that is, does it have a minimum and/or maximum value?

Some continuous quantities have exact lower bounds. For example, a river flow cannot be less than zero (assuming the river cannot reverse direction). Some quantities also have exact upper bounds. For example, the percentage of a population that is exposed to an air pollutant cannot be greater than 100%.

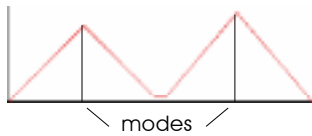
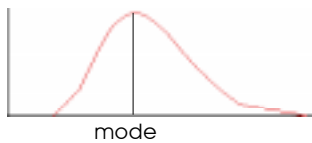
Most real world quantities have *de facto* bounds — that is, you can comfortably assert that there is zero probability that the quantity would be smaller than some lower bound, or larger than some upper bound, even though there is no precise way to determine the bound. For example, you can be sure that no human could weigh more than 5000 pounds; you might be less sure whether 500 pounds is an absolute upper bound.

Many standard continuous probability distributions, such as the normal distribution, are unbounded. In other words, there is some probability that a normally distributed quantity is below any finite value, no matter how small, and above any finite value, no matter how large.

Nevertheless, the probability density drops off quite rapidly for extreme values, with near exponential decay, in fact, for the normal distribution. Accordingly, people often use such unbounded distributions to represent real world quantities that actually have finite bounds. For example, the normal distribution generally provides a good fit for the distribution of heights in a human population, even though you may be certain that no person's height is less than zero or greater than 12 feet.

### How many modes does it have?

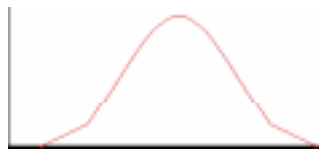
The mode of a distribution is its most probable value. The mode of an uncertain quantity is the value at the highest peak of the density function, or, equivalently, at the steepest slope on the cumulative probability distribution.



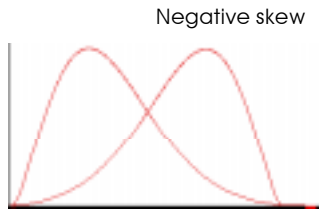
Important questions to ask about a distribution are how many modes it has, and approximately where it, or they, are? Most distributions have a single mode, but some have several and are known as multimodal distributions.

If a quantity has two or more modes, you can usually view it as a combination of two or more populations. For example, the distribution of ages in a daycare center at leaving time might include one mode at age 3 for the children and another mode at age 27 for the parents and caretakers. There is obviously a population of children and a population of parents. It is generally easier to decompose a multimodal quantity into its separate components and assess them separately than to assess a multimodal distribution. You can then assess a unimodal (single mode) probability distribution for each component, and combine them to get the aggregate distribution. This approach is often more convenient, because it lets you assess single-mode distributions, which are easier to understand and evaluate than multimodal distributions.

## Is the quantity symmetric or skewed?



Symmetric



Negative skew



Positive skew

A symmetrical distribution is symmetrical about its mean. A skewed distribution is asymmetric. A positively skewed distribution has a thicker upper tail than lower tail; and vice versa, for a negatively skewed distribution.

Probability distributions in environmental risk analysis are often positively skewed. Quantities such as source terms, transfer factors, and dose-response factors, are typically bounded below by zero. There is more uncertainty about how large they might be than about how small they might be.

## A standard or custom distribution?

The next question is whether to use a standard parametric distribution (e.g., normal, lognormal, or beta) or a custom distribution, where the assessor specifies points on the cumulative probability or density function.

Considering the physical processes that generate the uncertainty in the quantity may suggest that a particular standard distribution is appropriate. More often, however, there is no obvious standard distribution to apply.

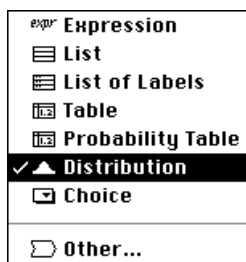
It is generally much faster to assess a standard distribution than a full custom distribution, because standard distributions have fewer parameters, typically from two to four. You should usually start by assigning a simple standard distribution to each uncertain quantity using a quick judgment based on a brief perusal of the literature or telephone conversation with a knowledgeable person. You should assess a custom distribution only for those few uncertain inputs that turn out to be critical to the results. Therefore, it is important to be able to select an appropriate standard distribution quickly for each quantity.

## 13.2 Defining a variable as a distribution

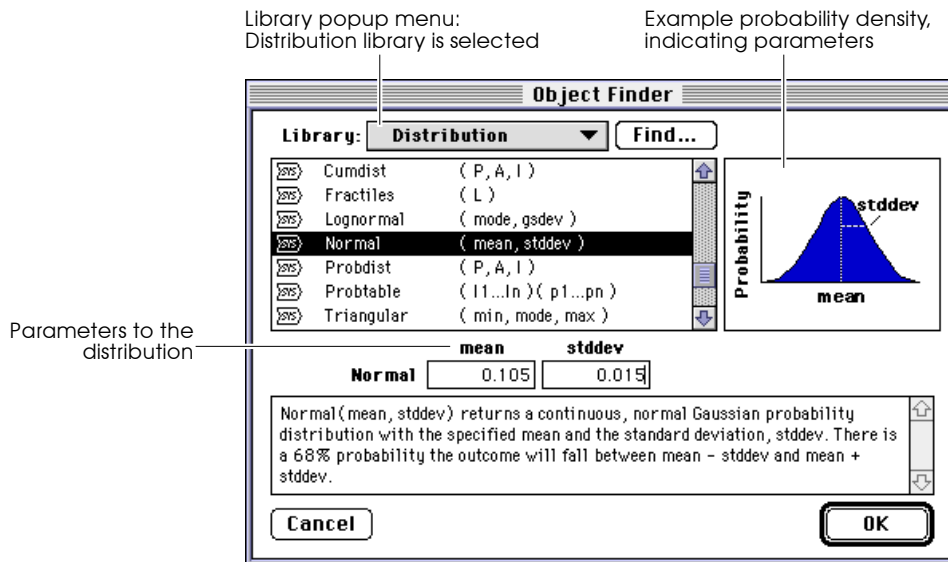
To define a variable as an Analytica probability distribution, first select the variable and open either the variable's Object window or the Attribute panel of the diagram (see page 1-14) with **Definition** selected from the Attribute popup menu (see page 1-15).

To define the distribution:

1. Press the Expression popup menu above the definition field and select **Distribution**.



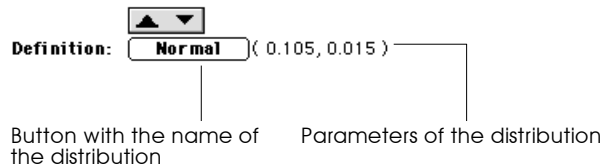
The Object Finder opens, showing the Distribution library.



2. Select the distribution you wish to use.

3. Enter the values for the parameters. You can use an expression or refer to other variables by name in the parameter fields.
4. Click on **OK** to accept the distribution.

If the parameters to the distribution are single numbers, a button appears with the name of the distribution, indicating that the variable is defined as a distribution. To edit the parameters, click on this button.



If the parameters to the distribution are complex expressions, the distribution displays as an expression, e.g.

```
Normal((Price/Mpy) * Mpg, Mpg/10)
```

## Entering a distribution as an expression

Alternatively, you can directly enter a distribution as an expression:

1. Set the cursor in the definition field and type in the distribution name and parameters, e.g.

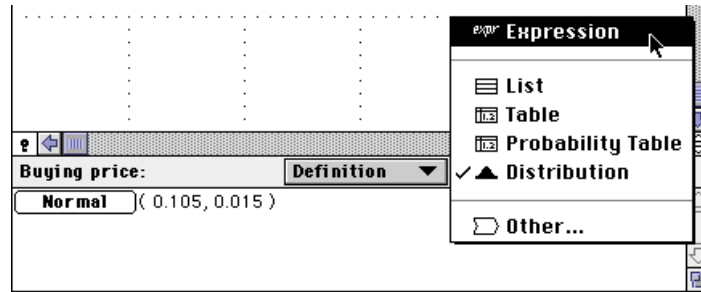
```
Normal(.105, 0.015)
```

2. Press *enter* or click on the  button.

You can also paste a distribution from the Distribution library in the **Definition** menu (see Section 8.5, "Pasting from a library in the Definition menu").



You can edit a distribution as an expression, whether it was entered as a distribution from the Distribution library or as an expression, by selecting **expr** from the Expression popup menu.



### 13.3 Including a distribution in a definition

You can enter a distribution anywhere in a definition, including in a cell of an edit table. Thus, you can have arrays of distributions.

To enter a distribution:

1. Set the insertion point where you wish to enter the distribution in the definition field or edit table cell.
2. Enter the distribution in any of the following ways:
  - Type in the name of the distribution.
  - Paste it from the from the **Distribution** Library under the **Definition** menu.
  - Select **Paste Identifier** from the **Definition** menu to paste it from the **Object Finder**.
3. Type in missing parameters, or replace parameters enclosed as <<x>>.

## 13.4 Overview of built-in probability distributions

Analytica has the following built-in probability functions in the distribution library. The continuous functions are described in Chapter 14 and the discrete functions are described in Chapter 15.

Continuous	Beta()
	Cumdist()
	Fractiles()
	Lognormal()
	Normal()
	Probdist()
	Triangular()
	Uniform()
Discrete	Bernoulli()
	Certain()
	Chancedist()
	Protable

## 13.5 Probabilistic Calculation

Analytica performs probabilistic evaluation of probability distributions through simulation — by computing a random sample of values from the actual probability distribution for each uncertain quantity. The result of evaluating a distribution is represented internally as an array of the sample values, indexed by *Run*. *Run* is an index variable that identifies each sample iteration by an integer from 1 to *Samplesize*.


You can display a probabilistic value using a variety of uncertainty view options — the mean, statistics, probability bands, probability density (or mass function), and cumulative distribution function (see Section 2.4, "Uncertainty view options"). All these views are derived or estimated from the underlying sample array, which you can inspect using the last uncertainty view, *Sample*.

**Example:**

A: Normal (10, 2) →

*Iteration (Run)* ▶

	1	2	3	4	5	6
	10.74	13.2	9.092	11.44	9.519	13.03

**Note:** 

The values in a sample are generated at random from the distribution; if you try this example and display the result as a table, you may see values different from those shown here. To reproduce this example, reset the random number seed to 99 and use the default sampling method and random number method (see Section 13.6).

For each sample run, a random value is generated from each probability distribution in the model. Output variables of uncertain variables are calculated by calculating a value for each value of *Run*.

**Example:**

B: Normal (5, 1) →

*Iteration (Run)* ▶

	1	2	3	4	5	6
	5.09	4.94	4.65	6.60	5.24	6.96


C: A + B →

*Iteration (Run)* ▶

	1	2	3	4	5	6
	15.83	18.13	13.75	18.04	14.76	19.99

Note that each sample value of *C* is equal to the sum of the corresponding values of *A* and *B*.

To control the probabilistic simulation, as well as views of probabilistic results, use the Uncertainty Setup dialog box (see Section 13.6).

**Note:**  If you try to apply an array reducing function (see Section 12.4) to a probability distribution across *Run*, Analytica returns the distribution's Mid value.

**Example:**

`X: Beta(2,3)`

`Mid(X) → 0.3857` and `Max(X,Run) → 0.3857`


To evaluate the input parameters probabilistically and reduce across *Run*, use `Sample()` (see page 16-7).

**Example:**

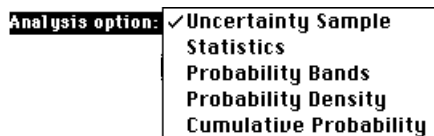
`Max(Sample(X),Run) → 0.8892`

## 13.6 Uncertainty Setup dialog box

Use the Uncertainty Setup dialog box to inspect and change the sample size, sampling method, statistics, probability bands, and samples per plot point for probability distributions. All settings are saved with your model.

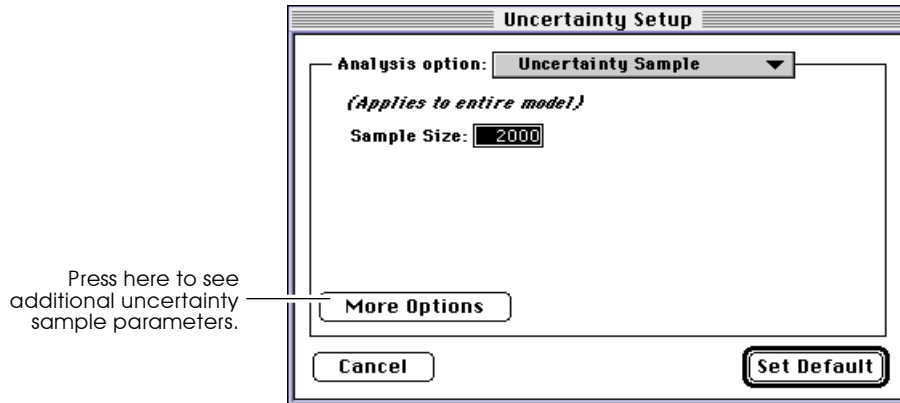
To open the Uncertainty Setup dialog box, select **Uncertainty Options...** from the **Result** menu or -U. To set values for a specific variable, select the variable before opening the dialog box.

The five options for viewing and changing information in the Uncertainty Setup dialog box can be accessed using the Analysis option popup menu.

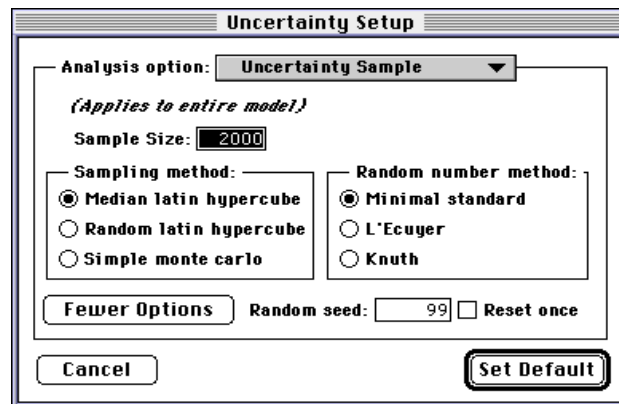


### Uncertainty Sample

To change the sample size or sampling method for the model, select the **Uncertainty Sample** option from the **Analysis options** popup menu.



The default dialog box shows only a field for sample size. To view and change the sampling method, random number method, or random seed, press the **More Options** button.



## Sample size

This number specifies how many runs or iterations Analytica performs to estimate probability distributions. Larger sample sizes take more time and memory to compute, and produce smoother distributions and more precise statistics. See Appendix D, “Selecting the Sample Size,” for guidelines on selecting a sample size. The sample size must be between 2 and 32,000. You can access this number in expressions in your models as the system variable *Samplesize*.

**Sampling method** The sampling method is used to determine how to generate a random sample of the specified sample size,  $m$ , for each uncertain quantity,  $X$ . Analytica provides three options:

### Simple Monte Carlo

The simplest sampling method is known as Monte Carlo, named after the randomness prevalent in games of chance, such as at the famous casino in Monte Carlo. In this method, each of the  $m$  sample points for each uncertain quantity,  $X$ , is generated at random from  $X$  with probability proportional to the probability density (or probability mass for discrete quantities) for  $X$ . Analytica uses the inverse cumulative method; it generates  $m$  uniform random values,  $u_i$  for  $i=1,2,\dots,m$ , between 0 and 1, using the specified random number method (see below). It then uses the inverse of the cumulative probability distribution to generate the corresponding values of  $X$ ,

$$X_i \text{ where } P(x \leq X_i) = u_i \text{ for } i=1,2,\dots,m.$$

With the simple Monte Carlo method, each value of every random variable  $X$  in the model, including those computed from other random quantities, is a sample of  $m$  independent random values from the true probability distribution for  $X$ . You can therefore use standard statistical methods to estimate the accuracy of statistics, such as the estimated mean or fractiles of the distribution, as for example described in Appendix D, “Selecting the Sample Size”.

### Median Latin hypercube (the default method)

With median Latin hypercube sampling, Analytica divides each uncertain quantity  $X$  into  $m$  equiprobable intervals, where  $m$  is the sample size. The sample points are the medians of the  $m$  intervals, that is, the fractiles

$$X_i \text{ where } P(x \leq X_i) = (i-0.5)/m, \text{ for } i=1,2,\dots,m.$$

These points are then randomly shuffled so that they are no longer in ascending order, to avoid nonrandom correlations among different quantities.

### Random Latin hypercube

The random Latin Hypercube method is similar to the median Latin hypercube method, except that instead of using the median of each of the  $m$  equiprobable intervals, Analytica samples at random from each interval. With random Latin hypercube sampling, each sample is a true random sample from the distribution. However, the samples are not totally independent.

### Choosing a sampling method

The advantage of Latin hypercube methods is that they provide more even distributions of samples for each distribution than simple Monte Carlo sampling. Median Latin hypercube is still more evenly distributed than random Latin hypercube. If you display the PDF of a variable that is defined as a single continuous distribution, or is dependent on a single continuous uncertain variable, using median Latin hypercube sampling, the distribution will usually look fairly smooth even with a small sample size (such as 20), whereas the result using simple Monte Carlo will look quite noisy.

If the variable depends on two or more uncertain quantities, the relative noise-reduction of Latin hypercube methods is reduced. If the result depends on many uncertain quantities, the performance of the Latin hypercube methods may not be discernibly better than simple Monte Carlo. Since the median Latin hypercube method is sometimes much better, and almost never worse than the others, Analytica uses it as the default method.

Very rarely, median Latin hypercube can produce incorrect results, specifically when the model has a periodic function, with a period similar to the size of the equiprobable intervals. For example, with

```
X: Uniform(1, Samplesize)
```

```
Y: Sin(2*Pi*X)
```

median Latin hypercube method will give very poor results. In such cases, you should use random Latin hypercube or simple Monte Carlo. If your model has no periodic function of this kind,

you do not need to worry about the reliability of median Latin hypercube sampling.

## **Random number method**

The random number method is used to determine how random numbers are generated for the probability distributions. Analytica provides three different methods for calculating a series of pseudorandom numbers.

### **Minimal Standard** (the default method)

The Minimal Standard random number generator is an implementation of Park and Miller's Minimal Standard (based on a multiplicative congruential method) with a Bays-Durham shuffle. It gives satisfactory results for less than 100,000,000 samples.

### **L'Ecuyer**

The L'Ecuyer random number generator is an implementation of L'Ecuyer's algorithm, based on a multiplicative congruential method, which gives a series of random numbers with a much longer period (sequence of numbers that repeat). Thus, it provides good random numbers even with more than 100,000,000 samples. It is slightly slower than the Minimal Standard generator.

### **Knuth**

Knuth's algorithm is based on a subtractive method rather than a multiplicative congruential method. It is slightly faster than the Minimal Standard generator.

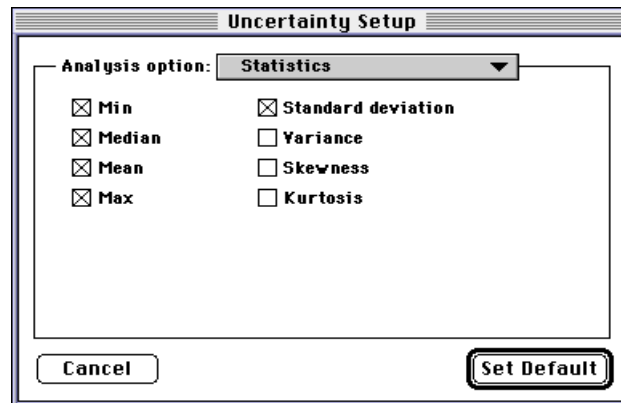


**Random seed** This value must be a number between 0 and 100,000,000 ( $10^8$ ). The series of random numbers starts from this seed value when:

- A model is opened
- The value in this field is changed
- The Reset Once box is checked, and the Uncertainty Setup dialog box is closed by clicking on the Accept or Set Default button.

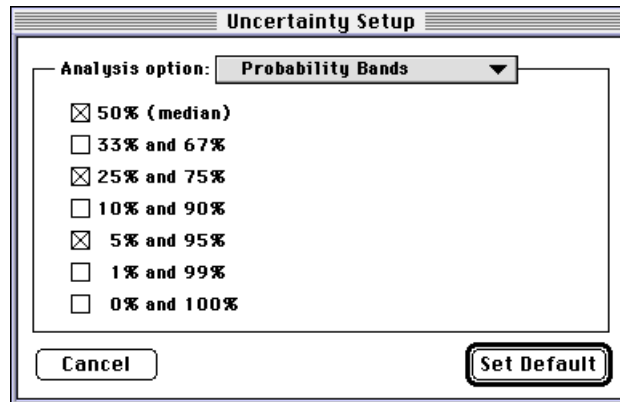
**Reset Once** Check the Reset Once box to produce the exact same series of random numbers.

**Statistics option** To change the statistics reported when you select **Statistics** as the uncertainty view for a result, select the **Statistics** option from the **Analysis option** popup menu.



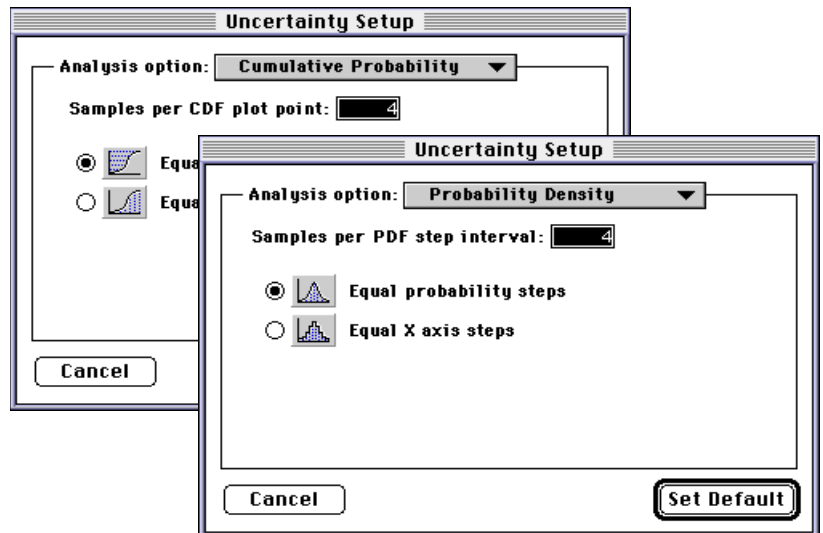
## Probability Bands option

To change the probability bands displayed when you select **Probability Bands** as the uncertainty view for a result, select the **Probability Bands** option from the **Analysis option** popup menu.



## Probability Density and Cumulative Probability options

To change how probability density or the cumulative probability values are drawn or to change their resolution, select the respective option from the **Analysis option** popup menu.



Analytica estimates the probability density function and cumulative distribution function, like other uncertainty views, from the underlying array of sample values for each uncertain quantity. As with any simulation-based method, each estimated distribution will have some noise and variability from one evaluation to the next.

### **Samples per plot point**

This number controls the average number of sample values used to estimate each point on the probability density function (PDF) or cumulative distribution function (CDF) curves.

For a small number of samples per plot point (less than 10), more points are each estimated from fewer sample values and so are more susceptible to random noise. If the quantity is defined by a single probability distribution, and if you use median Latin hypercube method (the default), this noise will be slight and the curve will look smooth. In other cases, the noise may have a large effect, and using a larger number of samples per plot point will produce a smoother curve. There is a trade-off; with larger numbers the smoothing may miss details of the shape of the curve. PDFs may be much more susceptible to random noise than CDFs, so you may wish to use larger numbers for PDFs than CDFs. Ultimately, to reduce the noise, use a larger sample size (for details on selecting the sample size, see Appendix D, “Selecting the Sample Size”).

### **Equal probability steps**

With this option, Analytica estimates from the sample a set of  $m$  fractiles (quantiles),  $X_p$ , at equal probability intervals, where  $p=0, q, 2q, 3q, \dots, 1$ , and  $q = 1/(m+1)$ . In the cumulative probability view, these points are distributed at equal probability intervals along the vertical axis. In the probability density view, the areas under the density function between successive fractiles are equal because they each represent the same probability,  $q$ . The height of the line at each plotted point — the probability density — is estimated as the average density of sample values between the previous and the next fractile. Analytica uses linear interpolation between these points.


**Equal X axis steps** With this option, Analytica estimates cumulative probability for equally spaced points along the  $X$  axis. In the probability density graph view, it shows a histogram where the height of each horizontal is estimated as the fraction of the sample values that fall within that  $X$  interval.



# 14

## Using Continuous Probability Distributions

This chapter describes Analytica's built-in continuous probability distributions. Additional continuous probability distributions are included in the Libraries folder distributed with Analytica.

**Note:**  To reproduce the graphs in this chapter, use a sample size of 1000.

### 14.1 Continuous distribution functions

**Beta (  $X, Y, lower, upper$  )** Creates a distribution of numbers between 0 and 1 with  $\frac{X}{(X+Y)}$  representing the mean, if the optional parameters *lower* and *upper* are omitted. For bounds other than 0 and 1, specify the optional *lower* and *upper* bounds to offset and expand the distribution.

$X$  and  $Y$  must be positive.

#### When to use:

Use a beta distribution if the uncertain quantity is bounded by 0 and 1 (or 100%), is continuous, and has a single mode. This distribution is particularly useful for modeling an opinion about the fraction of a population that has some characteristic. For example, if you have observed  $n$  members of the population, of which  $r$  display the characteristic  $c$ , you can represent the uncertainty about the true fraction with  $c$  using a beta distribution with parameters  $X = r$  and  $Y = n - r$ .

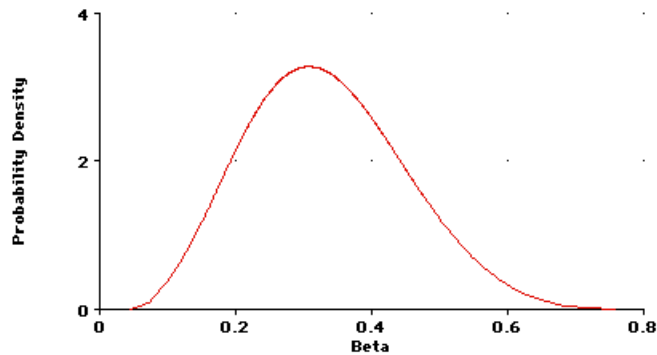
If the uncertain quantity has lower and upper bounds other than 0 and 1, include the lower and upper bounds parameters to obtain a *transformed beta* distribution. The transformed beta is a very

flexible distribution for representing a wide variety of bounded quantities.

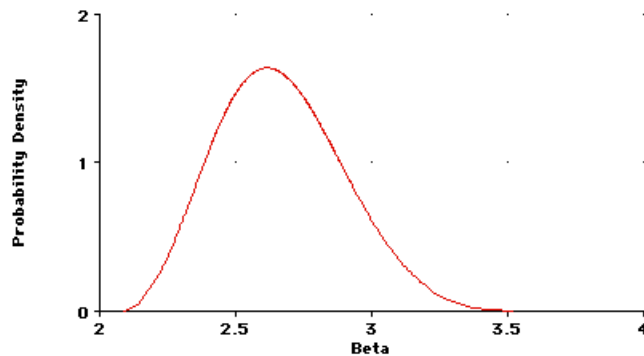
**Library:** Distribution

**Examples:**

Beta (5, 10) →



Beta (5, 10, 2, 4) →



**Cumdist (  $P, R, I$  )** Specifies a continuous probability distribution by an array of cumulative probabilities,  $P$ , for an array of corresponding outcome values,  $R$ , for the quantity. Either  $R$  must be an index of  $P$ , or  $P$  and  $R$  must have an index in common. If  $P$  or  $R$  have more than one index, you must specify the relevant index for linking  $P$  and  $R$  as a third parameter,  $I$ .

`CumDist()` uses linear interpolation of the cumulative distribution between the specified points, which implies a piecewise uniform distribution.

The values of  $P$  must be nondecreasing.  $P$ 's first value must be equal to or greater than 0, and its last value must be 1.0. The values of  $R$  must be increasing.

**Library:** Distribution

**Example:**

*Array\_b:*

*Index\_a* ►

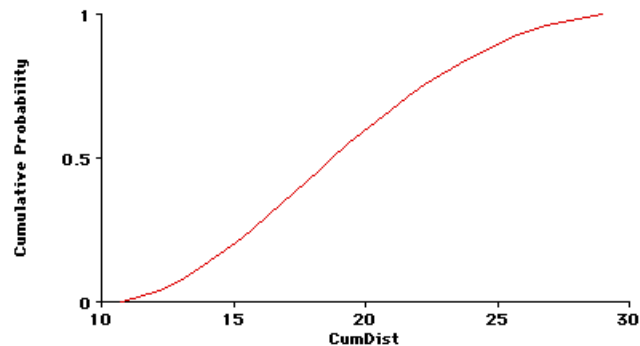
	1	2	3
	0	0.6	1.0

*Array\_x:*

*Index\_a* ►

	1	2	3
	10	20	30

`CumDist(Array_b, Array_x)` →



**Fractiles ( $L$ )** Specifies a continuous probability distribution by an array of evenly spaced fractiles,  $L$ .  $L$  must be a 1 dimensional array of nondecreasing numbers. If  $L$  contains  $n+1$  numbers, then  $L_i$  is the  $i/n$  fractile – that is, the for an uncertain quantity,  $x$ ,  $P(x \leq L_i) = i/n$ . `Fractiles()` uses linear interpolation on the cumulative



distribution between the specified fractiles, which implies a piecewise uniform distribution.

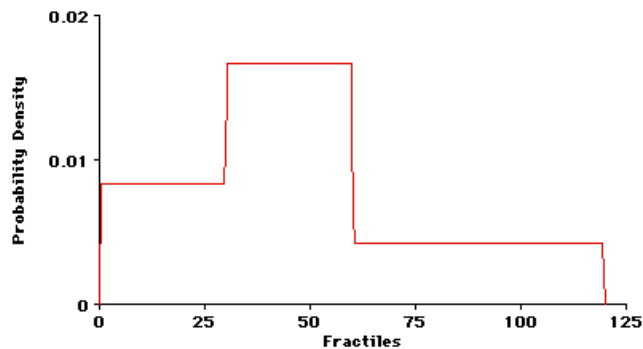
If any value in  $L$  is probabilistic, its mid value is used to obtain the fractile.

**Library:** Distribution

**Example:**

The following definition describes a distribution over the range 0 to 120 (0 and 100% fractiles), with its median at 45 (50% fractile), and quartiles at 30 and 60 (25% and 75% fractiles):

Fractiles( [0, 30, 45, 60, 120] ) →



**Lognormal** (*median*,  
*gsdev*)

Creates a lognormal distribution with median of *median* and geometric standard deviation of *gsdev*. The geometric standard deviation must be 1 or greater. The range [*median*/*gsdev*, *median* × *gsdev*] encloses about 68% of the probability. *Gsdev* is sometimes also known as the *uncertainty factor* or *error factor*.) *Median* and *gsdev* must be positive.

The log of a lognormal quantity has a normal distribution with mean of  $\text{Log}(\textit{median})$  and standard deviation of  $\text{Log}(\textit{gsdev})$ .

While the lognormal distribution is unbounded above, Analytica's `Lognormal()` function truncates sample values at  $\textit{median}/\textit{gsdev}^3$  on the bottom and  $\textit{median} \times \textit{gsdev}^3$  at the top.

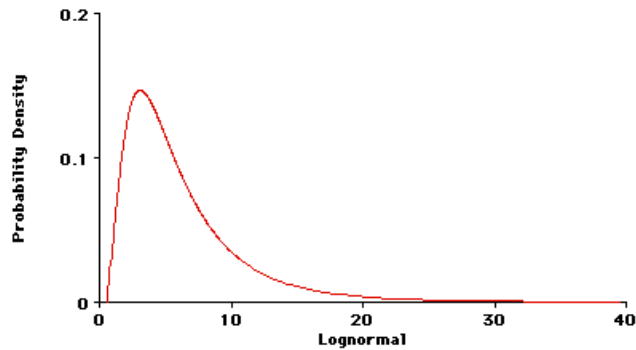
**When to use:**

Use the lognormal distribution if you have a sharp lower bound of zero but no sharp upper bound, a single mode, and a positive skew. This distribution is particularly appropriate if you believe that the uncertain quantity is the product (or ratio) of a large number of independent random variables.

**Library:** Distribution

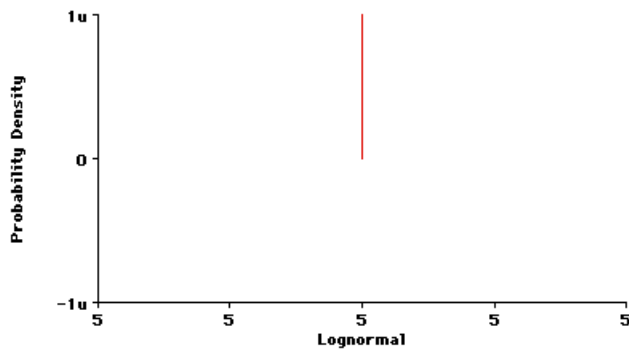
**Examples:**

`Lognormal(5, 2)` →



The case of  $gsdev=1$  gives a delta function (spike) at *median*.

`Lognormal(5, 1)` →



**Normal ( *mean*, *stddev* )** Creates a normal or Gaussian probability distribution with *mean* and standard deviation *stddev*. The standard deviation must be 0 or greater. The range [*mean-stddev*, *mean+stddev*] encloses about 68% of the probability.

While the normal distribution is unbounded above and below, Analytica's `Normal ( )` function truncates sample values at  $3 \times \textit{stddev}$  above and below the mean.

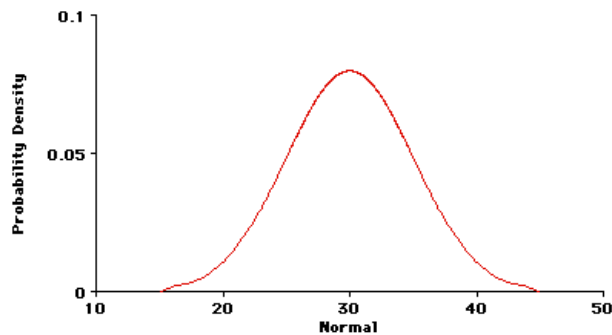
**When to use:**

Use a normal distribution if the uncertain quantity is unimodal and symmetric and the upper and lower bounds are unknown, possibly very large or very small (unbounded). This distribution is particularly appropriate if you believe that the uncertain quantity is the sum or average of a large number of independent, random quantities.

**Library:** Distribution

**Example:**

`Normal ( 30, 5 )` →



**Probdist ( *P*,*R*,*I* )** Specifies a continuous probability distribution as an array of probability density values, *P*, for an array of corresponding outcome values, *R*, for the quantity. Probdist performs a linear interpolation between the points on the density function. The values of *P* must be nonnegative. They will be normalized so that

the total probability enclosed is 1.0. The values of  $R$  must be increasing.

The values of  $P$  should start and end at 0. If the first (or last) value of  $P$  is not zero, Analytica assumes zero at  $2R_1 - R_2$  (or  $2R_n - R_{n-1}$ ).

Either  $R$  must be an index of  $P$ , or  $P$  and  $R$  must have an index in common. If  $P$  or  $R$  have more than one index, you must specify the relevant index for linking  $P$  and  $R$  as a third parameter,  $I$ .

**Library:** Distribution

**Example:**

Array\_p:

Index\_a ►

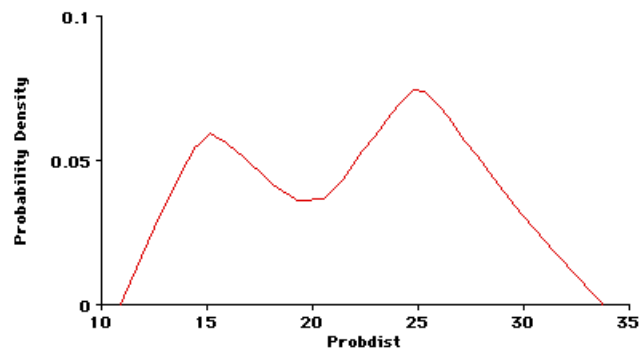
	1	2	3	4	5	6
	0	0.4	0.2	0.5	0.2	0

Array\_r:

Index\_a ►

	1	2	3	4	5	6
	10	15	20	25	30	35

Probdist(Array\_p, Array\_r) →



**Triangular ( *min*, *mode*, *max* )**

Creates a triangular distribution, with minimum *min*, mode *mode*, and maximum *max*. *Min* must not be greater than *mode*, and *mode* must not be greater than *max*.

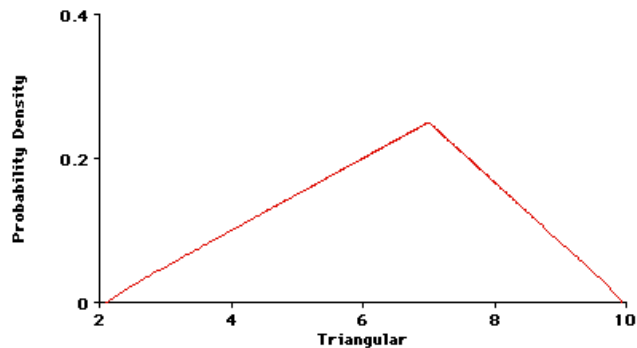
**When to use:**

Use the triangular distribution when you have the bounds and the mode, but have little other information about the uncertain quantity.

**Library:** Distribution

**Example:**

`Triangular(2, 7, 10) →`



**Uniform ( *min*, *max* )** Creates a uniform distribution between values *min* and *max*.

**When to use:**

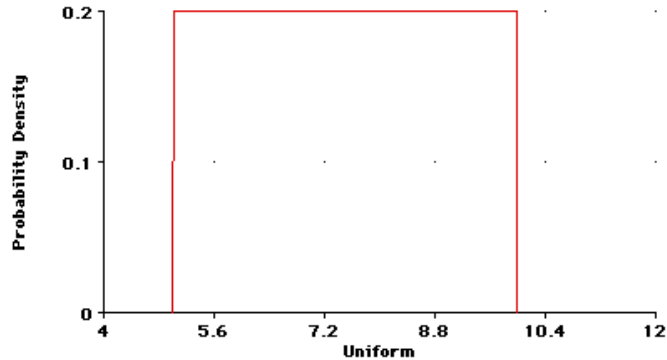
If you know nothing about the uncertain quantity other than its bounds, a uniform distribution between the bounds is appealing. However, situations in which this is truly appropriate are rare. Usually one end, or the middle, of the range is more likely than

the rest; that is, the quantity has a mode. In such cases, a beta or triangular distribution is a better choice.

**Library:** Distribution

**Example:**

`Uniform(5, 10) →`



## 14.2 Related function

**Truncate (*Dist*, *X*)** Truncates a probabilistic value *Dist* at and below deterministic value *X*. If *Dist* is not a distribution, *Truncate* returns *Dist*.

Truncate does not discard sample values; it generates a new complete sample for the quantity with the same probability distribution as *Dist* above *X*, and 0 below *X*.

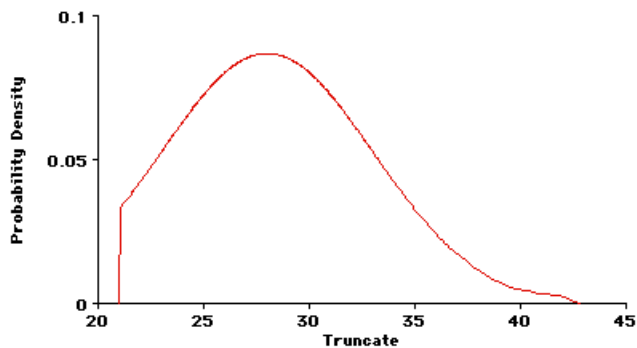
Since `Truncate()` resamples from the truncated distribution, the result will be nearly independent of *Dist*. Hence, importance and other measures that depend on correlations with *Dist* or with probabilistic variables on which *Dist* depends will be near zero, which may be misleading.

Library: Special

### Examples:

*Mpg*: Normal(28, 5)

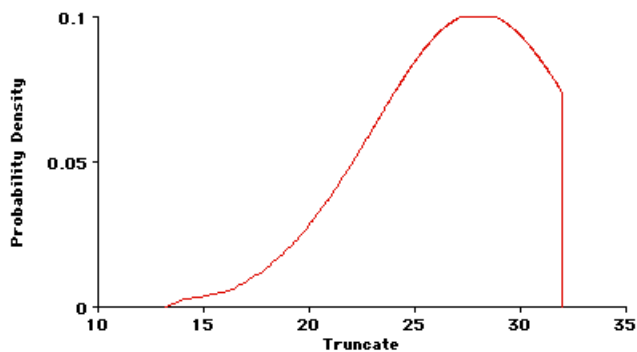
Truncate(*Mpg*, 21) →



To truncate a distribution at or above a specified value, use:

`-Truncate(-Dist, -X)`

`-Truncate(-Mpg, -32) →`









---

# 15

## Using Discrete Probability

---

This chapter describes Analytica’s discrete probability table functions and discrete probability distributions. Additional discrete probability distributions are included in the Libraries folder distributed with Analytica.

### 15.1 Using a probability table

To describe a variable as a discrete uncertainty, Analytica provides a special kind of edit table called a *probability table*. (See also Chapter 11, “Modeling with Arrays and Tables.”)

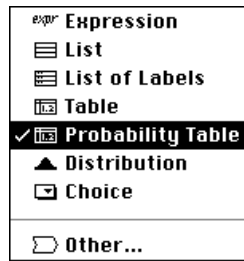
#### Creating a probability table

To define a variable as a discrete probability distribution in a probability table:


1. Determine the variable’s *domain* — the list of possible outcomes.
2. Select the variable and open one of the following:
  - The variable’s Object window.
  - The Attribute panel of the Diagram window (see page 1-14).

In the Attribute panel, select **Definition** from the Attribute popup menu (see page 1-15) as the attribute to display.

- Press the Expression popup menu above the definition field and select **Probability Table**.

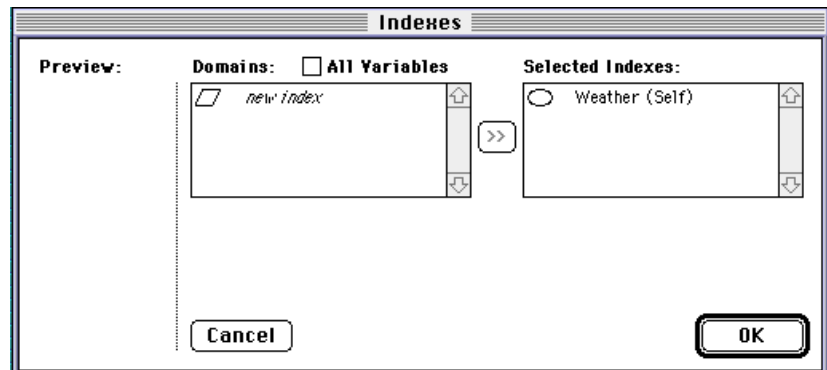



If the variable already has a definition, a dialog box confirms that you wish to replace it.

**Note:** 

If the definition of a variable is already a probability table, a **ProbTable** button appears in the definition. Click on it to see the Edit Table window (see page 11-4).

- The Indexes dialog box opens to confirm your choices for the indexes of the table. Only variables with a domain of List of numbers or List of labels are shown by default. The variable being defined is already listed as a selected index (with *Self* in parentheses). Add or remove any other discrete inputs (or other Index variables).

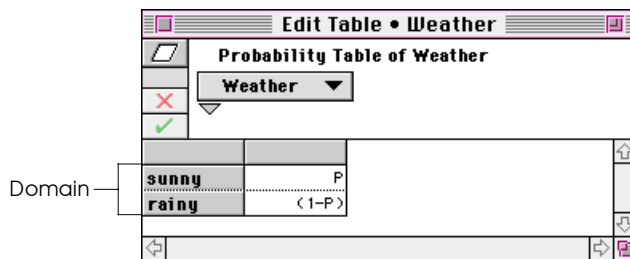


**Note:**  *Self* is required as an index of a probability table – it refers to this variable’s domain values.

5. Click on the **OK** button. An Edit Table window appears.
6. Enter the possible outcomes (the domain) in the first column. If the outcomes are numeric, they must be in increasing order.
7. Enter the probability of each possible outcome in the second column. (The probabilities should sum to 1.)

**Example:**

If  $P$  is a variable whose value is a probability (between 0 and 1) and the possible weather outcomes are sunny and rainy, then the following is a probability table for weather:

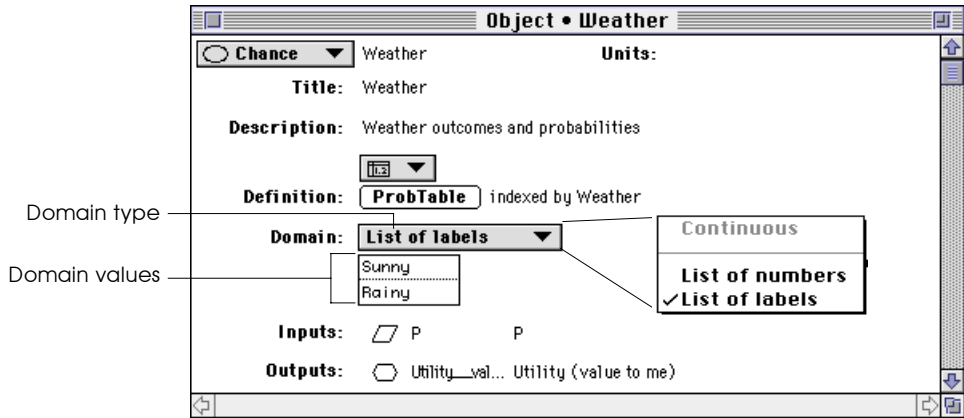


Probability Table of Weather	
Weather	
sunny	P
rainy	<1-P>

## Editing the domain

The domain attribute has values (the possible outcomes) and a type. In a probability table, you can edit the values directly in the first column of the edit table window, as an index of the table. Each entry must be a number or label (text); it cannot be an expression.

You can also edit the domain values in the Object window and Attribute panel.



## Changing the domain type

The domain popup menu shows the domain type. For a probability table, the domain type is either a list of numbers or a list of labels, and is set by your entry in the first row's cell in the edit table.

To change the domain type, press on the popup menu and select the desired type.

## Expression view of probability table

When you select the expression view of a definition that was created as a probability table, it has the following appearance. You cannot create a probability table as an expression.

### **Probtable** (*I1, I2, ... In*) (*p1, p2, p3, ... pm*)

Describes an  $n$ -dimensional conditional probability table, indexed by the indexes  $I1, I2, \dots In$ . One index must be *Self*.

$p1, p2, p3, \dots pm$  are the probabilities in the array.

### Example:

The *Weather* probability table on page 15-3, when viewed as an expression, looks like this:

```
Probtable (Self)(P, (1-P))
```

Note that the domain values do not appear in the expression view.

## Using labels in a probability table

A discrete probability distribution can describe the probability that a variable falls into a category, for example:

Low	sunny
Medium	rainy
High	

In a probability table, Analytica assumes that label outcomes are ordered, with the first value being the minimum and the last value being the maximum. In the first example above, this ordering has meaning; in the second example above, it does not. This ordering is used to compute the following statistics. Use these statistics with caution, since they are a function of the order sequence of the qualitative outcomes.

### **Statistics available for label valued distributions**

Frequency (use `Frequency(X,X)`)  
 Mid Value (Median)  
 Min  
 Max  
 Probability bands  
 Sample

### **Statistics *not* available for label valued distributions**


Correlation  
 Kurtosis  
 Getfract  
 Mean  
 Rankcorrel  
 Skewness  
 Standard deviation  
 Variance

Also use caution applying the logical operators (`>`, `<`, `=`) to a label valued distribution. The logical operators use the ASCII sort sequence, not the ordering of label outcomes.

## Adding dimensions to a probability table

You may wish to add dimensions to a probability table. For example, in the *Weather* probability table (see page 15-3), you may wish to distinguish between daylight and evening, with different probabilities for rainy weather in daylight and evening. So you would add a dimension with two values: daylight and evening.


You can add indexes or decision variables defined as lists, similar to adding indexes to an edit table, as follows:

1. Open the Edit Table window by clicking on the **ProbTable** button.
2. Click on the Indexes () button to open the Indexes dialog box.
3. Click in the **All variables** checkbox above the left hand list.
4. Move the desired variables to add them as indexes.
5. Click on the **OK** button to accept the changes.

### Creating a conditional dependency

After you have defined several probability tables, you may want to make some probabilities in a probability table conditional on the outcomes of other variables. This is called a *conditional dependency*.

To create a conditional dependency in a probability table:

1. Open the Edit Table window by clicking on the **ProbTable** button.
2. Click on the Indexes () button. Other variables that are defined as probability tables appear in the list of domains.
3. Move the variables you wish to be conditionally dependent, to add them as indexes.
4. Click on the **OK** button to accept the changes.

The resulting table is indexed by both the domain of your variable and the domains of the conditionally dependent variables.

---

**Note:** 

You must have already specified the variables as probability tables, before adding them with the Indexes dialog box.

---

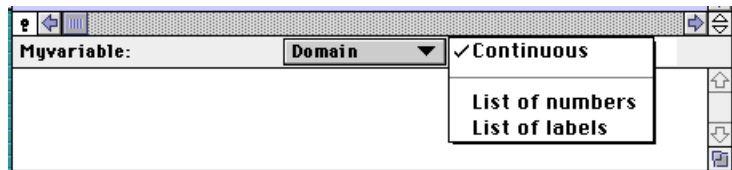
## 15.2 Other discrete distribution functions

To define a variable as a discrete probability distribution other than probability table and view its Probability Mass Function, you must first set its domain type and then assign the range of possible outcomes to its domain.


To set a discrete domain type and assign domain values:

1. Select the variable and open the Attribute panel of the Diagram window (see page 1-14).
2. Select the **Domain** attribute from the popup menu.

The domain type popup menu shows the default of Continuous. Select either List of numbers or List of Labels.



3. Analytica displays a list containing one element. Enter the domain values like any list (see page 11-11).

**Note:** 

The domain must include all the values that appear in the sample of the discrete probability distribution. If it does not, then the total Probability Mass Function will be less than 1.

**Bernoulli (  $P$  )**

Creates a discrete probability distribution with probability  $P$  of result 1 and probability  $(1 - P)$  of result 0.  $P$  is a probability value or array of probabilities, each between 0 and 1. The Bernoulli distribution is defined as:

$$\text{If Uniform}(0, 1) < P \text{ Then } 1 \text{ Else } 0$$



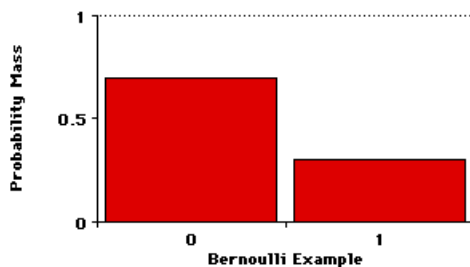
If  $P$  is greater than 1, the distribution is made up of all 1's. If  $P$  is less than 0, the distribution is made up of all 0's.

**Library:** Distribution

**Example:**

The domain, List of numbers, is [0, 1].

*Bernoulli\_ex*: Bernoulli (0.3) →



**Certain ( $U$ )** Returns the value of  $U$ .

**Library:** Distribution

**When to use:**

Use `Certain()` when an input node is defined as a distribution (see “Using input nodes,” Section 9.1), and, in browse mode, you want to replace the distribution with a nonprobabilistic value.

**Example:**

*Index\_a*:

1	2	3
---	---	---

*Array\_p*:

*Index\_a* ►

	1	2	3
	0.3	0.4	0.3

`Certain (Array_p )` →

	1	2	3
	0.3	0.4	0.3

**Chancedist ( *P*, *A*, *I* )** Creates a discrete probability distribution. *A* is an array of outcomes, and *P* is the corresponding array of probabilities. *A* and *P* must both be indexed by *I*.

The values of *A* must be unique; if *A* is numeric the values must be increasing.

#### When to use:

Use `Chancedist()` instead of the probability table when:

- The array of outcomes *A* is multidimensional, or
- The outcomes and probabilities arrays are defined as other variables; the variables can be used in other parts of your model.

**Library:** Distribution

#### Example:

*Index\_b*:

Red	White	Blue
-----	-------	------

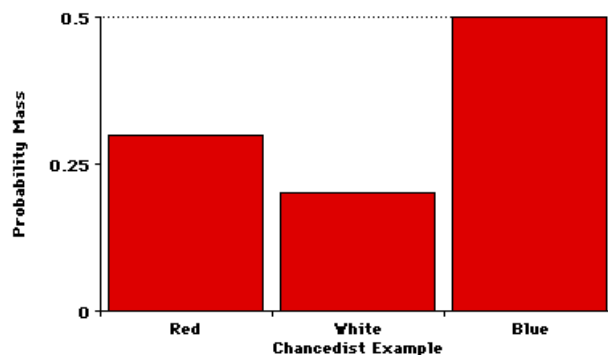
*Array\_q*:

*Index\_b* ►

	Red	White	Blue
	0.3	0.2	0.5

The domain, List of labels, is [ 'Red', 'White', 'Blue' ].

`Chancedist(Array_q, Index_b, Index_b)` →



## 15.3 Using a deterministic conditional table

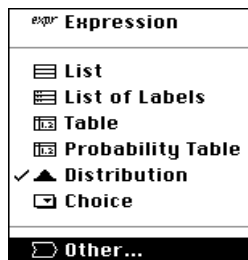
Sometimes a variable's value is deterministic (not uncertain) and conditionally dependent on the outcomes of discrete uncertain variables. The `Determtable()` function defines this dependency.

The `Determtable()` function appears similar to an edit table or a probability table. Each cell contains nonprobabilistic (deterministic) values. At least one index is a probability table (a discrete probabilistic variable). Other indexes are typically decision variables defined as lists. The `Determtable()` function returns an array that is reduced across its probabilistic index(es) — the evaluation result shows the value considering the uncertain distribution of each probabilistic index.

### Creating a `determtable`

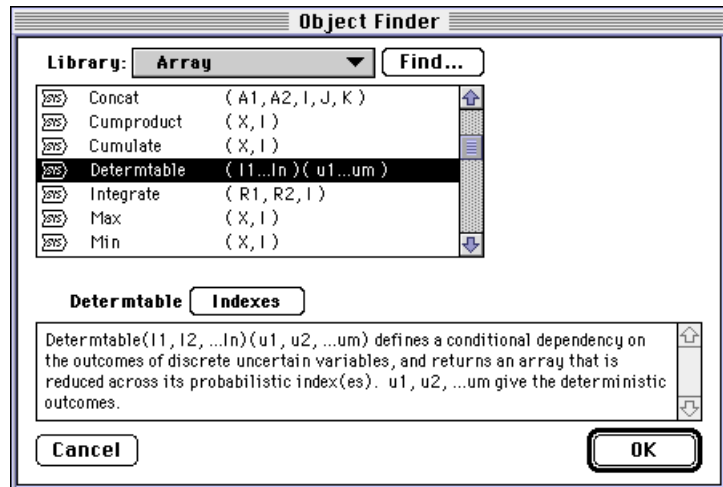
To define a variable as a `determtable`:

1. Determine the variable's domain — the list of possible outcomes.
2. Press the Expression popup menu above the definition field and select **Other**.

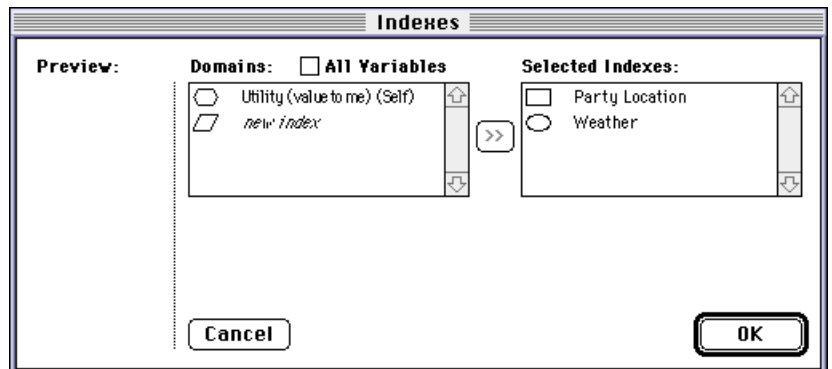


Analytica opens the Object Finder dialog box (see page 8-7).

3. Select **Array** from the **Library** popup menu and select **Determtable** from the function list.



4. Click on the **Indexes** button to specify discrete probability variables as inputs. The Indexes dialog box appears.



5. Click on **OK** to accept the indexes and open an Edit Table window.
6. Enter the outcomes corresponding to each outcome of your discrete inputs.

## Expression view of a determtable

When you select the expression view of a definition that was created as a determtable, it has the following appearance. You cannot initially create a determtable as an expression.

**Determtable**(  $I_1, I_2, \dots, I_n$  )  
(  $r_1, r_2, r_3, \dots, r_m$  )

Describes an  $n$ -dimensional conditional deterministic table, indexed by the indexes  $I_1, I_2, \dots, I_n$ . The last index,  $I_n$ , is the innermost index, varying the most rapidly.  $r_1, r_2 \dots r_m$  are the outcomes in the array. Determtable returns an array that is reduced across its indexes that are probability tables.

### Example:

In Section 15.1, *Weather* is defined as a probability table. If  $P_s$  the probability of “sunny”, is 0.4, then the probability of “rainy” is 0.6. *Party location* is a decision variable with values [ 'outdoors', 'porch', 'indoors' ]. *Value to Me* is a determtable, containing utility values (or “payoffs”) for each combination of *Party location* and *Weather*:

	sunny	rainy
outdoors	100	0
porch	90	20
indoors	40	50

Evaluating *Value to Me* gives the value of each party location, considering the uncertain distribution of *Weather*. The mean value of *Value to Me* is the expected utility:

outdoors	40
porch	48
indoors	46

---

# 16

## Analyzing Uncertainty and Sensitivity


---

This chapter describes Analytica's tools for analyzing the uncertainty of variables, relationships between uncertain variables, and sensitivity of outputs to changes in inputs. It covers the statistical functions, sensitivity analysis functions, scatter graphs, and importance analysis.

### 16.1 Statistical functions

This section describes Analytica's built-in statistical functions, for use in variable definitions. Many of these functions are used in the Result window Uncertainty View options (see Section 2.4). These functions can assist with analysis of probabilistic variables.


---

**Note:**  All statistical functions produce estimates from the underlying random sample for each probabilistic quantity. These estimates are not exact, but will vary from one evaluation to the next due to the variability inherent in random sampling. Hence, your results may not exactly match the results shown in the examples here. For greater precision, use a larger sample size (see Appendix D on how to select a sample size).

---

The calculation formulas use the following notation:

- $x_i$  the  $i$ th sample value of probabilistic variable  $X$
- $\bar{x}$  the mean of probabilistic variable  $X$  (see `Mean()`)
- $\sigma$  standard deviation (see `Sdeviation()`)
- $m$  sample size (see Appendix D).

**Note:**  These statistical functions will not calculate statistics for an array of data unless it is a sample indexed by *Run*. To obtain statistics on an array of data with another index, see the Data Statistics library in the Libraries folder.

The examples in this section use the following variables:

*Alt\_fuel\_price*: Normal(1.25, 0.1)

*Fuel\_price*: Normal(1.19, 0.1)

*Skfuel\_price*: Beta(4, 2, 1, 1.5)

**Correlation ( X,Y)** Returns an estimate of the correlation between the probabilistic expressions *X* and *Y*, where -1 means perfectly negatively correlated, 0 means no correlation, and 1 means perfectly positively correlated.

*Correlation(X, Y)*, a measure of probabilistic dependency between uncertain variables, is sometimes known as the Pearson product moment coefficient of correlation, *r*. It measures the strength of the linear relationship between *X* and *Y*, using the formula:

$$\frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \times \sum_i (y_i - \bar{y})^2}}$$

**Library:** Statistical

**Example** (With *SampleSize* set to 100 and number format set to two decimal digits):

*Correlation(Alt\_fuel\_price + Fuel\_price, Fuel\_price)* → 0.71

Correlation of two independent, uncorrelated distributions approaches 0 as the sample size approaches infinity.

**Example:**with *Samplesize* = 20:

```
Correlation(Normal(1.19,0.1), Normal(1.19,0.1))
→ -.28
```

with *Samplesize* = 1000

```
Correlation(Normal(1.19,0.1),Normal(1.19,0.1))
→ 0.03
```

**Frequency ( *X*, *l* )** If *X* is a discrete uncertain variable, returns an array indexed by *l*, giving the frequency, or number of occurrences of discrete values *l*. *l* must contain unique values; if numeric, the values must be increasing.

If *X* is a continuous uncertain variable and *l* is an index of numbers in increasing order, it returns an array indexed by *l*, with the count of values in the sample *X* that are equal to or less than each value of *l* and greater than the previous value of *l*.

If *X* is nonprobabilistic, `Frequency()` returns *Samplesize* for each value of *l* equal to *X*.

Since `Frequency()` is computed by counting occurrences in the probabilistic sample, it is a function of *Samplesize* (see Section 13.6). If you want the relative frequency rather than the count of each value, divide the result by *Samplesize*.

**Library:** Statistical

**Example (Continuous):**

*Index\_a*: [1.2, 1.25]

```
Frequency(Fuel_price, Index_a) →
Index_a ►
```

	1.2	1.25
	54	19

**Example (Discrete):**

*Bern\_out*: [0, 1]

(Possible outcomes of the Bernoulli Distribution)



With *Samplesize* = 100:

Frequency(Bernoulli (0.3), Bern\_out) →

*Bern\_out* ►

	<b>0</b>	<b>1</b>
	70	30

With *Samplesize* = 25:

Frequency(Bernoulli (0.3), Bern\_out) →

*Bern\_out* ►

	<b>0</b>	<b>1</b>
	18	7

(Compare the Bernoulli example in Chapter 15).

**Getfract ( X, P )** Returns an estimate of the *P*th fractile (also known as quantile or percentile) of *X*. This is the value of *X* such that *X* has a probability *P* of being less than that value. If *X* is nonprobabilistic, all fractiles are equal to *X*.

The value of *P* must be a number or array of numbers between 0 and 1, inclusive.

**Library:** Statistical

**Examples:**

Getfract(*X*, 0.5) returns an estimate of the median of *X*.

Getfract(Fuel\_price, 0.5) → 1.19

The following returns a table containing estimates of the 10%ile and 90%ile values, that is, an 80% confidence interval.

*Fract*: [0.1, 0.9]

Getfract(Fuel\_price, *Fract*) →

*Fract* ►

	<b>0.10</b>	<b>0.90</b>
	1.06	1.32

**Kurtosis ( X )** Returns an estimate of the kurtosis of  $X$ .  $X$  must be probabilistic.

Kurtosis is a measure of the peakedness of a distribution. A distribution with long thin tails has a positive kurtosis. A distribution with short tails and high shoulders, such as the uniform distribution, has a negative kurtosis. A normal distribution has zero kurtosis.

`Kurtosis(X)` uses the formula:

$$\left( \frac{1}{m} \sum_{i=1}^m \left[ \frac{x_i - \bar{x}}{\sigma} \right]^4 \right) - 3$$

**Library:** Statistical

**Example:**

`Kurtosis(Skfuel_prices) → -0.48`

**Mean ( X )** Returns an estimate of the mean of  $X$  if  $X$  is probabilistic. Otherwise, returns  $X$ .

`Mean(X)` uses the formula:

$$\frac{1}{m} \sum_{i=1}^m x_i = \bar{x}$$

**Library:** Statistical

**Examples:**

`Mean(Fuel_price) → 1.19`

`Mean(Skfuel_price) → 1.33`

**Mid ( X )** Returns the mid value of  $X$ . `Mid(X)` forces deterministic evaluation in contexts where  $X$  would otherwise be evaluated probabilistically.

The mid value is calculated by substituting the *median* for most full probability distributions in the definition of a variable or expression, and using the mid value of any inputs. The mid value of a variable or expression is *not* necessarily equal to its true median, but is usually close to it.

**Library:** Statistical

**Example:**

`Mid(Fuel_price)` → 1.19

**Probability(*B*)** Returns an estimate of the probability or array of probabilities that the Boolean value *B* is true.

**Library:** Statistical

**Example:**

`Probability(Fuel_price < 1.19)` → 0.5

**Probbands (*X*)** Returns an estimate of probability or “confidence” bands for *X* if *X* is probabilistic. Otherwise returns *X* for every band. The probabilities are specified in the Uncertainty Setup dialog box, Probability Bands option (see Section 13.6).

**Library:** Statistical

**Example:**

`Probbands(Fuel_price)` →  
*Probability* ►

	0.05	0.25	0.5	0.75	0.95
	1.025	1.123	1.19	1.257	1.355

**Rankcorrel (*X*, *Y*)** Returns an estimate of the rank-order correlation coefficient between the distributions *X* and *Y*. *X* and *Y* must be probabilistic.

`Rankcorrel(X, Y)`, a measure of the dependence between *X* and *Y*, is sometimes known as Spearman’s rank correlation coefficient,  $r_s$ .

Rank-order correlation is measured by computing the ranks of the probability samples, and then computing their correlation. By using the rank order of the samples, the measure of correlation is not affected by skewed distributions or extreme values, and is, therefore, more robust than simple correlation. Rank-order correlation is used for importance analysis (see Section 16.5).

**Library:** Statistical

**Example:**

with *Samplesize* = 100:

`Rankcorrel(Fuel_price, Alt_fuel_price) → .02`

**Sample ( X )** Evaluates *X* probabilistically and returns a sample of values from the distribution of *X* in an array indexed by the system variable *Run*. If *X* is not probabilistic, returns *X*. The system variable *Samplesize* specifies the size of this sample. You can set *Samplesize* in the Uncertainty dialog box (see Section 13.6).

**Library:** Statistical

**When to use:**

Use when you want to force probabilistic evaluation, or look at raw sample values.

**Example:**

Here are the first six values of a sample:

`Sample(Fuel_price) →`

*Iteration(Run)* ►

	1	2	3	4	5	6
	1.191	1.32	1.19	1.164	1.191	0.962

**Sdeviation ( X )** Returns an estimate of the standard deviation of *X* from its sample if *X* is probabilistic. If *X* is nonprobabilistic, returns 0.

`sdeviation(X)` uses the formula:

$$\sqrt{\frac{1}{m-1} \sum_{i=1}^m (x_i - \bar{x})^2} = \sigma$$

**Library:** Statistical

**Example:**

`Sdeviation(Fuel_price) → 0.10`

**Skewness (X)** Returns an estimate of the skewness of  $X$ .  $X$  must be probabilistic.

Skewness is a measure of the asymmetry of the distribution. A positively skewed distribution has a thicker upper tail than lower tail, while a negatively skewed distribution has a thicker lower tail than upper tail. A normal distribution has a skewness of zero.

Skewness( $X$ ) uses the formula:

$$\frac{1}{m} \sum_{i=1}^m \left[ \frac{x_i - \bar{x}}{\sigma} \right]^3$$

**Library:** Statistical

**Example:**

Skewness(Skfuel\_price) → -0.45

**Statistics (X)** Returns an array of statistics of  $X$ . Select the statistics in the Uncertainty Setup dialog box, Statistics option (see Section 13.6).

**Library:** Statistical

**Example:**

Statistics(Fuel\_price) →  
Statistics ►

	Min	Median	Mean	Max	Std. Dev.
	0.93	1.19	1.19	1.45	0.10

**Variance (X)** Returns an estimate of the variance of  $X$  if  $X$  is probabilistic. If  $X$  is nonprobabilistic, returns 0.

Variance( $X$ ) uses the formula:

$$\frac{1}{m-1} \sum_{i=1}^m (x_i - \bar{x})^2 = \sigma^2$$

**Library:** Statistical

**Example:**

Variance(Fuel\_price) → 0.01

## 16.2 Sensitivity analysis functions

Sensitivity analysis enables you to examine the effect of a change in the value of an input variable on the values of its output variables.

**Examples:** The examples in this section refer to the following variables:

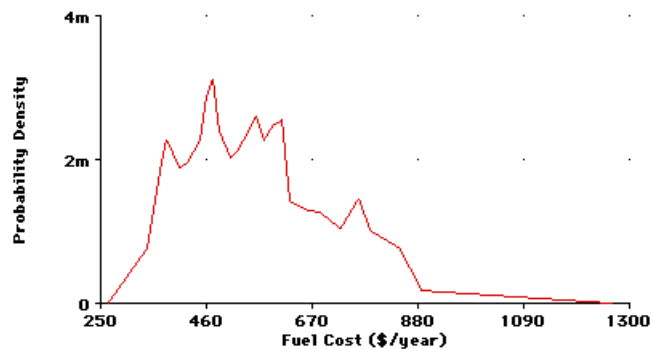
*Gasprice*: Normal(1.3, .3)  
(cost of gasoline per gallon within market fluctuations)

*Mpy*: 12K  
(the average number of miles driven per year)

*Mpg*: Normal(28, 5)  
(fuel consumption averaged over driving conditions)

*Fuelcost*: Gasprice \* Mpy / Mpg  
(annual cost of fuel)

Probability density of *Fuelcost*:



**Dydx (Y, X)** Returns the derivative of expression  $Y$  with respect to variable  $X$ , evaluated at mid values. This function returns the ratio of the change in  $Y$  to a small change in  $X$  that affects  $Y$ . The “small change” is  $\frac{X}{10000}$ , or 10E-6 if  $X = 0$ .

**Library:** Special

**Examples:**

Because *Fuelcost* depends on *Mpg*, a small change in *Mpg* seems to have a modest negative effect on *Fuelcost*:

`Dydx(Fuelcost, Mpg) → -19.7`

The reverse is not true, because *Mpg* is not dependent on *Fuelcost*. That is, *Fuelcost* does not cause any change in *Mpg*:

`Dydx(Mpg, Fuelcost) → 0`

In this model of *Fuelcost*, a small change in *Gasprice* has by far the largest effect of all its inputs:

`Dydx(Fuelcost, Gasprice) → 428.6`

`Dydx(Fuelcost, Mpy) → 0.04643`

**Elasticity ( Y, X )** Returns the percent change in variable *Y* caused by a 1 percent change in a dependent variable *X*.

`Elasticity()` is related to `Dydx()` in the following manner:

`Elasticity(Y,X) = Dydx(Y,X)*(X/Y)`

**Library:** Special

**Examples:**

`Elasticity(Fuelcost, Mpg) → -0.9901`

`Elasticity(Fuelcost, Gasprice) → 1`

A 1% change in variables *Mpg* and *Gasprice* cause about the same degree of change in *Fuelcost*, although in opposite directions.

*Mpg* is inversely proportional to the value of *Fuelcost*, while *Gasprice* is proportional to it.

**Whatif ( Ident, Tempval, X )** Returns the value of expression *Ident* when variable *Tempval* is set to the value of expression *X*. *Tempval* must be a variable. The original definition of *Tempval* is restored after evaluation of the `whatif()` expression, allowing you to explore the effect of a change in *Tempval* without permanently changing it.

**Library:** Special

**Example:**

Fuelcost → 557.1

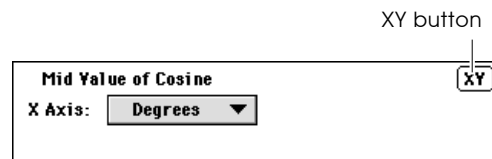
Whatif(Fuelcost, Mpy, 14K) → 650

## 16.3 X-Y results

When evaluating a variable, you can specify another variable to view it against, for Mid, Mean, Statistics, Probability Bands, and Sample.

To graph one variable against another:

1. Open a Result window for the  $y$ - (vertical axis) variable.
2. Click on the **XY** button located in the top right corner of the window to open the Object Finder dialog box.



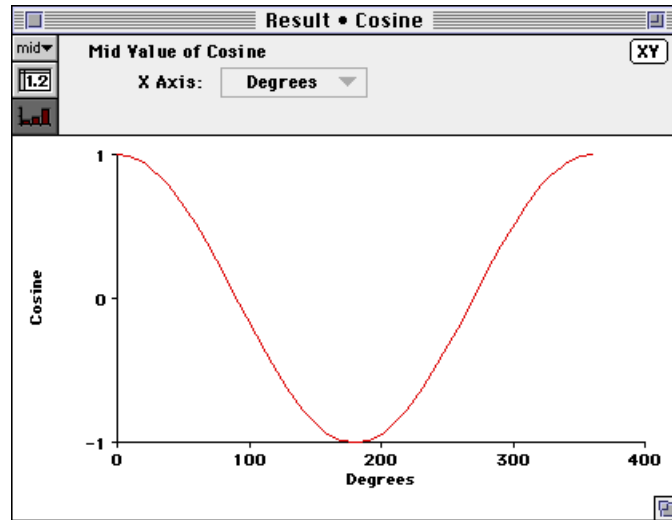
3. In the Object Finder, select the  $x$ - (horizontal axis) variable

The two variables in an XY window must share at least one index, and all indexes of  $X$  must also be indexes of  $Y$ . The popup menu in the index selection area becomes **Common Index**—only indexes of both  $X$  and  $Y$  may be selected.

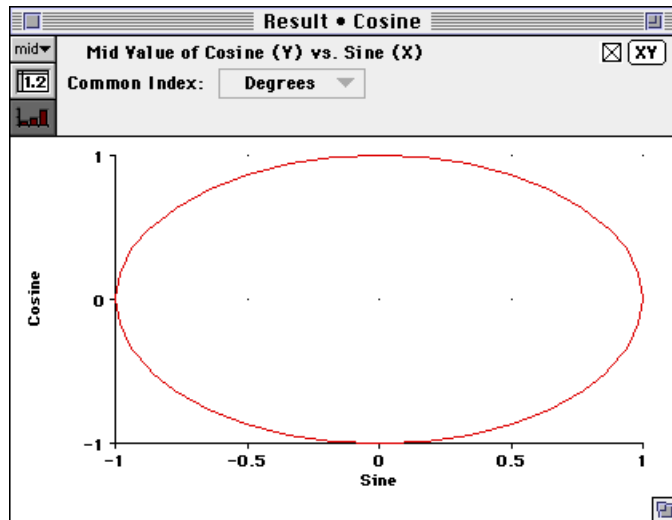
**Example:** *Degrees:* Sequence(0, 360, 10)  
*Sine:* Sin(Degrees)



*Cosine*:  $\text{Cos}(\text{Degrees}) \rightarrow$



Click on the **XY** button. In the Object Finder dialog under Current Module select the variable *Sine* to display:



Click on the Table View button to display:

	X	Y
0	0	1
10	0.1736	0.9848
20	0.342	0.9397
30	0.5	0.866
40	0.6428	0.766

To return to the graph or table of *Cosine vs. Degrees*, click in the XY checkbox.

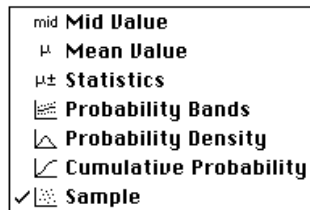
## 16.4 Scatter plots

A scatter plot, graphing the samples of two probabilistic variables against each other, can provide insight into their probabilistic relationship.

To generate a scatter plot for two variables,  $X$  and  $Y$ :

1. Open a Result window for  $Y$ .
2. Click on the **XY** button located in the top right corner of the window to open the Object Finder dialog box.
3. In the Object Finder, select the  $X$  variable.

4. In the Uncertainty View popup menu (at the top left of the Result window), select the Sample view.



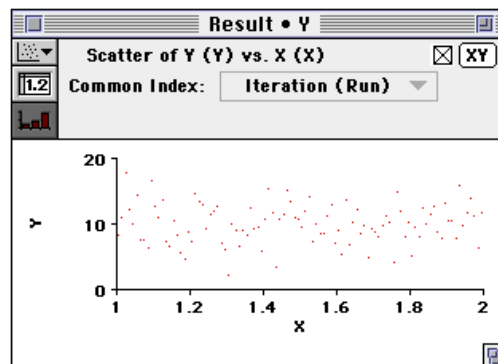
If the variables are independent, the scatter plot points will fall randomly on the graph. If the variables are totally dependent, the scatter plot points will fall along a single line. The strength of the relationship is indicated by the degree to which the points are close to a line. If the line is straight, the relationship is linear; if the line is curved, the relationship is nonlinear.

You can superimpose several scatter plots of  $Y$  in an array of uncertain quantities depending on  $X$ . The different quantities will be represented by different color dots or symbols.

**Example:**  $X$ : Uniform(1, 2)

$Y$ : Normal(10, 3)

The resulting scatter plot, of two independent variables, is:



## 16.5 Importance analysis

In most complex models, many of the input variables are uncertain. It is often useful to understand how much each uncertain input contributes to the uncertainty in the output. Typically, a few uncertain inputs are responsible for the lion's share of the uncertainty in the output, while the rest have little impact.

The importance analysis features in Analytica can help you quickly learn which inputs contribute the most uncertainty to the output. You can then concentrate on getting better estimates or building a more detailed model for the one or two most important inputs without spending considerable time investigating issues that turn out not to matter very much.

### Importance analysis defined

Importance is the absolute rank-order correlation between the sample of output values and the sample for each uncertain input. It is a robust measure of the uncertain contribution because it is insensitive to extreme values and skewed distributions. Unlike commonly used deterministic measures of sensitivity, it averages over the entire joint probability distribution. Therefore, it works well even for models where the sensitivity to one input depends strongly on the value of another.

### Creating an importance variable

To create an importance analysis variable:

1. Be sure you are in Edit mode. Select an output variable's node (usually your model's objective node, but it can be any variable that has uncertain inputs).
2. Select **Make Importance** from the **Object** menu.

Analytica creates two new variables, an index variable and a general variable. If *Output Variable* is the title of the node you selected, the index variable is titled *Output Variable Inputs*, and the general variable is titled *Output Variable Importance*.

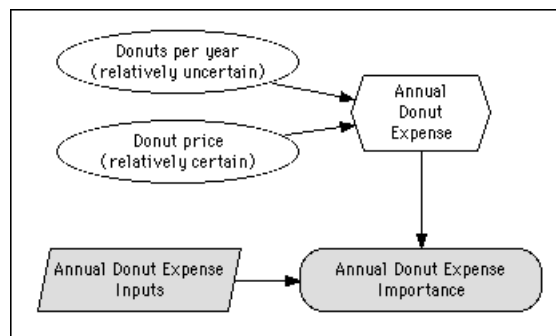
**Example:** *Donuts\_per\_year*: Normal(150,50)

*Donut\_price*: Normal(0.4,0.04)

*Annual\_donut\_expense*: *Donuts\_per\_year* \* *Donut\_price*

Since the two inputs are multiplied, we would expect the input with the greater relative uncertainty, *Donuts\_per\_year*, to contribute more uncertainty to *Annual\_donut\_expense*.

After selecting *Annual\_donut\_expense* and then **Make Importance** from the **Object** menu, the diagram contains two new variables.



*Annual\_donut\_expense Inputs* is a one-dimensional Edit Table of the chance variables. Its index contains the titles of the chance nodes, and its values are the identifiers of those nodes. .

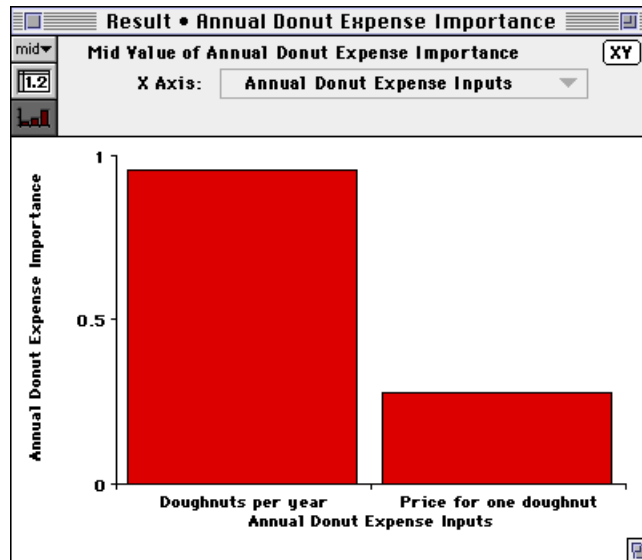
Edit Table • Annual Donut Expense Inputs	
Edit Table of Annual Donut Expense Inputs	
Annual Donut Expense Inputs	
<b>Doughnuts per year</b>	Donuts_per_year
<b>Price for one doughnut</b>	Donut_price

*Annual donut expense Inputs* evaluates to a set of probability distributions, one for each chance variable.


*Annual donut expense Importance* is defined as  

$$\text{Abs}(\text{Rankcorrel}(\text{Annual\_donut\_expense\_inputs}, \text{Annual\_donut\_expense}))$$
.

The `Rankcorrel()` function computes the rank-order correlation of each input to the output, and then the `Abs()` function computes the absolute value, yielding a positive relative importance.



As expected, *Donuts\_per\_year* contributes considerably more uncertainty to *Annual\_donut\_expense* than *Donut\_price*.

**Note:**  Importance, like every other statistical measure, is estimated from the random sample. The estimates may vary slightly from one sample to another due to random noise. For a sample size of 100, an importance of 0.1 may not be significantly different from zero. But an importance of 0.5 is significantly different from zero. The main goal is to discover those uncertain inputs, typically only two to five, that are the primary contributors to the uncertainty in the output. For greater precision, use a larger sample size.

## Editing importance variables

If you create an importance analysis variable for a model, and subsequently refine the model by redefining or adding uncertain inputs, you may want to change the set of input variables used for the importance analysis. Or, you may want to remove variables that you already know don't contribute significantly to the uncertainty in the result. Do one of the following:

- Select *Output Variable* and then **Make Importance** from the **Object** menu. The *Output Variable Inputs* node will be updated.
- Open *Output Variable Inputs*' Edit Table and edit the list of input variables.

---


# 17

## Modeling Changes over Time

---

A *dynamic variable* is a quantity that changes over time, for example, the effect of inflation on car prices over a ten year period. The system function `Dynamic()` and system variable *Time* enable you to model changes over time.

---

**Important:**  Read Chapter 11, “Modeling with Arrays and Tables” before using these features.

---

The term *dynamic* is used in this chapter to refer to the `Dynamic()` function.

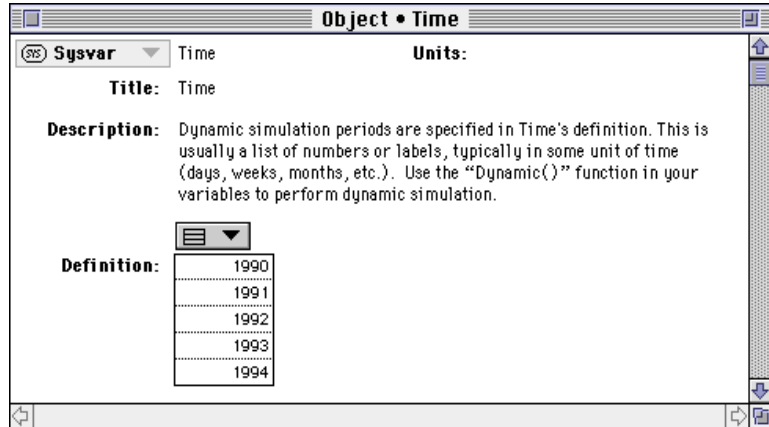
### 17.1 The Time index

Dynamic simulation time periods are specified in the system variable *Time*. To perform dynamic simulation, you must provide a definition for *Time*.


To edit the definition of *Time*, select **Edit Time** from the **Definition** menu to open the Object window for *Time*.



*Time* is defined by default as a list of three numbers 0, 1, and 2. You may want to define *Time* as a list of years, as in the following example:



*Time* becomes the index for the array that results from the `Dynamic()` function.

**Note:** 

A model can have only one definition *Time*, that is, one set of time periods for `Dynamic()` functions. Any number of variables in the model can be defined using `Dynamic()`.


## 17.2 Using the Dynamic function

**Dynamic** (*initial1, initial2...*,  
*initialn, Expr*)


Performs dynamic simulation, calculating the value of its defined variable at each element of *Time*. The result of `Dynamic()` is an array, indexed by *Time*.

*Initial1, ...initialn* are the values of the variable for the first *n* time periods. *Expr* is an expression giving the value of the variable for each subsequent time period. *Expr* can refer to the variable in earlier time periods, that is, contain its own identifier in its definition. If variable *Var* is defined using `Dynamic()`, *Expr* can be

a function of  $\text{Var}[\text{Time}-t]$  or  $\text{Self}[\text{Time}-t]$ , where  $t$  is any integer between 1 and  $n$ .

**Note:**  Square brackets ([ ]) are necessary around  $\text{Time}-t$ .

The `Dynamic()` function must appear at the topmost level of a definition. It cannot be used inside another expression.

When a dynamic variable refers to itself, it appears in its own list of inputs and outputs, with a symbol for cyclic dependency: .

**Library:** Special

#### When to use:

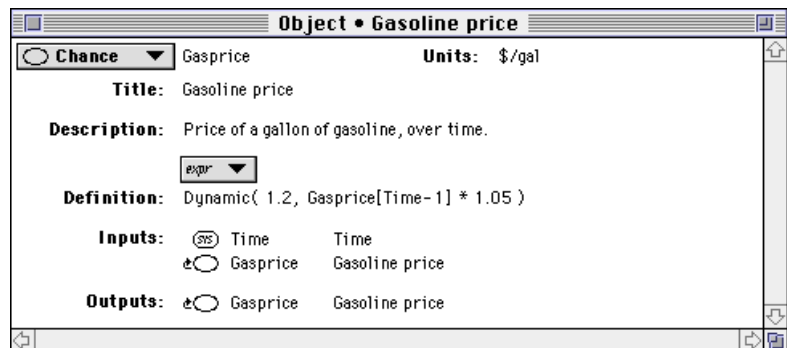
Use `Dynamic()` for defining variables that are cyclically dependent. This is the only function in Analytica that permits reference to the same variable, or other Dynamic variables, at earlier time periods.

#### Example:

`Dynamic()` can be used to calculate the effect of inflation on the price of gasoline in the years 1990 to 1994.

If the initial value is \$1.20 per gallon and the rate of inflation is 5% per year, then *Gasprice* can be defined as:

`Dynamic(1.2, Gasprice[Time-1] * 1.05)` or  
`Dynamic(1.2, Self[Time-1] * 1.05)`.



Clicking on the Result button and viewing the mid value as a table displays the following results:

Year	Mid Value of Gasoline price
1990	1.2
1991	1.26
1992	1.323
1993	1.389
1994	1.459

For 1990, Analytica uses the initial value of *Gasprice* (1.2). For each subsequent year, Analytica multiplies the value of *Gasprice* at *Time*-1 by 1.05 (the 5 percent inflation rate).

***Ident* [ *Time*-*t* ]** Given a variable *Ident* and brackets enclosing *Time* minus an integer *t*, returns the value for *Ident*, *t* time periods back from the current time period. This function is only valid for variables defined using the `Dynamic()` function.

Library: Special

## 17.3 More about the Time index

### Reference to earlier time

*Time*-*t* in the expression `var[Time-t]` refers to the position of the elements in the *Time* index, not values of *Time*.

For example, if *Time* equals [1990, 1994, 1998, 2002, 2006], then the value of *Time*-3 in year 2006 is 1994.

In any expression of [Time-t], *t* must be an integer; a variable name is not allowed. For example, [Time-B] results in a syntax error message.

**Cannot reference future time** In any expression of `[Time-t]`,  $t$  must be a positive number between 1 and  $n$ , the number of initial values. For example, `[Time--2]` results in a syntax error message.

**Defining time** There are three ways to define the *Time* index, each of which has different advantages:

- List (numeric)
- List of labels (text)
- Sequence

#### **Time as a list (numeric)**

When *Time* is defined as a list, it must consist of increasing numbers. The intervals between entries can be unequal, and the values for *Time* can be used in other expressions.

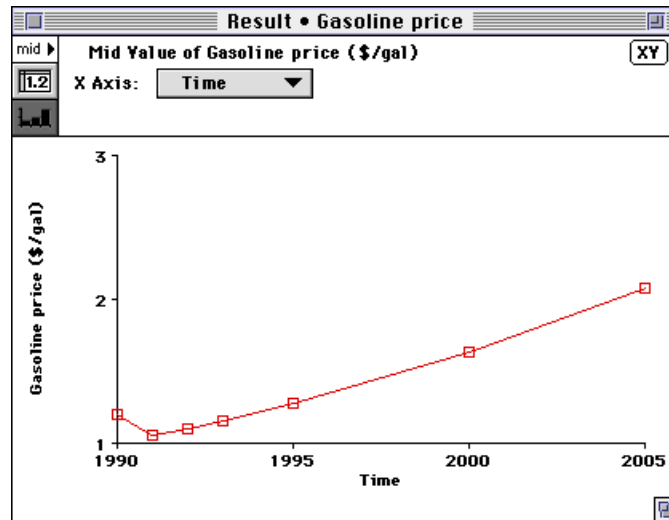
The resulting graph of any `DYNAMIC()` function, with the  $x$ -axis set to *Time*, will have tick marks at equal  $x$ -axis intervals for the value of *Time*.

#### **Example:**

*Time:*

1990
1991
1992
1993
1995
2000
2005

```
Gasprice: Dynamic(1.2, Gasprice[Time - 1] *
(1.05 ^ (Time - 1990))) →
```



### Time as a list of labels (text)

When *Time* is defined as a list of labels, *Time* values cannot be used in other expressions as numbers.

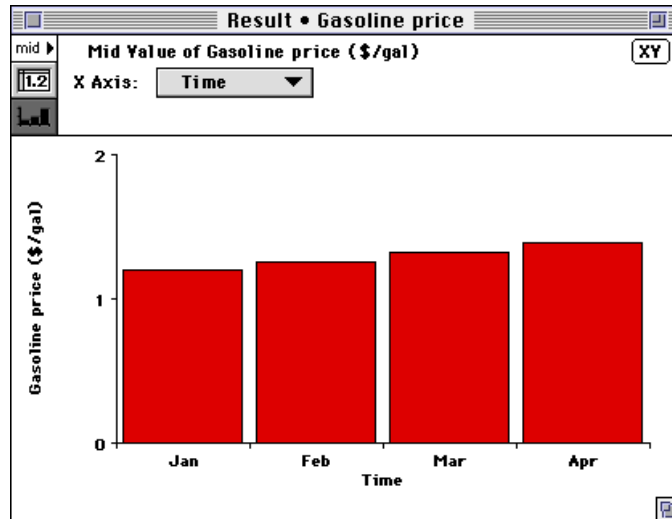
The resulting graph of any `Dynamic()` function, with the *x*-axis set to *Time*, will show the labels at equal *x*-axis intervals.

### Example:

*Time:*

Jan
Feb
Mar
Apr

*Gasprice*: `Dynamic(1.2, Gasprice[Time-1] * 1.05) →`



### Time as a sequence

Using the `Sequence()` function is the easiest way to define *Time* with equal intervals (see Sections 11.4 and 11.5). The numeric values for *Time* can be used in other expressions.

### Example:

Definition: `Sequence`(1, 20, 1)

## Using *Time* in a model

You can use *Time* like any index variable; you can change only its title and definition. To include the *Time* node on a diagram:

1. Open the Object window for *Time* by selecting **Edit Time** from the **Definition** menu.
2. Select **Make Alias** from the **Object** menu (see Section 4.7, “Using an alias node”).

When the *Time* node displays on a diagram, arrows from *Time* to all dynamic variables display by default.

## 17.4 Initial values for Dynamic

A dynamic definition of *var* usually includes the expression `Self[Time-t]` or `var[Time-t]`, where *t* is the number of time steps to subtract from the current *Time* value. You must supply at least *t* initial values.

As an example when *t* in `[Time-t]` is greater than 1, suppose your car insurance policy depends on the premium you paid two years ago. To calculate your payments in 1992, you must refer to the amount paid in 1990. A dynamic variable representing such a rate for insurance needs two initial values for *Time*, such as:

*Insurance:*

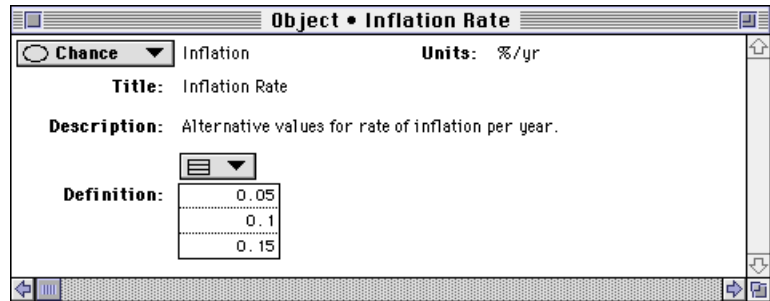
```
Dynamic(600, 700, Insurance[Time - 2] * 1.05) →
```

Year	Value
1990	600
1991	700
1992	630
1993	735
1994	661.5

## 17.5 Using arrays in Dynamic

The initial value of a dynamic variable, that is, the first parameter to the `Dynamic()` function, can be a number, variable identifier, or other expression that evaluates to a single number, list, or array. Analytica evaluates a dynamic variable starting from each initial value, in each time period, so the result is a correctly dimensioned array.

**Example:** Expanding the example (see Section 17.2), suppose the inflation rate of gasoline is uncertain. Instead of providing a single numerical value, you could define the inflation rate as a list:



Using the new *Inflation* variable in the definition for *Gasprice*:

*Gasprice*:

`Dynamic(1.2, Gasprice[Time - 1] * (1 + Inflation))` →

	1990	1991	1992	1993	1994
0.05	1.2	1.26	1.323	1.389	1.459
0.1	1.2	1.32	1.452	1.597	1.757
0.15	1.2	1.38	1.587	1.825	2.099

## 17.6 Dependencies with Dynamic

All variables with dynamic inputs are evaluated dynamically, that is, their results are arrays indexed by *Time*.

**Example:** A series of dynamic definitions produce equations for distance, velocity, and acceleration:

*Acceleration*:  $-9.8$



*Dt:* 0.5

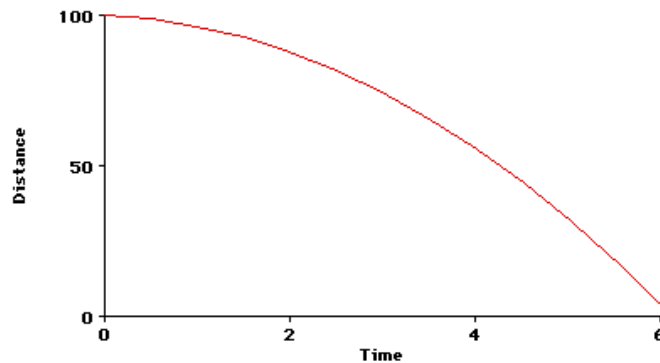
*Time:* Sequence(0, 6, Dt)

*Velocity:*

Dynamic(0, Self[Time-1] + Acceleration \* Dt/2)

*Distance:*

Dynamic(100, Self[Time-1] + Velocity \* Dt) →



## Dynamic dependency arrows

If a variable is dynamically dependent on another variable, a gray or dashed arrow is drawn between the variables.

To show or hide dynamic dependency arrows:

1. Select **Set Diagram Style...** from the **Diagram** menu to open the Diagram Style dialog box (see Section 6.5).
2. Click in the **Dynamic** checkbox to show dynamic arrows (or un-check it to hide the arrows).
3. Click **OK** to accept the change.

## 17.7 Uncertainty and Dynamic

Uncertain variables propagate uncertainty samples during dynamic simulation. If an uncertain variable is used in a dynamic simulation, its uncertainty sample is calculated only once, in the initial time period.

**Example:** The following definitions model population changes over time:

*Population:* `Normal(30, 2)`

*Birthrate:* `Normal(1.2, .3)`

*Pop\_by\_year:* `Dynamic(Population, Self[Time-1] + Birthrate)`

The uncertainty samples for *Population* and *Birthrate* are each calculated once, at the initial time period. The same samples are then used for each subsequent time period.

### Resampling

If you want to create a new uncertainty sample for each time period (that is, resample for each time period), place the distribution in the last parameter of the `Dynamic()` function. For example, replace *Birthrate* with its definition in *Pop\_by\_year*:

*Pop\_by\_year:* `Dynamic(Population, Self[Time - 1] + Normal(1.2, .3))`

An alternative way to create a new uncertainty sample for each time period is to make *Birthrate* a dynamic variable.

*Birthrate:* `Dynamic(Normal(1.2, .3), Normal(1.2, .3))`

*Pop\_by\_year:* `Dynamic(Population, Self[Time-1] + Birthrate)`



---

# 18

## Importing and Exporting Data

---

This chapter describes how to exchange data between Analytica and other applications. The primary methods are:

- Using the standard Macintosh **Copy** and **Paste** commands
- Using the **Import** and **Export** commands
- Using the Macintosh Publish and Subscribe facilities

In addition, you can exchange Analytica models with other modelers using electronic mail.

### 18.1 Copying and pasting

You can use the standard Macintosh **Copy** and **Paste** commands with any modifiable attribute of a variable, module, or function.

#### Pasting data from a spreadsheet

To paste tabular data from a spreadsheet into an Analytica table:

1. Select a group of cells in a spreadsheet.
2. Select **Copy** from that program's **Edit** menu, to copy the data to the clipboard.
3. Bring the Analytica model to the front and open the edit table window you want to paste the data into.
4. Select a top-left cell or the same number of cells that you originally copied.
5. Select **Paste** from the **Edit** menu (**⌘-V**).

**Pasting data from another program**

To paste data from a program other than a spreadsheet:

- Use tabs to separate items, and returns to separate lines.
- Use numbers in floating point or exponential format. You can use the suffixes that Analytica recognizes (including K, M, and m; see page 10-2 for a comprehensive list). Dollar signs (\$) and comma-thousands separators are not permitted.

**Copying a diagram**

To copy an influence diagram, including the objects represented by the nodes:

1. Select the group of nodes you wish to copy.
2. Select **Copy** from the **Edit** menu (⌘-C). The objects that the nodes represent, as well as a picture of the selected nodes with all of the relevant arrows between the selected nodes, are copied to the clipboard.

To copy an entire influence diagram window, select **Copy Diagram** from the **Edit** menu. The entire influence diagram is copied as a PICT representation without copying the objects that the nodes represent.

**Copying an object window**

To copy an object window:

1. Open the object window.
2. Select **Copy Object** from the **Edit** menu. The entire window is copied as a PICT representation.

**Copying an edit table or result table**

To copy data from an edit table or result table into tab-delimited text format (see Section 18.6):

1. Open the window containing the table.
2. Select cells and choose **Copy** from the **Edit** menu (⌘-C).

To copy the index elements in addition to all the elements of a table, select **Copy Table** from the **Edit** menu. The entire

multidimensional array is copied as a graphic and as a list of two-dimensional tables in a text format.

### Copying a result graph

To copy or export a result graph:

1. Open the Result window containing the graph.
2. Select **Copy** from the **Edit** (⌘-C) menu to copy a PICT representation of the graph.

## 18.2 Importing and exporting

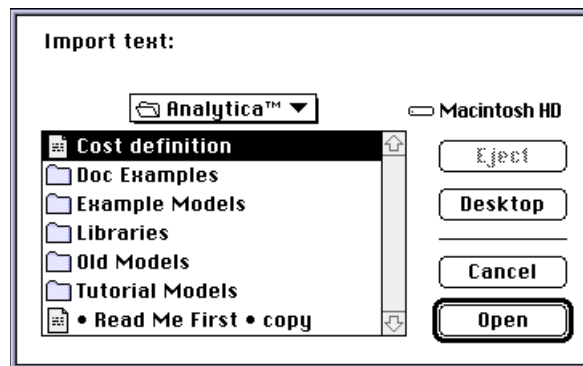
### Importing a definition

To import a definition from a text file into expression format:

1. Select the definition field of the variable in either the Object window or attribute panel definition view.

If the variable is defined as a List, List of Labels, or edit table, select the cell(s) in which to import.

2. Select **Import...** from the **File** menu. A dialog box prompts you for the file name from which to import.



To import into an edit table, the data must be in a tab-delimited text file (see Section 18.6). The indexes of the table must have been previously created as nodes.

**Exporting** To export a variable's definition or result table to a text file, first be certain that the text file is closed.

1. Select the variable to be exported from and open either the Object window, definition in the attribute panel or Result window.
2. Select the definition field, list cell(s), or table cell(s) for exporting.
3. Select **Export** from the **File** menu. A dialog box prompts you for the file name to export to.

Use the **Export** command to export a graph as a PICT file.

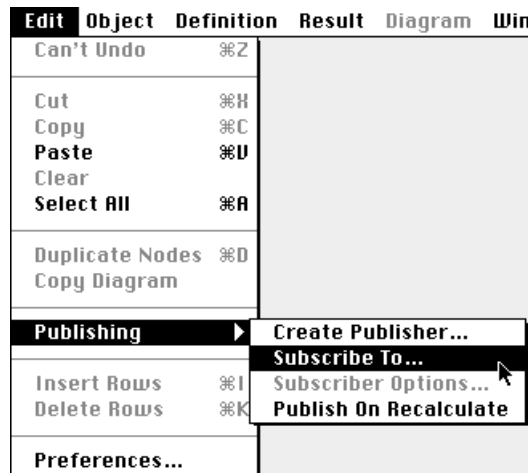
## 18.3 Publish and Subscribe

Use the **Publishing** submenu commands in the **Edit** menu to transfer input tables (by subscribing) and results (by publishing) between Analytica and other applications. These commands keep data up to date across applications.

Publish and Subscribe is a file-based data sharing mechanism on the Macintosh. The publishing application creates and updates a file containing the data that the subscribing application reads.

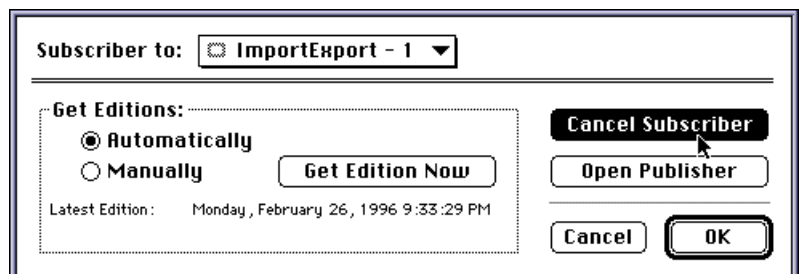
**Subscribing** To subscribe to data for a modifiable attribute, Edit mode must be selected.

1. Select the field, list, or edit table in an Object window or attribute panel. For the definition attribute, Expression must be selected in the expression popup menu.
2. Select **Subscribe to** from the **Publishing** submenu in the **Edit** menu.



**Unsubscribing** You can unsubscribe a previously subscribed attribute from either Edit or Browse mode.

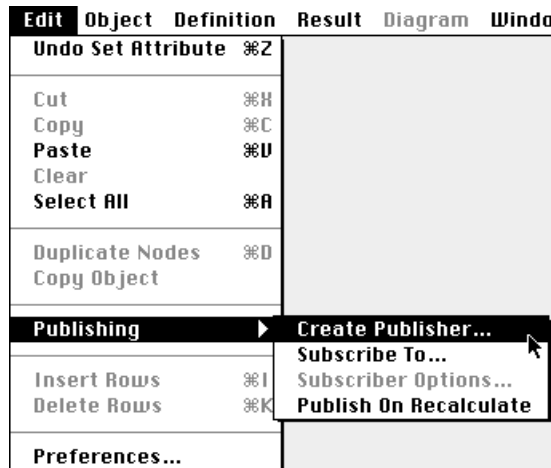
1. Select the field, list, or Edit Table.
2. Select **Subscriber Options** from the **Edit** menu.
3. In the dialog box, select the **Cancel Subscriber** button.





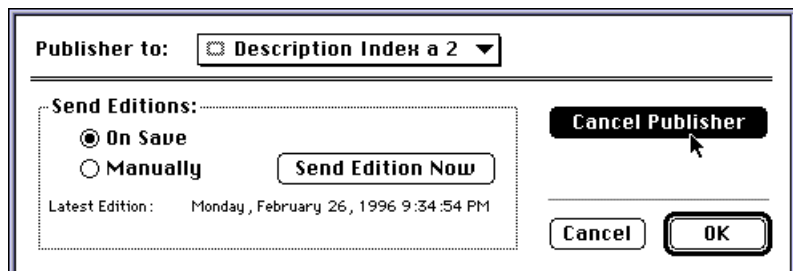
**Publishing** To publish a result window's table data or any modifiable attribute, Edit mode must be selected.


1. Open the result window or select the attribute's field.
2. Select **Create Publisher** from the **Publishing** submenu in the **Edit** menu.



**Unpublishing** You can unpublish a previously published result window or attribute from either Edit or Browse mode.

1. Open the result window or select the attribute's field.
2. Select **Publisher Options** from the **Edit** menu.
3. In the dialog box, select the **Cancel Publisher** button.



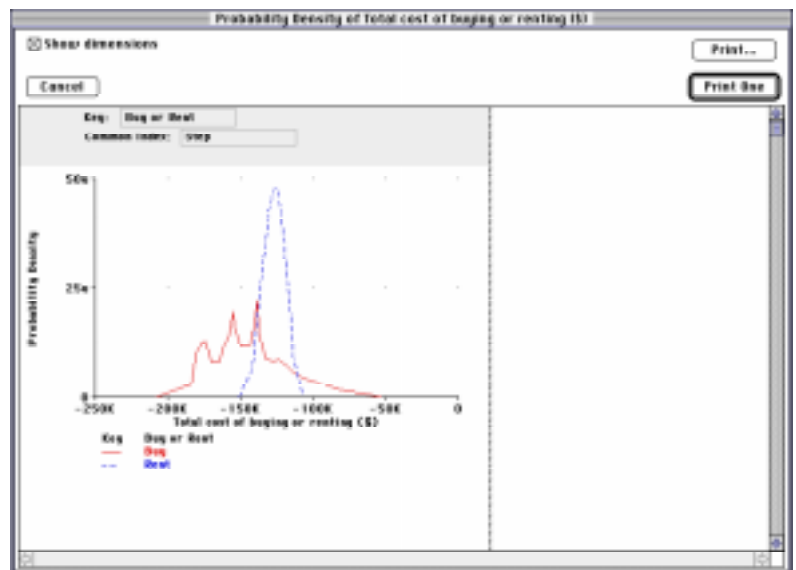
**Note:**  If you publish a table to another application, you must unpublish it before changing the dimensions of the table.

**Publish on Recalculate** Check this menu command to re-publish a published result whenever it is recalculated.

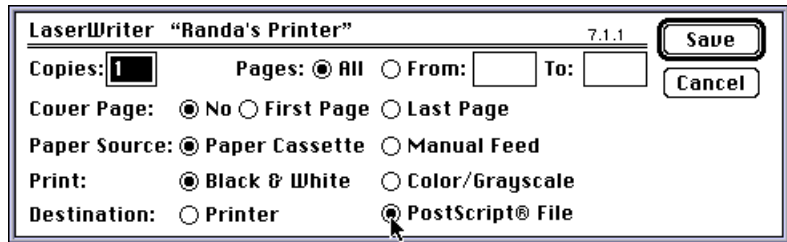
## 18.4 Printing to a file

Another way of exporting any diagram window, object window, or result window to a file is to print to a file:

1. Select **Print** from the **File** menu.
2. Select **Print** (not **Print One**) to display the printer dialog box.



3. Select the **File** radio button and press *enter*.



4. Enter the name of the file and the format for the file.

## 18.5 Sending a model via electronic mail

To send an Analytica model file via electronic mail, do any of the following:

- Include the model file as an attachment to a message.
- Transfer the model file as a binary file via FTP.
- If you cannot send attachments or use FTP, send the model file as text:
  1. Open the model file in a word processor, select the entire file, and copy it.
  2. Create an electronic mail message and paste the file into the message.

The recipient must then:

3. Save the file to the hard disk.
4. Open Analytica from the Finder and select **Open Model** from the **File** menu to open the model.

Note that using this method, the model resources, such as pictures pasted into the model and page setup information, will be lost.

## 18.6 Edit table data import/export format

Data being imported or copied into an edit table must be in a text file with the format described in this section. This is also the format in which an edit table or result table is exported.

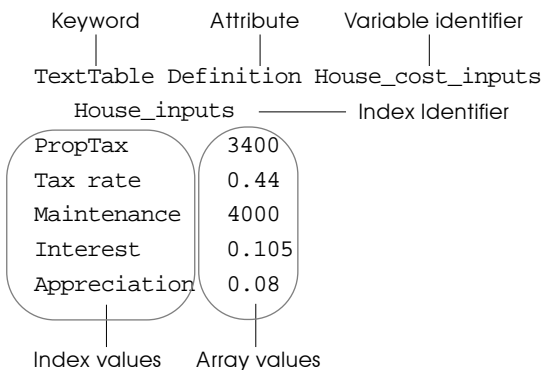
- `TextTable` is a keyword.
- `Attribute` is the name of the attribute into which the data is to be pasted (usually `Definition`).
- `Variable identifier` is the identifier of the variable node into which the data is to be pasted.
- `Index identifier` is the identifier of the index for this variable. This node must already exist in the model.
- Each index value and array value pair must be separated by tabs.

### One-dimensional array

The format for a one-dimensional array is:

```
TextTable <Attribute> <Variable identifier>
<line break>
<tab><Index identifier><line break>
<Index value><tab><Array value><line break>
```

#### Example:

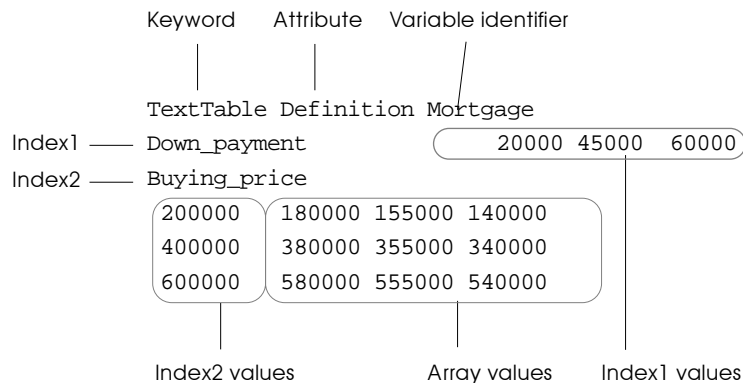


## Two-dimensional array

The format for a two-dimensional array is:

```
TextTable <Attribute> <Variable identifier>
<line break>
<Index1 identifier>< tab><Index1 values separated
by tabs><line break>
<Index2 identifier><line break>
<Index2 value1><tab><Array values separated by
tabs><line break>
<Index2 value2><tab><Array values separated by
tabs><line break>
<Index2 valueN><tab><Array values separated by
tabs><line break>
```

### Example:



## Three-dimensional array

The format for a three-dimensional array is:

```
TextTable <Attribute> <Variable identifier> <line
break>
<Index1 identifier><tab><Index1 Value1><line break>
<Index2 identifier><tab><Index2 values separated by
tabs><line break>
<Index3 identifier><line break>
<Index3 value1><tab><Array values separated by
tabs><line break>
<Index3 value2><tab><Array values separated by
tabs><line break>
<Index3 valueN><tab><Array values separated by
tabs><line break>
<Index1 identifier><tab><Index1 Value2><line break>
<Index2 identifier><tab><Index2 values separated by
tabs><line break>
<Index3 identifier><line break>
<Index3 value1><tab><Array values separated by
tabs><line break>
<Index3 value2><tab><Array values separated by
tabs><line break>
<Index3 valueN><tab><Array values separated by
tabs><line break>
```

and so on for each value of Index1.

**Example:**

	Keyword	Attribute	Variable identifier	
	TextTable	Definition	Net_diff	
Index1	Buying_price	200000		Index1 Value1
Index2	Years_owned	5 10 15		Index2 values
Index3	Down_payment			
Index3 values		20000 45000 60000	10112 12160 13525 10093 12158 13540 10073 12157 13555	Array values
Index1	Buying_price	400000		Index1 Value2
	Years_owned	5 10 15		
	Down_payment			
		20000 10180.	14201. 16867.	
		45000 10160.	14199. 16882.	
		65000 10141.	14198. 16897.	
Index1	Buying_price	60000		Index1 Value3
	Years_owned	5 10 15		
	Down_payment			
		20000 10248	16242 20209	
		45000 10228	16241 20224	
		60000 10208	16239 20239	

**Number Format** Numerical data can be imported in any format recognized by Analytica (see Section 7.6, “Number Format dialog box”).

Numerical data will be exported in the format set for the table, with two exceptions:

- Suffix format numbers will be exported in scientific exponential format.
- Fixed decimal point numbers of more than 9 digits will be exported in scientific exponential format.

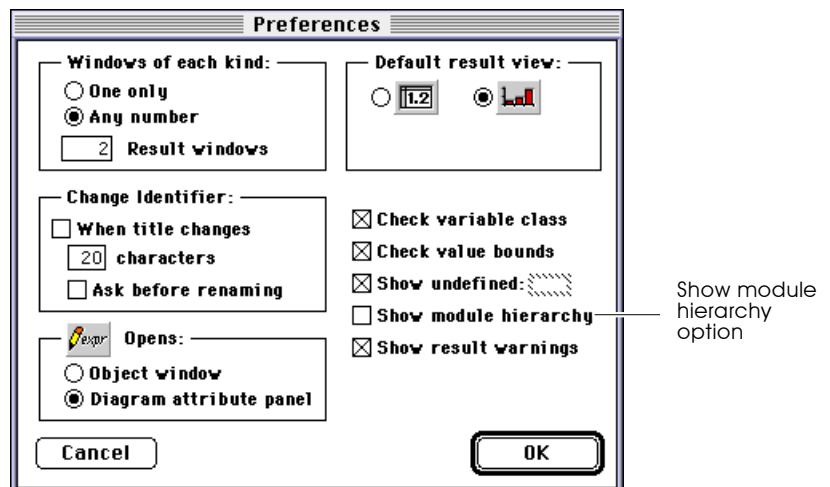
# 19

## Working with Large Models

Large models, which include many variables organized into multiple modules at several levels of hierarchy, can be challenging to find your way around. The first part of this chapter describes how to navigate larger models, using the hierarchy preference, the Outline window, and variable input and output attributes. The second part of this chapter describes how to combine existing models into an integrated model.

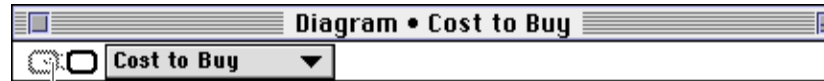
### 19.1 Show module hierarchy preference

Often a large model has many layers of hierarchy. You can see the hierarchy depth of each module at the top of its Diagram window by setting a preference. Select **Preferences...** from the **Edit** menu to display the Preferences dialog box.





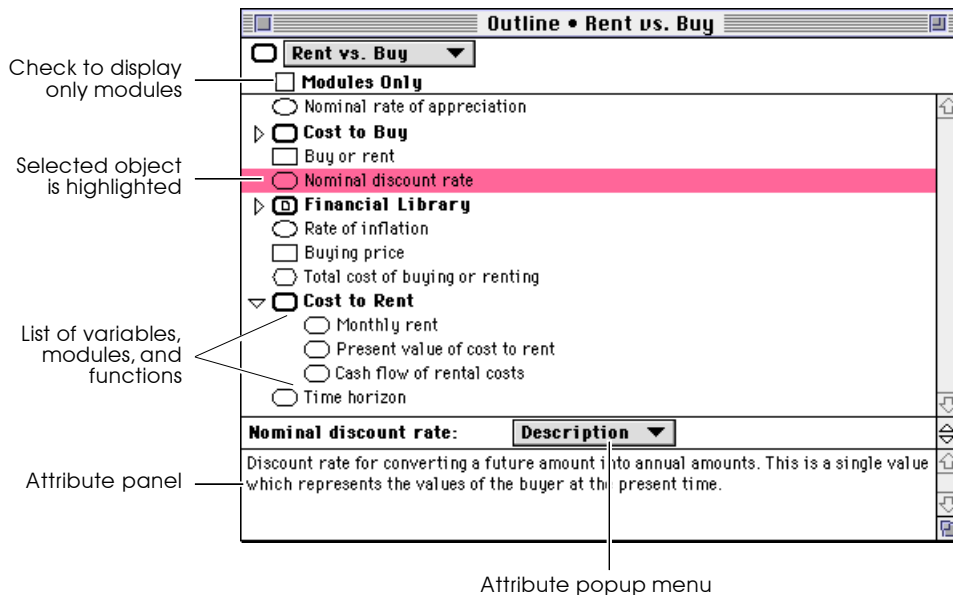
If you check the **Show module hierarchy** box, the top of the active Diagram window displays one or more module node shapes to indicate its hierarchy depth.




Indicates that this module has a parent in the model

## 19.2 The Outline window

The *Outline window* displays a listing of the nodes inside a model. It can also show the module hierarchy as an indented list of modules. It provides an easy way to orient yourself in, and to navigate, a large model.



## Opening the Outline window

To open the Outline window, click on the Outline button in the tool palette ()

The Outline window highlights the entry for the selected module or variable.

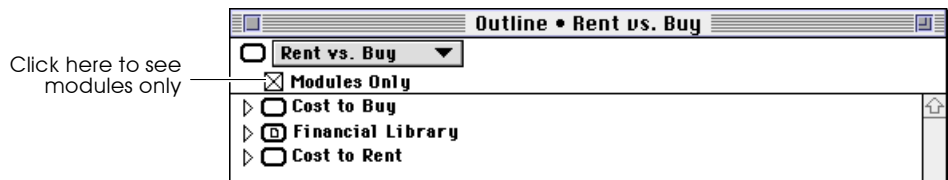
## Opening details from an outline



To display a module's Diagram window, double-click on its entry in the outline.



To display a variable's Object window, double-click on its entry in the outline.

## Expanding and contracting the outline

By default, the outline lists all nodes in the model. Check the **Modules Only** box to list only the modules (exclude variables and functions).



In the outline, each module entry has a triangle icon ( or ) to let you display or hide the module's contents. It behaves much like the triangle icon in list views in the Macintosh Finder under System 7.

-  • Indicates that the module's contents are *not* shown in the Outline window. Click on this icon to display the module's contents.
-  • Indicates that the module's contents are shown as an indented list. Click on this icon to hide the module's contents.

## Viewing and editing attributes

The Attribute panel at the bottom of the Outline window works just like the Attribute panel available at the bottom of a Diagram window (see page 1-14).

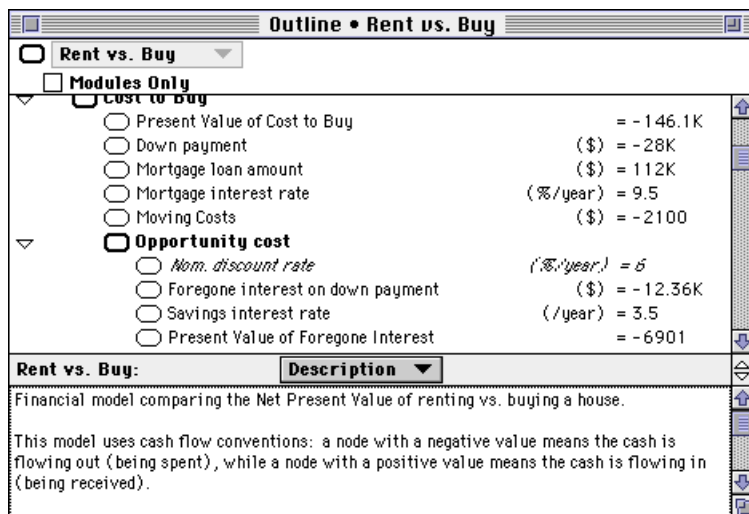
To view the attributes of a listed node:

1. Select the node by clicking on it.
2. Choose the attribute to examine from the Attribute popup menu (see page 1-15).

If you edit attributes in this panel, the changes are propagated to any other attribute panels and Object windows.

## Viewing values

To see the Outline window with mid values, select **Show With Values** from the **Object** menu. Each variable whose mid value has been evaluated and is a scalar will display in the window (see Section 1.9, “Showing mid values”).

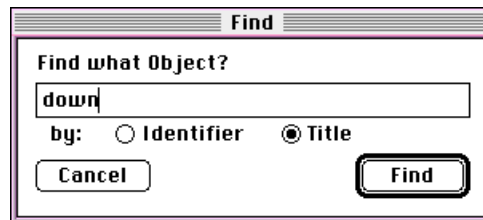


## 19.3 Finding variables

To locate a variable in its diagram, by identifier or by title, use the Find dialog box.

**Find dialog box** To display the Find dialog box:

1. Select **Find...** (⌘-F) from the **Object** menu.



2. Choose the attribute to search by: **Identifier** or **Title**.
3. In the text field, enter the identifier or title for the Analytica object for which you want to search. You can enter an incomplete identifier or title, such as “down” for “Down payment.”
4. Click on the **Find** button to initiate the search.

The Diagram window containing the object found is displayed, with the node of the object selected.

If the name you type does not match completely any existing identifier or title (depending on which attribute you are searching), the first identifier or title that is a partial match will be displayed.

To find the next object that is a partial match to the last identifier or title that you entered, select **Find Next** (⌘-G) from the **Object** menu.

To find an object whose identifier matches the selected text in an attribute field (such as a definition field), select **Find Selection** (⌘-H) from the **Object** menu.

## 19.4 Managing attributes

Every node in an Analytica model is described by a collection of *attributes*. For some models, you may want to control the display of attributes or create new attributes. Some attributes apply to all classes (variable, module, and function). Others apply to specific classes, as listed in the following table.

Attribute	Function	Module	Variable
Author		*	
Check	√		√
Class	*	*	*
<i>Created</i>		*	
Definition	*		*
Description	*	*	*
Domain			√
<i>File Info</i>		*	
Help	√	√	√
Identifier	*	*	*
<i>Inputs</i>	√		√
<i>Last Saved</i>		*	
<i>Outputs</i>	√		√
Parameters	*		
<i>Probvalue</i>			√
Title	*	*	*
Units	*		*
<i>Value</i>			√
User-created (up to 5)	√	√	√

Key:

plain = modifiable by user

\* = always displayed

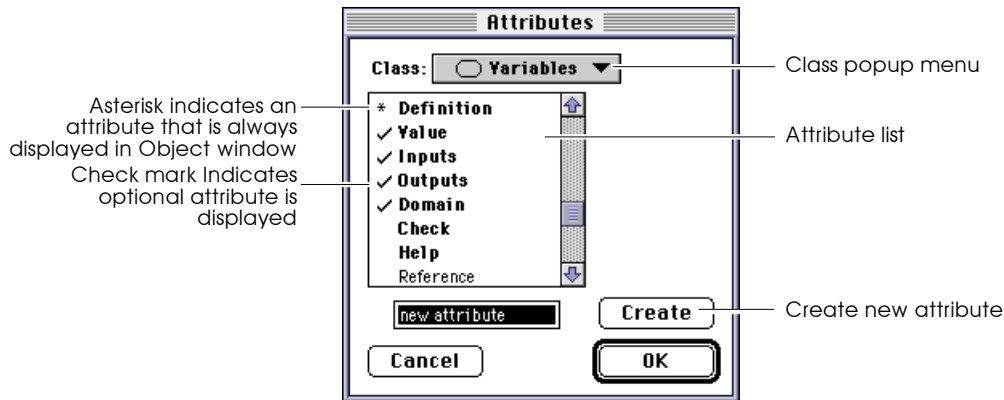
*italic* = set by Analytica

√ = optionally displayed

For descriptions of the attributes, see the Glossary.

**Attributes dialog box** Use the Attributes dialog box to control the display of optional attributes in the Object window and Attribute panel and to define new attributes.

To open the Attributes dialog box, select **Attributes. . .** from the **Object** menu.



### Class popup menu

Use this menu to select the attribute list for variables, modules, or functions.

### Attribute list

This list shows attributes for the selected class. Attributes with an asterisk (\*) are always displayed in the Object window and attribute panel. Attributes with a check mark (✓) are optionally displayed.

## Displaying optional attributes

To display an optional attribute in the Object window and attribute panel, click on it once to select it, then click on it again to show a check mark.

To hide an optional attribute, click on it once to select it, then click on it again to remove the check mark.

## Creating new attributes

You can create up to five additional attributes. For example, you could use a Reference attribute to include the bibliographic reference for a module or variable.

To create a new attribute in the Attributes dialog box:

1. Select **new attribute** from the attribute list to show the new attribute Title field and the **Create** button.
2. Enter the title for the new attribute in the Title field. The title can contain a maximum of 14 characters; 10 characters are the maximum recommended for visibility with all screen fonts.
3. Click on the **Create** button to define the new attribute.

A newly created attribute is displayed for modules, variables, and functions. To control whether or not it is displayed for modules, variables, or functions, select the Class popup menu in the Attributes dialog box, and turn the check mark on or off.

## Renaming an attribute

To rename a created attribute:

1. Select it in the attribute list. The Title field and the **Rename** button appear.
2. Edit the name of the attribute in the Title field.
3. Click on the **Rename** button.

## Referring to the value of an attribute

Analytica includes the following function for referring to the value of an attribute in a variable's definition:

### *Attrib Of Ident*


Returns the value of attribute *Attrib* of the object whose identifier is *Ident*.

The result is a text value for all attributes except Value and Probvalue, which may return a number, text, or array.

**Library:** Special

**Example:**

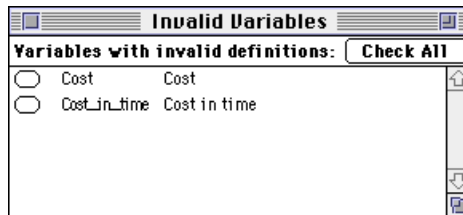
Units of Time → 'Years'


**Note:** 

When the value of an attribute changes, variables whose definitions include this expression are *not* automatically recomputed.

## 19.5 Invalid Variables

To locate all variables in a model with syntactically incorrect or missing definitions, select **Show Invalid Variables** from the **Definition** Menu.



Double-click on a variable to open its Object window. From the Object window, you can edit the definition, or click on the Parent Diagram button () to see the variable in its diagram.

## 19.6 Using filed modules and libraries



Modules and libraries can be components of a model. If you are building several similar models, or if you are building a large model composed of similar components, you can create modules and libraries for reuse. (See Chapter 20 for details about libraries).

To use a module or library in more than one model, create a *filed module* or *filed library*.




## Creating a filed module or library

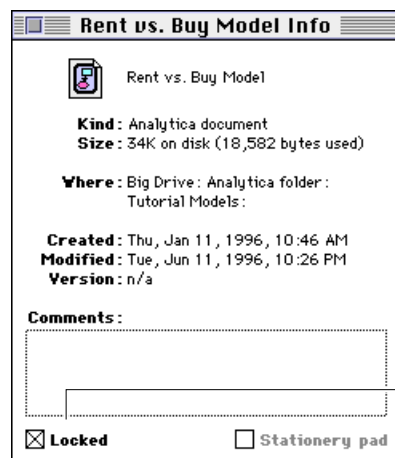
To create a filed module or library:

1. Create a module by dragging the module icon from the node palette onto the diagram, and give it a title.
2. Create functions and/or variables in the module, or create them elsewhere and move them into the module.
3. Change the class of the module to **Module** (  ) or **Library** (  ) (with the “document” icon) (see Section 4.9).
4. The Save As dialog box appears. Give the filed module or library a name and save it.
5. If you want the original model to load the new filed module or library the next time it is opened, save the model using the **Save** command.

## Locking a filed module or library

To prevent a filed module or library from being modified, lock it in the finder:

1. Close the filed module or library, or close Analytica.
2. In the finder, select the filed module or library (single-click).
3. Select **Get Info** ( -I) from the **File** menu to display the Info window for the file.



Check this option to lock a library or module file

4. Check the Locked checkbox.
5. Close the Info window.

### **Adding a module or library to a model**

To add a filed module or library to the active model, use the Add Module dialog box (see Section 19.7). You can either embed a module copy or link to the original of the filed module or library.

### **Removing a module or library from a model**

To remove a filed module or library from a model, first select it. Then, select **Cut** or **Clear** from the **Edit** menu. An embedded copy will be deleted; a linked original will still exist as a separate file.

---

#### **Warning:**

Any definitions that use a function in a deleted library or that have an input from a deleted module or library will have the deleted object removed and will be changed to `FunctionOf(remaining variables)`.

---

### **Saving changes**

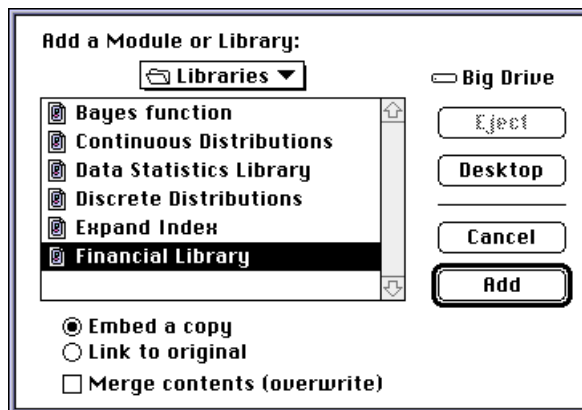
After you have linked to a filed module or library, the **Save** command saves every filed module and library that has changed, as well as the model containing them, in their corresponding files.


The **Save As** and **Save a Copy In** commands save only the active (top most window's) model, filed module or filed library.

## 19.7 Add Module dialog box

Use the Add Module dialog box to add a filed module or library to the active model.

To open the Add Module dialog box, select **Add Module** from the **File** menu (⌘-L).



Note: 

Be sure that the selected model or module was saved with a class of **filed module** or **filed library**. If it was saved with a class of model, when it is linked to the root model, its preferences and uncertainty settings will overwrite the preferences and uncertainty settings of the root model.

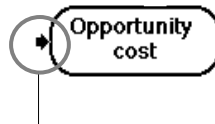
An added module or library must be either embedded or linked. You can optionally overwrite any nodes with the same identifiers.

### Embed a copy

Embeds a copy of the selected module or library in the active model, making it a part of, and saving it with, the model. Any changes to the copy will not affect the original filed module or library.

**Link to original** Creates a link to the selected module or library, which can be separately opened and saved. If you make changes to a linked module or library from one model, the changes are saved in the original's file and any other models linked to the original will be affected by the changes.

A linked module or library has a bold arrow pointing into it on the diagram.



Bold arrow indicates that this is a linked module

**Merge Contents (Overwrite)** Select this checkbox to overwrite existing nodes in the active model with nodes having the same identifiers from the added module or library. This is useful if the file being added contains updates from a previous version.

If you do not select this checkbox and a node in the file being added has an identifier that duplicates one in the active model, a warning message displays. If you proceed, the identifier in the active model is modified, and the identifier in the file being added remains unchanged.

## 19.8 Combining models into an integrated model

Large models introduce a unique set of modeling issues. Modelers may want to work on different parts of a model simultaneously, or at remote locations. During construction, a large model may be more tractable when broken into modular pieces (modules), but all modules should use a common set of indexes and functions. Analytica provides the functionality required to support large-scale, distributed modeling efforts.

This section describes how to best use Analytica for large modeling projects and contains suggestions for planning a large model where responsibility for each module is assigned to different people (or teams).

## Define public variables

The first step to creating an integrated model is to define public variables for use by all modules, and agree on module linkages.

Every integrated model will have variables that are used by two or more projects (e.g., geographical, organizational, or other indexes, modeling parameters, and universal constants). These public variables should be defined in a separate module, and distributed to all project teams. Each team uses **Add Module** (see Section 19.7) to add the public variables module to its model at the outset of modeling. Using a common module for public variables avoids duplication of variables and facilitates the modules' integration.

Source control over the public variables module must be established at the outset so that all teams are always working with the same public variables module. Modelers should not add, delete, or change variables in the public variables module unless they inform the source controller, who can then distribute a new version to all modelers.

If multiple teams will be working on separate projects, it is essential that the teams agree upon inputs and outputs. Modelers must specify the input variables, units, and dimensions that they are expecting as well as the output variables, units, and dimensions that they will be providing. The indexes of these inputs and outputs should be contained in the public variables module.

## Create a modular model

By keeping large pieces of a model in separate, or filed modules, modelers can work on different parts of a model simultaneously. You can break an existing model into modules, or combine modules into an integrated model. In both cases, the result is a top-level model, into which the modules are added.

To save pieces of a large model as a set of filed modules, see Section 19.6.

To combine existing models into a new, integrated model:

1. Create or open the model that will be the top level of the hierarchy. This is the model to which all submodels will be added.
2. Using **Add Module** (see Section 19.7), add in the submodels. Be sure to check the Merge option in the Add Module dialog box. Add the modules in the following sequence:
  - Any public variable modules
  - All remaining modules in order of back to front; that is:  
first, the module(s) whose outputs are not used by any other module, and  
last, the module(s) which take no inputs from any other module.
3. Save the entire integrated model, using the **Save** command.

The two alternative methods of controlling each module's input and output nodes so the modules can be easily integrated are:

- Identical identifiers
- Redundant nodes.

### **Identical identifiers**

Assign the input nodes in each module the exact same identifiers as the output nodes in other modules that will be feeding into them. When you add the modules beginning with the last modules first (i.e., those at the end of model flow diagram), the input nodes will be overwritten by the output nodes, thus linking the modules and avoiding duplication.

With identical identifiers, the individual modules cannot be evaluated alone because they are missing their input data; they can be evaluated only as part of the integrated model.

### **Redundant nodes**

Place the output node identifiers in the definition fields of their respective input nodes. This method requires more memory than using identical identifiers, due to the node redundancy, and is

therefore less desirable when large tables of data are passed between modules. However, since no nodes are overwritten and lost upon integration, this method preserves the modules' structural integrity, with both input and output nodes visible in each module's diagram.

With redundant nodes, each module can be opened and evaluated alone, using stand alone shells.

## Stand alone shells

With redundant nodes, you can create a top-level model that contains one or more modules and the public variables module plus dummy inputs and outputs. Such a top-level model is called a stand alone shell because it allows you to open and evaluate a single module 'standing alone' from the rest of the integrated model. Stand alone shells are useful when modelers want to examine or refine a particular module without the overhead of opening and running the entire model.

To create a stand alone shell for module *Mod1*, which is a filed module:

1. Open the integrated model and evaluate all nodes that feed inputs to *Mod1*.
2. Use the **Export** command (see Section 18.2) to save the value of each feeding node in a separate file. Make a note of:
  - the identifier of each node and the indexes by which its results are dimensioned,
  - the identifiers of *Mod1*'s output nodes, if you want to include their dummies in the stand alone shell.
3. Close the integrated model.
4. Create a new model, to be the stand alone shell.
5. Use **Add Module** to add the public variables module.
6. For each input node, create a node containing an edit table, using the identifier and dimensions of the feeding nodes you noted from the integrated model.

7. Use the **Import** command (see Section 18.2) to load the appropriate data into each node's edit table.
8. Use **Add Module** to add *Mod1* into the stand alone shell.
9. To include output nodes at the top level of the hierarchy, create nodes there and define them as the identifiers of *Mod1*'s outputs.
10. Save the shell.

The shell now has all the components necessary to open and evaluate *Mod1*, without loading the entire model. As long as modelers do not make changes to the dimensions or identifiers of module inputs and outputs, they can modify a module while using the stand alone shell, and the resulting module will be usable within the integrated model.

## Cautions in combining models

### Identifiers

Every object in a model must have a unique identifier. The identifiers of filed libraries and filed modules that you add to a model, as well as their variables and functions, cannot duplicate identifiers in the root model. See “Merge Contents (Overwrite)” on page 19-13.

### Created attributes

When you combine models with created attributes, the maximum number of defined attributes is five (see Section 19.4).

### Location of linked modules and libraries

If the model will eventually be distributed to other computers, all modules and libraries should be on the same drive as the root model prior to being added to the root model. When the model is distributed, distribute it with all linked modules and libraries.



## 19.9 Managing windows

An Analytica model can potentially display thousands of Diagram, Object, and Result windows. To prevent your screen from becoming cluttered, Analytica limits the number of windows of each type that can be open at once. The default limits are:

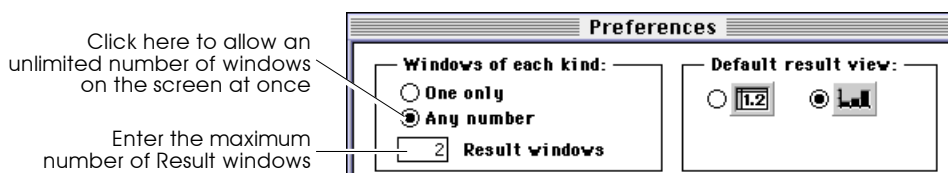
- The top-level Diagram window and not more than one Diagram window for each lower level in the hierarchy
- One Object window
- Two Result windows

The oldest window of the same type is deleted whenever you display a new window that would otherwise exceed these limits.

### Overriding the limits on the number of windows

To display more windows of the same type, override the default limits in one of the following ways:

- Open a second Object window, or open a Diagram window without closing an existing Diagram window at the same level, by pressing the *command* key (⌘) while you click or double-click to open the new window.
- Use the Preferences dialog box (see Section 4.10) to change the limits. Select **Preferences...** from the **Edit** menu. .



In the “Windows of each Kind” area, select **Any Number** instead of **One Only**.

To display more Result windows and keep the limit on Diagram and Object windows, enter the maximum number of Result windows.

---

# 20

## Building Functions and Libraries



---

You can create your own functions to perform calculations you use frequently. A function has one or more parameters; its definition can be an arbitrary expression containing these parameters. To control the evaluation of the parameters, or to check that they are of the appropriate type (for example, scalar or array, deterministic or probabilistic), you can specify various qualifiers.

A collection of functions in a library can be used by more than one model. Libraries of functions are available for some applications. (They are located in the Libraries folder inside the Analytica folder on your hard disk.) You may want to look at some of these libraries to see if they provide functions that you can use or that you can refer to as a starting point for creating your own functions.

### 20.1 Creating a function

To define a function:

1. Make sure the Edit tool is selected and you can see the node palette (see page 4-3).
2. If you don't see the Function node icon () , scroll the node palette by clicking on the down arrow ().
3. Drag the Function node icon from the node palette into the Diagram area.
4. Title the node, and double-click on it to open its Object window.

5. Enter the new function's attributes (described in the next section).

## 20.2 Attributes of a function

A function has attributes in common with other nodes, such as identifier, title, description, and definition, and a unique attribute, parameters.

**Identifier** If you are creating a library of functions, make a descriptive identifier. This identifier appears in the function list for the library under the **Definition** menu, and is used to call the function. Analytica makes all characters after the first lower case.

**Title** If you are creating a library of functions, limit the title to 22 characters. This title appears in the Object Finder dialog box to the right of the function.

**Units** If desired, use the units field to document the units of the function's result. The units are not used in any calculation.

**Parameters** The parameters to be passed to the function must be enclosed in parentheses, separated by commas. For example:  $(x, y, z)$

The parameters may have type qualifiers (see the next section).

If you are creating a library of functions, use descriptive abbreviations for the parameters and give them a logical sequence. The parameters will appear in the Object Finder dialog box and they will be pasted when the function is pasted from its library in the **Definition** menu.

**Description** If you are creating a library of functions, enter an explanatory description. This text appears in a scrolling box in the bottom half

of the Object Finder dialog. The first 256 characters appear in the balloon help for the function's node, in the diagram window.

**Definition** The definition of a function is an expression that includes all of its parameters. When you select the definition field of a function, the Inputs popup menu lists the parameters.

## 20.3 Parameter qualifiers

The expressions passed into a function are evaluated, by default, according to their context, that is, deterministically or probabilistically. You can control the evaluation of a function more precisely by specifying parameter qualifiers.

**Function parameter type qualifiers** You can specify one of the following types for each parameter of a function:

**ArrayType** An array of one or more dimensions.

**Ascending** A list or one-dimensional array of increasing numeric values.

**Context, Expr** Default qualifier; evaluates according to the surrounding context.

**DetermType** Forces deterministic evaluation.

**IndexType** A variable defined as a list or list of labels, or an array indexed by self.

**Numeric** A number or an array of numbers.

**Positive** A single positive number.

**Prob** Forces probabilistic evaluation.

**Samp** Forces probabilistic evaluation and returns an error message if the result is not indexed by *Run*.

**Scalar** A single number.

**Vector** A literal list, or a variable defined as a list or one-dimensional array.

**Syntax** `(Parameter1: Qualifier1; Parameter2: Qualifier2)`

**Examples:**

The default qualifier type is `Context`, or `Expr` (the two are equivalent). A parameter list, `x` and `y`, using the default qualifier looks like this:

`(x, y)`

- To apply the `Prob` parameter type qualifier to `x`:

`(x: Prob; y)`

- To apply `Prob` to both `x` and `y`:

`(x, y: Prob)`

- To apply `Prob` to only `y`:

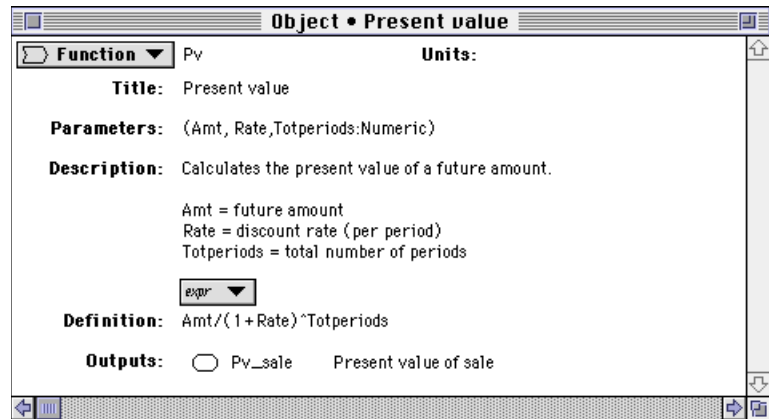
`(x; y: Prob)`

- To apply `IndexType` to `x` and `Prob` to `y`:

`(x: IndexType; y: Prob)`

## 20.4 Example function

The following function,  $Pv()$ , computes the present value of a future amount.



**Parameters** The three parameters, *Amt*, *Rate*, and *Totperiods*, are qualified to be numeric.

**Definition** The definition is an expression using *Amt*, *Rate*, and *Totperiods*.



**Sample usage** A variable uses the  $Pv()$  function similar to the built-in functions in Analytica. For example, if *Discount* is defined as the annual discount rate, *Sales\_price* is defined as a future value, and *Years* is defined as the number of years until selling, then

*Present Value*:  $Pv(\text{Sales\_price}, \text{Discount}, \text{Years})$

## 20.5 Libraries

When you place functions and variables in a library, the library, with its functions and variables, is available as an extension to the system libraries. Up to eight user libraries can be used in a model.

There are two types of user libraries (see also Section 4.9):

- A library (  ) is a module within the current model.
- A filed library (  ) is saved in a separate file, and can be shared among several models.

### Creating a library

To create a library of functions and/or variables:

1. Create a module by dragging the module icon from the node palette onto the diagram, and give it a title.
2. Change the class of the module to library or filed library (see Section 4.9).
3. Create functions and/or variables in the new library or create them elsewhere in the model and then move them into the library.

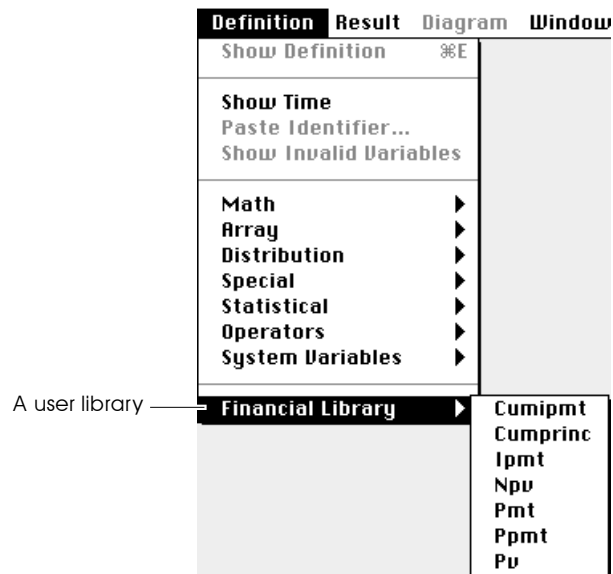
Functions and variables in the top level of the library can be accessed from the **Definition** menu or Object Finder. Use modules within the library to hold functions and variables (such as test cases) that will not be accessible to models using the library.

### Adding a filed library to a model

Add a filed library to a model using the Add Module dialog box (see Section 19.7).

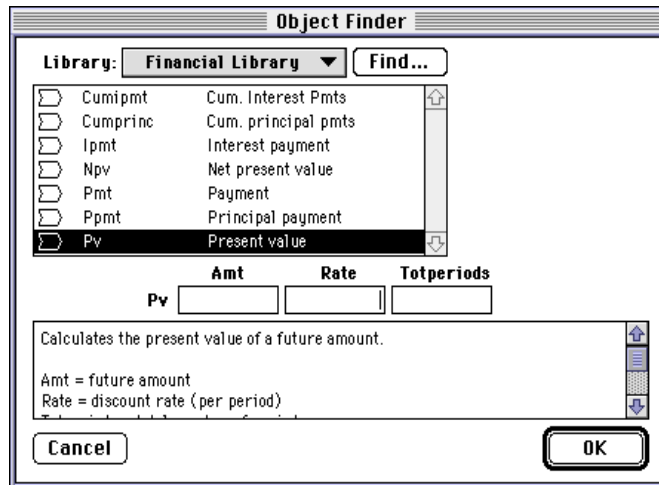
**Using a library** When defining a variable, you can use a function or variable from a library in any of the following ways:

- Type it in.
- Select **Paste Identifier** from the **Definition** menu to open the Object Finder.
- Select **Other** from the Expression popup menu to open the Object Finder.
- Paste from the library under the **Definition** menu.





**Example** Compare the  $PV()$  function's appearance in the Object window (page 20-5) to its appearance in the Object Finder:



# Menus

 **File Edit Object Definition Result Diagram Window**

## A.1 File menu

The **File** menu contains commands to open, create, and save files, as well as to import and export them. In addition, the **Page Setup**, **Print**, and **Quit** commands are in this menu.

File	
New Model	⌘N
Open Model...	⌘O
Add Module...	⌘L
Close	⌘W
Close Model	
Save	⌘S
Save As...	
Save A Copy In...	
Import...	
Export...	
Page Setup...	
Print...	⌘P
Print Report...	
Quit	⌘Q

**New Model** Starts a new model.

**Open Model** Opens an existing, previously saved model.

**Add Module** Adds a filed module or filed library to the active model.

<b>Close</b>	Closes the active window.
<b>Close Model</b>	Closes the active model.
<b>Save</b>	Saves the active model to a file, and saves each changed linked module and linked library to its own file. If the model has never been saved before, prompts for a file name and folder.
<b>Save As</b>	Saves the active model, filed module, or filed library as a new file. Prompts for a file name and folder.
<b>Save A Copy In</b>	Saves a copy of the active model, filed module, or filed library into a new file, leaving the active file name for future saves. Prompts for a file name and folder.
<b>Import</b>	Imports the contents of a text or data file into the selected variable definition. See Section 18.2, “Importing and exporting”.
<b>Export</b>	Exports the contents of the selected field or cells into a file. See Section 18.2, “Importing and exporting”.
<b>Page Setup</b>	Displays a dialog box for selecting paper size, orientation, and other printing options.
<b>Print</b>	Displays a dialog box for selecting the number of copies you want to print and other printing options.
<b>Print Report</b>	Displays a dialog box for printing multiple diagrams, object windows, and result windows at the same time. See Section 1.10, “Printing”.
<b>Quit</b>	Quits the Analytica application. Confirms if you wish to save changes to the current model.

## A.2 Edit menu

The **Edit** menu contains commands to manipulate objects, text or graphics, and display the Preferences dialog.

<b>Edit</b>	
<b>Can't Undo</b>	⌘Z
<b>Cut</b>	⌘H
<b>Copy</b>	⌘C
<b>Paste</b>	⌘U
<b>Clear</b>	
<b>Select All</b>	⌘A
<b>Duplicate Nodes</b>	⌘D
<b>Copy Object</b>	
<b>Publishing</b>	▶
<b>Insert Rows</b>	⌘I
<b>Delete Rows</b>	⌘K
<b>Preferences...</b>	

**Undo** Undoes your last action.

**Cut** Cuts the selected text, nodes, graph, or table cells into the clipboard temporarily for pasting.

**Copy** Copies the selected text, nodes, graph, or table cells into the clipboard temporarily for pasting.

**Paste** Pastes the contents of the clipboard at the insertion point or replaces the current selection.

**Clear** Deletes the selected text or node(s).

**Select All** Selects all text, nodes, or table cells.

**Duplicate Nodes** Duplicates the selected nodes. See “Duplicating nodes” on page 4-6.

- Copy Diagram/Table/Object** When a Result table or Edit Table is active, this command is **Copy Table** which copies the entire multidimensional value as a tab-delimited list of tables. When a Diagram window is active, this menu command is **Copy Diagram**, which copies a picture of the diagram without copying the objects they represent. When an object window is active, this command is **Copy Object**, which copies the object window.
- Publishing** Publish and subscribe options. See Section 18.3, “Publish and Subscribe”.
- Insert Rows** Inserts an item in a list, or a row in a table, by copying the current item, or row. If a column in a table is selected, this command changes to **Insert Columns**.
- Delete Rows** Deletes the selected item or items in a list, or rows in a table. If a column in a table is selected, this command changes to **Delete Columns**.
- Preferences** Displays the Preferences dialog box to examine or change various options. See Section 4.10, “Preferences dialog box”.

## A.3 Object menu

The **Object** menu contains commands that find and create Analytica objects and display their attributes.

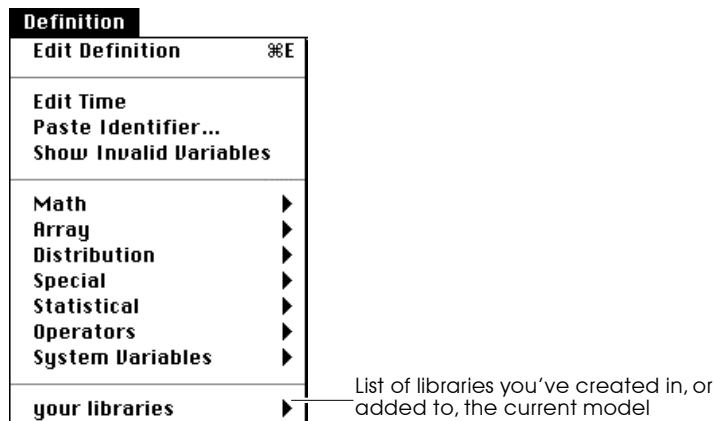
Object	
Find...	⌘F
Find Next	⌘G
Find Selection	⌘H
Make Alias	⌘M
Make Importance	
Make Input Node	
Make Output Node	
Show By Identifier	⌘Y
Show With Values	
Attributes...	

---

<b>Find</b>	Displays a dialog box to find an object by its identifier or title. See Section 19.3, “Finding variables”.
<b>Find Next</b>	Finds the next object that partially matches the previously defined text string.
<b>Find Selection</b>	Finds an object by its identifier that matches the currently selected text.
<b>Make Alias</b>	Creates an alias for the selected object. See Section 4.6, “Creating alias nodes”.
<b>Make Importance</b>	Creates an Index and General variable for the selected variable to compute the uncertainty importance (rank correlation) contributions of its inputs. See Section 16.5, “Importance analysis”.
<b>Make Input Node</b>	Creates an input node as an alias of the selected object. See Section 9.1, “Using input nodes”.
<b>Make Output Node</b>	Creates an output node as an alias of the selected object. See Section 9.3, “Using output nodes”.
<b>Show By Identifier</b>	Shows variables by their identifier in the current diagram, Edit Table, Result window, or Outline view.
<b>Show With Values</b>	Shows the mid values of the variable and all its inputs. See Section 1.9, “Showing mid values”, and Section 19.2, “The Outline window”.
<b>Attributes</b>	Opens the Attribute dialog box to set the visibility of attributes and define new attributes. See Section 19.4, “Managing attributes”.

## A.4 Definition menu

The **Definition** menu contains commands for editing variable definitions. It lists Analytica's built-in function libraries, as well as any user libraries that are currently open.



**Edit Definition** Opens the appropriate view for editing the definition of the selected variable. If the variable is defined as a distribution or sequence, the Object Finder opens. If it is defined as a table or probability table, its Edit Table window opens. Otherwise, an Object window or Attribute panel opens, depending on the Edit attributes setting in the Preferences dialog box. See Section 4.10, “Preferences dialog box”.

**Edit Time** Opens the Object window for the *Time* system variable. See Section 17.1, “The Time index”.

**Paste Identifier** Opens the Object Finder dialog box for examining functions and variable identifiers, entering function parameters, and pasting them into definitions. See Section 8.4, “Object Finder dialog box”.

**Show Invalid Variables** Displays a window listing all variables with invalid or missing definitions. See Section 19.5, “Invalid Variables”.

**Math** Displays a list of the mathematical functions in the Math library. See Section 10.5, “Math functions”.

<code>Abs()</code>	<code>Logten()</code>
<code>Arctan()</code>	<code>Round()</code>
<code>Cos()</code>	<code>Sin()</code>
<code>Exp()</code>	<code>Sqr()</code>
<code>Factorial()</code>	<code>Sqrt()</code>
<code>Ln()</code>	

**Array** Displays a list of functions for creating and transforming arrays in the Array library. See Chapter 12, “Array Function Reference”.

<code>Area()</code>	<code>Product()</code>
<code>Array()</code>	<code>Rank()</code>
<code>Average()</code>	<code>Sequence()</code>
<code>Choice()</code>	<code>Size()</code>
<code>Concat()</code>	<code>Slice()</code>
<code>Cumproduct()</code>	<code>Sortindex()</code>
<code>Cumulate()</code>	<code>Subscript()</code>
<code>Determtable()</code>	<code>Subset()</code>
<code>Integrate()</code>	<code>Sum()</code>
<code>Max()</code>	<code>Table()</code>
<code>Min()</code>	<code>Uncumulate()</code>
<code>Normalize()</code>	

**Distribution** Displays a list of functions for creating probability distributions in the Distribution library. See Chapter 14, “Using Continuous Probability Distributions” and 15, “Using Discrete Probability”.

<code>Bernoulli()</code>	<code>Lognormal()</code>
<code>Beta()</code>	<code>Normal()</code>
<code>Certain()</code>	<code>Probdist()</code>
<code>Chancedist()</code>	<code>Prohtable()</code>
<code>Cumdist()</code>	<code>Triangular()</code>
<code>Fractiles()</code>	<code>Uniform()</code>



**Special** Displays a list of unusual or less commonly used functions in the Special library.

Argmax()	Ident[time-...]
Attrib of ...	Invert()
Cubicinterp()	Linearinterp()
Decompose()	Stepinterp()
Determinant()	Subindex()
Dydx()	Transpose()
Dynamic()	Truncate()
Elasticity()	Using...Do
For...Do	WhatIf()
Ident[I=...]	

**Statistical** Displays a list of statistical functions in the Statistical library. See Section 16.1, “Statistical functions”.

Correlation()	Probbands()
Frequency()	Rankcorrel()
Getfract()	Sample()
Kurtosis()	Sdeviation()
Mean()	Skewness()
Mid()	Statistics()
Probability()	Variance()

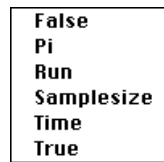
**Operators** Displays a list of arithmetic, comparison, logical, and conditional operators in the Operators library. See “Operators” on page 10-4.

( - )	<>
+	≠
-	>=
*	≥
/	>
÷	Not
^	Or
<	And
≤	If Then Else
<=	IfAll Then Else
=	

**System Variables** Displays a list of system variables that you can use in definitions (see the next section).

***your libraries*** Any libraries that you have defined or added to the model are listed at the bottom of the **Definition** menu, each with a submenu that lists the functions contained in the library.

**System variables submenu** The system variables submenu lists the Analytica variables and constants that can be used in variable definitions.



**False** The logical (Boolean) constant that evaluates numerically to zero.

**Pi** The ratio of circumference to the diameter of a circle.

**Run** The index for uncertainty sampling, defined as `Sequence(1, Samplesize)`.

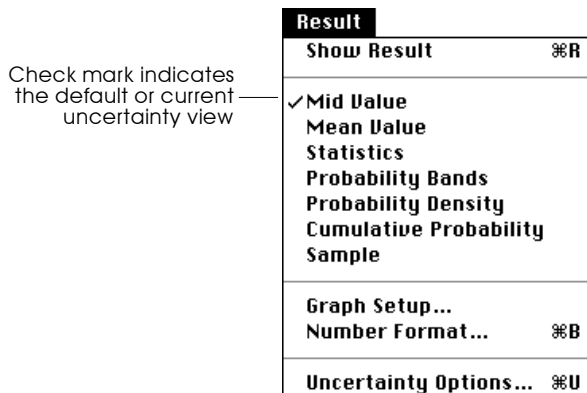
**Samplesize** The number of sample iterations for probabilistic simulation. See Section 13.6, "Uncertainty Setup dialog box".

**Time** The index variable identifying the dimension for dynamic simulation (the `Dynamic()` function). See Section 17.1, "The Time index".

**True** The logical (Boolean) constant that evaluates numerically to nonzero.

## A.5 Result menu

The **Result** menu contains commands for opening Result windows, changing the appearance of graphs and tables, and setting uncertainty options.



- Show Result** Opens a Result window for the selected object. See Section 2.1, “The Result window”.
- Mid Value** Displays the deterministic value, holding most uncertain variables to their median value. See Section 2.4, “Uncertainty view options”.
- Mean Value** Displays the mean of the uncertain value. See Section 2.4, “Uncertainty view options”.
- Statistics** Displays the statistics of the uncertain value in a table as set in the Uncertainty Setup dialog box. See Section 2.4, “Uncertainty view options”.
- Probability Bands** Shows probability bands as set in the Uncertainty Setup dialog box. See Section 2.4, “Uncertainty view options”.
- Probability Density** Displays a probability density graph for an uncertain value. For a discrete probability distribution, **Probability Mass** replaces this command. See Section 2.4, “Uncertainty view options”.

- Cumulative Probability** Displays a cumulative probability graph representing the probability that a variable's value is less than or equal to each possible (uncertain) value. See Section 2.4, “Uncertainty view options”.
- Sample** Displays a table of the values determined for each uncertainty sample iteration. See Section 2.4, “Uncertainty view options”.
- Graph Setup** Displays a dialog box to specify the graphing tool, graph frame, and graph style.
- Number Format** Displays a dialog box to set the number format for displays of results. See Section 7.6, “Number Format dialog box”.
- Uncertainty Options** Displays a dialog box to specify the uncertainty sample size and sampling method and to set options for statistics, probability bands, probability density, and cumulative probability. See Section 13.6, “Uncertainty Setup dialog box”.

## A.6 Diagram menu

The **Diagram** menu contains commands for changing the display of the diagram, including nodes, arrows, and fonts.

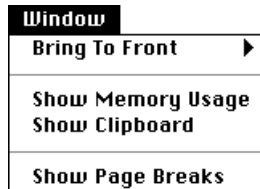


- Set Diagram Style** Displays a dialog box to set default arrow displays, node size, and font. See Section 6.5, “Diagram Style dialog box”.

<b>Set Node Style</b>	Displays a dialog box to set arrow display and font for specific nodes. See Section 6.6, “Node Style dialog box”.
<b>Show Color Palette</b>	Displays a palette to set the color of the diagram background or of selected nodes. See Section 6.4, “Changing background or node colors”.
<b>Align to Grid</b>	Aligns selected node(s) to the diagram grid. See “Aligning to the grid” on page 4-6.
<b>Adjust Size</b>	Adjusts the selected node’s size to match the default node size, or to fit the title label. See “Default node size” on page 6-12.
<b>Move Into Parent</b>	Moves the selected object from the current diagram to its parent diagram. See Section 1.4, “Examining an influence diagram window”.
<b>Resize Centered</b>	If checked, when you resize a node, the node’s center stays unmoved. If unchecked, when you resize a node by dragging a corner handle, the opposite handle stays unmoved. See “Align nodes horizontally or vertically” on page 6-5.
<b>Set Diagram Size</b>	Displays a dialog box to set the number of diagram pages. See Section 6.7, “Changing the size of the diagram”.
<b>Turn Grid Off</b>	Turns alignment to the diagram grid on or off in edit mode. See “Aligning to the grid” on page 4-6.
<b>Edit Icon</b>	Opens a window to edit the icon for the selected node. See Section 9.7, “Adding icons to nodes”.

## A.7 Window menu

The **Window** menu contains commands for bringing windows to the front, and for opening special windows.



- |                          |   |
|--------------------------|---|
| <b>Bring to Front</b>    | Displays a list of the current windows; select one to display on top.   |
| <b>Show Memory Usage</b> | Displays a window showing memory usage. See Appendix C, “Memory usage”. |
| <b>Show Clipboard</b>    | Shows the contents of the clipboard in a window.                        |
| <b>Show Page Breaks</b>  | Shows page breaks for the currently active diagram.                     |



# Analytica Specifications

---

These are the specifications for Analytica 1.1.1

## Hardware and software

<b>CPU<sub>s</sub> supported</b>	Power Macintosh.
<b>System Software</b>	System-7 and later.
<b>DeltaGraph</b>	3.0 and up
<b>Memory requirements</b>	4MB minimum, 8 MB recommended, 16 MB or more for large models (allocated RAM)
<b>Application size</b>	1.9MB

---

## Objects

<b>Number of system objects</b>	400
<b>Maximum user-defined objects</b>	14900

---



<b>Uncertainty</b>	<b>Probability methods</b>	Random Latin HyperCube Median Latin HyperCube Monte Carlo
	<b>Maximum sample size</b>	32000
	<b>Random sampling methods</b>	SANE Minimal Standard L'Ecuyer Knuth

---

<b>Numbers and Arrays</b>	<b>Number precision</b>	14 digits, +- 10e306
	<b>Maximum elements in a dimension</b>	32000
	<b>Maximum dimensions in an array</b>	15

---

<b>Formats</b>	<b>Copy/Export</b>	<b>Paste/Import</b>
<b>Tables</b>	tab- or return-delimited text	tab- or return-delimited text
<b>Graphs</b>	PICT	n/a
<b>Diagrams</b>	PICT	n/a
<b>Icons</b>	ICON	ICON
<b>Graphics</b>	n/a	PICT

---

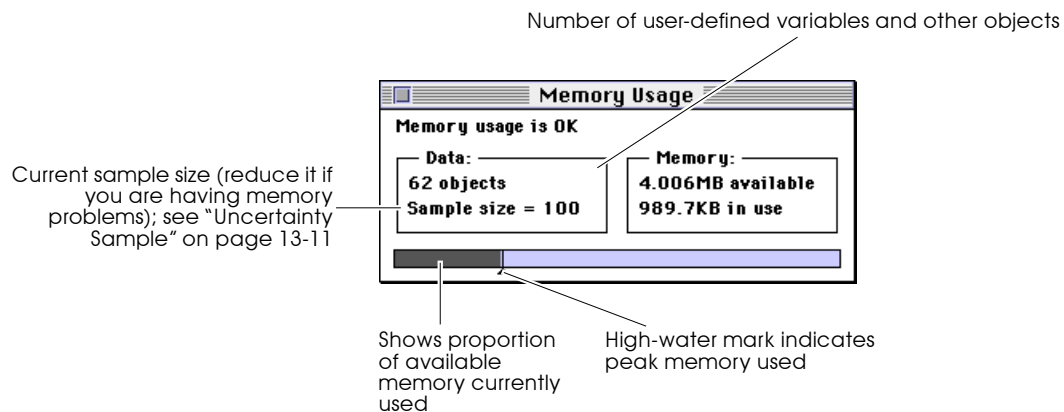
**System 7 Support** Balloon Help, Publish and Subscribe, Stationery, Aliases, Apple Events.

# Memory

## C.1 Memory usage

The Memory Usage window provides information on how much RAM memory the model is using, and the number of objects and sample size that affect it.

To display the Memory Usage window, select **Show Memory Usage** from the **Window** menu.



**Note:** 

This window appears automatically when Analytica runs low on memory.

If your model runs low on memory, you should first increase Analytica's memory allocation (see section C.2). If the maximum RAM memory allocation is not sufficient, consider adding memory or turning on virtual memory. Using


virtual memory allows you to run larger models, but may reduce your computer's performance noticeably.

If you have a Power Macintosh, it came with virtual memory already turned on. With virtual memory turned on, Analytica uses less memory than when virtual memory is turned off. If you check the Get Info box before and after turning on virtual memory, you'll see that the numbers in the Memory Requirements portion of the box have changed. This is because turning on virtual memory allows Analytica to use memory more efficiently. On a Power Macintosh, try setting virtual memory to 50% more than the amount of physical memory (RAM) — for instance, if your computer has 8 MB of RAM, set VM to 12 MB.

### Memory message on opening a model

When you save a model, the number of megabytes of peak memory used during the session is also saved. On opening the model, the saved peak memory is compared to the amount of memory allocated to Analytica. If the saved peak memory exceeds 95% of Analytica's memory allocation, a message will either recommend reducing the sample size (see section 13.6) or changing the application memory size (see next section).

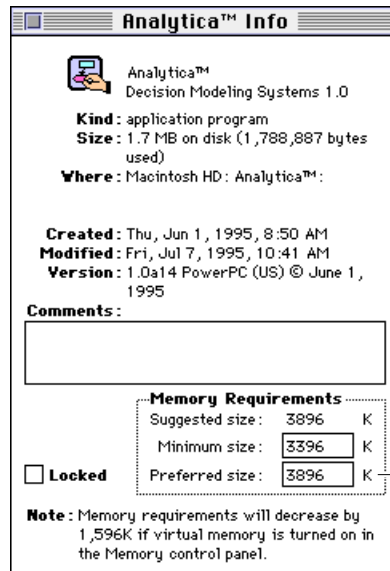
## C.2 Increasing memory allocation

To find out how much memory your system requires, quit all applications and select **About this Macintosh** from the finder's **Apple** () menu. The largest unused block is the maximum amount of memory available for allocation to Analytica.

If you run low on memory, and your Macintosh has sufficient RAM available, you can increase the memory assigned to Analytica:

1. Make sure you are in the finder and Analytica is not currently running.

2. Select the Analytica application with a single-click to highlight it. Do not double-click, which will start up Analytica.
3. Select **Get Info** (⌘-I) from the **File** menu to display the Info window for Analytica.



4. Enter the amount of memory, in kilobytes, to be assigned to Analytica. This is the maximum memory it can use.
5. Close the Info window.
6. Start up Analytica.

### What if the memory isn't allocated?

It is possible that after setting the preferred memory size, say to 12000KB, and starting Analytica, the Mac OS will allocate only a fraction of the preferred memory, say 8000KB. To obtain more memory for Analytica, do the following:

1. Close Analytica and all other applications that are running.
2. Start Analytica before starting any other applications.

This will solve the two likely problems:

### **Insufficient free memory**

When you start Analytica, the computer allocates whatever memory is remaining in your system to Analytica. Closing all applications frees up more memory for starting up Analytica.

### **Smaller continuous block**

When you start Analytica while other programs are running, the computer memory may be fragmented. Analytica will be assigned the largest continuous block of memory available, which may not be as large as the preferred memory size. Closing all applications and starting Analytica first gives Analytica the preferred memory size.

# Selecting the Sample Size

Each probabilistic value is simulated by computing a random sample of values from the actual probability distribution.

You can control the sampling method and sample size using the Uncertainty Setup dialog box (see Section 13.6). This appendix briefly discusses how to select a sample size.

## Choosing an appropriate sample size

There is a clear trade-off for using a larger sample size in calculating an uncertainty variable. When you set the sample size to a large value, the result is less noisy, but it takes a longer time to compute the distribution. For an initial probabilistic calculation, a sample size of 20 to 50 is usually adequate.

How should you choose the sample size  $m$ ? It depends both on the cost of each model run, and what you want the results for. An advantage of the Monte Carlo method is that you can apply many standard statistical techniques to estimate the precision of estimates of the output distribution. This is because the generated sample of values for each output variable is a random sample from the true probability distribution for that variable.

## Selecting the sample size: uncertainty about the mean

First, suppose you are primarily interested in the precision of the mean of your output variable  $y$ . Assume you have a random sample of  $m$  output values generated by Monte Carlo simulation:

$$(y_1, y_2, y_3, \dots, y_m) \quad (1)$$

You can estimate the mean and standard deviation of  $y$  using in the following equations:

$$\bar{y} = \sum_{i=1}^m \frac{y_i}{m} \quad (2)$$

$$s^2 = \sum_{i=1}^m \frac{(y_i - \bar{y})^2}{(m-1)} \quad (3)$$

This leads to the following confidence interval with confidence  $\alpha$ , where  $c$  is the deviation for the unit normal enclosing probability  $\alpha$ :

$$\left( \bar{y} - c \frac{s}{\sqrt{m}}, \bar{y} + c \frac{s}{\sqrt{m}} \right) \quad (4)$$

Suppose you wish to obtain an estimate of the mean of  $y$  with an  $\alpha$  confidence interval smaller than  $w$  units wide. What sample size do you need? You need to make sure that:

$$w > 2c \frac{s}{\sqrt{m}} \quad (5)$$

or, rearranging the inequality,

$$m > \left( \frac{2cs}{w} \right)^2 \quad (6)$$

To use this, first make a small Monte Carlo run with, say, 10 values to get an initial estimate of the variance of  $y$ , that is,  $s^2$ . You can then use Equation (6) to estimate how many samples will reduce the confidence interval to the requisite width  $w$ .

For example, suppose you wish to obtain a 95% confidence interval for the mean that is less than 20 units wide. Suppose your initial sample of 10 gives  $s = 40$ . The deviation  $c$  enclosing a probability of 95% for a unit normal is about 2. Substituting these numbers into Equation (6), you get:

$$m > \left( \frac{2 \times 2 \times 40}{20} \right)^2 = 8^2 = 64 \quad (7)$$

So, to get the required precision for the mean, you should set the sample size to about 64.

## Estimating confidence intervals for fractiles

Another criterion for selecting sample size is the precision of the estimate of the median and other fractiles, or more generally, the precision of the estimated cumulative distribution. Assume that the sample  $m$  values of  $y$  are relabeled so that they are in increasing order,

$$y_1 \leq y_2 \leq \dots y_m$$

and  $c$  is the deviation enclosing probability  $\alpha$  of the unit normal. Then the following pair of sample values constitutes the confidence interval:

$$(y_i, y_k)$$

where

$$i = \lfloor mp - c\sqrt{mp(1-p)} \rfloor \tag{8}$$

$$k = \lceil mp + c\sqrt{mp(1-p)} \rceil \tag{9}$$

Suppose you want to achieve sufficient precision such that the  $\alpha$  confidence interval for the  $p$ th fractile  $Y_p$  is given by  $(y_i, y_k)$ , where  $y_i$  is an estimate of  $Y_{p-\Delta p}$ , and  $y_k$  is an estimate of  $Y_{p+\Delta p}$ . In other words, you want  $\alpha$  confidence of  $Y_p$  being between the sample values used as estimates of the  $(p - \Delta p)$ th and  $(p + \Delta p)$ th fractiles. What sample size do you need? Ignoring the rounding, you have approximately

$$i = m(p - \Delta p) , \quad k = m(p + \Delta p) \tag{10}$$

Thus,

$$k - i = 2m\Delta p \tag{11}$$

From Equations (8) and (9) above, you have

$$k - i = 2c\sqrt{mp(1-p)} \tag{12}$$

Equating the two expressions for  $k - i$ , you obtain

$$2m\Delta p = 2c\sqrt{mp(1-p)} \tag{13}$$



$$m = p(1 - p)\left(\frac{c}{\Delta p}\right)^2 \quad (14)$$

For example, suppose you want to be 95% confident that the estimated fractile  $Y_{.90}$  is between the estimated fractiles  $Y_{.85}$  and  $Y_{.95}$ . So you have  $\Delta p = 0.05$ , and  $c \approx 2$ . Substituting the numbers into Equation (14), you get:

$$m = 0.90 \times (1 - 0.90) \times (2/0.05)^2 = 144 \quad (15)$$

On the other hand, suppose you want the credible interval for the least precise estimated percentile (the 50th percentile) to have a 95% confidence interval of plus or minus one estimated percentile. Then,

$$m = 0.5 \times (1 - 0.5) \times (2/0.01)^2 = 10,000 \quad (16)$$

Note that these results are completely independent of the shape of the distribution. If you find this an appropriate way to state your requirements for the precision of the estimated distribution, you can determine the sample size before doing *any* runs to see what sort of distribution it may be.

---

# Error Message Types

---

There are several types of error messages in Analytica. Many messages are designed to inform you that something in the model needs to be corrected; some messages indicate that Analytica cannot continue or complete your request. Each error message begins with its message type, one of: warning, lexical, syntax, evaluation, system, and fatal errors.

In general, Analytica allows you to continue working on your model unless it cannot proceed until a problem has been corrected. When you are editing a variable definition, you can request an error message by pressing the *enter* key or by clicking on the definition Warning icon (⚠).

## E.1 Warning

A warning indicates that there is a possible problem. For example:

**Warning:**

**Log of non-positive number.**

A warning is reported during result evaluation to inform you that continuing may yield unexpected results.

You can suppress evaluation warnings for all variables by disabling the Show result warnings preference (see Section 4.10, “Preferences dialog box”). When Show result warnings is unchecked, any warning conditions encountered during result evaluation will be ignored.

If an identifier in a module that you are adding to a model has a name conflict with an identifier in the model, you will see a warning similar to the following:

**Warning:**

**Can't declare Variable Location because the Identifier is already in use as Attribute Location.**

**Declare using the Identifier Location1?**

## E.2 Lexical error

A lexical error occurs when a component of an expression was expected and is missing or is invalid. For example, if you enter a number with an invalid number suffix, you may get a message similar to the following:

**Lexical error while checking:**

**2sdf**  
**^**

**Invalid exponent code.**

## E.3 Syntax error

A syntax error occurs when an expression contains a syntax mistake. Analytica often reports the mistake together with the fragment of the expression that contained the error. For example:

**Syntax error while checking:**

**2 ++ 3**  
**^**

**Expression expected.**

The following are two common syntax errors:

**expecting “,”**

Indicates a comma is missing, or there are too few parameters to a function.

**expecting “)”**

Indicates there are too many parameters to a function.

If you attempt to change the identifier for a variable, and the new identifier is assigned to another node, you will see a message similar to the following:

**Syntax error:**

**The Identifier “Location” is already in use.**

## E.4 Evaluation error

An evaluation error occurs when there is a problem while evaluating a variable, user-defined function, or system function. You are asked if you want to edit the definition of the variable currently being evaluated:

**Error during evaluation of Ch1.**

**Do you want to edit the Definition of Ch1?**

If a system function expects a specific kind of argument, an error message similar to the following is displayed:

**Evaluation error:**

**First argument of Sysfunction Argmax must be a table.**

This message indicates that an argument passed to the function is of a different type or cannot be handled by that function. You may

need to redefine a variable being used as an argument to the function, or change an expression being passed as an argument.

## E.5 Invalid number

If a calculation such as division by zero is performed, a warning is displayed with an option to continue calculating. Two possible error codes may be returned as a result of an invalid calculation:

Code	Meaning
INF	Infinity
NAN()	Invalid argument, such as <code>Sqrt(-1)</code> , or Invalid division, such <code>0/0</code> , etc.

## E.6 System error

If you see this message type, please contact Lumina Decision Systems to report the error.

## E.7 Out of memory error

Indicates that Analytica has used up all available memory and cannot complete the current command. If this occurs, first save your model. Before attempting to evaluate again, close some windows, use a smaller sample size, or expand the memory available to Analytica (see Appendix C, “Memory”).

# Reserved Words

You cannot use the following terms as identifiers. If you attempt to assign one as an identifier, Analytica appends a number to the new identifier to keep it distinct from the existing term.

Abbreviation	Choice	Depth	Evalindex	Help
Abs	Cicn	Derived	Every	HyperTalk
Activate	Class	Description	Exp	lacQuit
Alias	Clipboard	Determ	Expand	Icon
Aliases	Clock	Determinant	Export	Identifier
All	Close	Determtable	Expr	If
AllShow	CloseGraph	DetermType	Factorial	If0
And	Collapse	Developeron	False	Ifall
Any	Color	Diagram	Field	Ifonly
Append	Command	DiagState	FileInfo	Ifpos
Arctan	Comment	DisplayInputs	FixedCheck	Import
Area	Concat	DisplayOutputs	FixedDef	Imports
Argmax	Constant	Distresol	Flip	In
Array	Contains	Distribsize	FontStyle	Include
ArrayT	Context	Diststeps	For	Includexzero
ArrayType	Correlation	Do	Form	Includeyzero
Ascending	Cos	Domain	FormNode	Includezzero
AskAttribute	Cubicinterp	Dydx	Fractiles	Index
AskPause	Cumdist	Dynamic	Frame	IndexType
Attribute	Cumproduct	DynamicValue	Framequto	IndexVals
Author	Cumulate	DynInputs	Frequency	Inf
Average	D2	DynOutputs	FullScreen	InFromRec
Balloonhelp	D3	Each	Function	InObjs
Bernoulli	Date	Edit	FunctionOf	Inputs
Beta	DdeExecute	Edition	Get	Integrate
Bigbrother	DdelInitiate	Editor	Getfract	Intercept
Button	DdeTerminate	EditTable	GetResource	Invert
Bye	Decision	Elasticity	Graph	IsIn
Cdf	Decompose	Else	GraphSetup	It
Cdfresol	DefaultSize	End	Graphvar	KeyOff
Cdfsteps	Definition	EndPhoto	Graphwindows	Keyword
Chancedist	DefnState	EndRecord	Grid	Kurtosis
Chancevar	Delete	EndUpdate	Hance	Laser
Checking	Delimit	Evaldynamic	Graphwindows	Laserheight

LaserP	NodeSize	Profile	Sdeviation	Trash
Laserwidth	Normal	Project	Section	Tree
Launch	Normalize	Publisher	Self	Triangular
Levels	Normal_fracs	Publishoneval	Send	True
Library	Not	Put	Sequence	Truncate
Linear	NumberFormat	Randomseed	Set	Tutorial
Linearinterp	Numberwidth	Randomtype	SetPath	Typechecking
Linestyle	Numeric	Range	Show	Uncumulate
LinkLibrary	Object	Rank	Showhier	Undef
LinkModule	Objective	Rankcorrel	Showundef	Undefined
List	Of	ReadFile	Sin	Unevaluated
Ln	Open	Record	Size	Uniform
LocalVar	OpenFile	Rectangular	Skewness	Units
Location	Or	Reform	Slice	Untitled
Lognormal	OrigContains	ReformDef	Slide	Update
Logten	Original	ReformVal	SlideP	UseCheck
Lookup	Orphans	Renae	SoftwareVersion	Usetable
MacScribe	Output	Repeat	Sortindex	Using
Makerect	Outputs	Repeatln	Sqr	Usinglocal
MakeSticky	Pagesetup	Request	Sqrt	Value
Manual	Pan	ResumePhoto	Statistics	ValueState
Map	Parameters	ResumeRecord	Statslabels	Valuevar
MapVar	ParamNames	Role	Statsselect	Variable
Max	ParamTypes	Rotation	Sticky	Variance
Mean	Path	Round	Subindex	Varlist
Mesh	PausePhoto	Run	Subscriber	VarType
Mid	Pdf	Samp	Subscript	Vector
Min	PdfValue	Sample	Subset	Verbosity
Modauthor	Photo	Samplesize	Sum	Version
Mode	Pi	Samplotype	Symbolsize	View
Model	Pict	Save	SysFunction	Webhelper
Module	Picture	SaveAuthor	Sysfunctions	What
MoreHelp	Plotheight	SaveDate	SysVa	Whatif
Move	Plotwidth	Saveformat	Table	Why
Namesize	Positive	Saveoptions	Tabletype	WhyAll
Naming	Pred	Savevalues	Tabwidth	Windows
Nan	Preference	Scales	Terminaltype	With
Need	PrintTable	Scalar	Text	Xintervals
Neededby	Prob	Scatter	TexType	Xmaximum
Needs	Probability	SchedulePublish	Then	Xminimum
News	Probbands	Scope	Ticks	Yes
No	Prodist	Scoping	Tilt	Yintervals
NodeColor-	Probindex	Screenheight	Time	Ymaximum
DiagramColor				
NodeFont	Prohtable	Screenwidth	Title	Yminimum
NodeInfo	ProbValue	Scribe	To	Zmaximum
Node location	Product	Script	Transpose	Zminimum

## References

---

- Morgan, M. Granger and Henrion, Max. *Uncertainty: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis*, Cambridge University Press (1990).

Written by the original authors of Analytica, this text provides extensive background on how to represent and analyze uncertainty in quantitative models. It includes chapters on:

Building good policy models

Categorizing types and sources of uncertainty

How people make judgments under uncertainty

Encoding expert judgment in the form of probability distributions

Choosing a computational method for propagating uncertainty in a model

Analyzing uncertainty in very large models

Displaying and communicating uncertainty

How to tell if representing uncertainty could make a significant difference to your conclusions, or “the value of knowing how little you know”

We recommend that you ignore Chapter 10, which describes Demos, the precursor of Analytica, because it is quite out of date!

- Clemen, Robert T. *Making Hard Decisions: An Introduction to Decision Analysis*. Duxbury Press (1991).
- Howard, R., and Matheson, J. Influence Diagrams. In *Readings on the Principles and Applications of Decision Analysis*,



- eds. R. Howard and J. Matheson. pp. 721-762. Menlo Park, Calif.: Strategic Decisions Group (1981).
- Keeney, R. *Value - Focused Thinking: A Path to Creative Decision Making*, Cambridge, MA: Harvard University Press (1992).
  - Knuth, D.E. *Seminumerical Algorithms, 2nd ed., vol. 2 of The Art of Computer Programming*, Reading, MA: Addison-Wesley (1981).
  - L'Ecuyer, P. *Communications of the ACM*, **31**, 742-774 (1988).
  - Park, S.K., and K.W. Miller. *Communications of the ACM*, **31**, 1192-1201 (1988).
  - Pearl, J. *Probabilistic Reasoning in Intelligent Systems*, San Mateo, Calif.: Morgan Kaufmann (1988).

# How to Contact Us

---

If you have any questions or comments about our products, please contact us.

## **For Sales or Customer Service**

### **By mail**

Lumina Decision Systems, Inc.  
26010 Highland Way  
Los Gatos, CA, 95033

### **By electronic mail**

info@lumina.com

### **By fax**

(650) 240-2230

### **By phone**

(877) 6-LUMINA  
(650) 212-1212

## **For Technical Support**

### **By mail**

Lumina Decision Systems, Inc.  
26010 Highland Way  
Los Gatos, CA, 95033

### **By electronic mail**

support@lumina.com

### **By fax**

(650) 240-2230

### **By phone**

(877) 6-LUMINA  
(650) 212-1212

Or, please visit our Web Site located at <http://www.lumina.com>



# Glossary

---

- Alias** A node in a diagram that refers to a variable or other node located somewhere else, usually in another module. An alias permits you to display a variable in more than one module. An alias node is distinguished by having its title in italics.
- Array** A collection of values that can be viewed as one or more tables. An array has one or more dimensions; each dimension is identified by an index.
- Arrow** An arrow or influence from one variable node to another indicates that the origin node affects (influences) the destination node. If the nodes depict variables, the origin variable usually appears in the definition of the destination variable.
- Arrow tool** The Arrow tool, or Influence Arrow tool, is in the shape of a left-to-right pointing arrow cursor. The Arrow tool is used to draw arrows connecting variables to create relations between them.
- Attribute** A property or descriptor of an object, such as its title, description, definition, value, or inputs.
- Attribute panel** An auxiliary window pane that you can open below a diagram or outline window. Use the Attribute panel to rapidly examine one attribute at a time of any variable in the model, by selecting the variable and then the attribute from a popup menu.
- Author** The person or people who created the model. By default, the author is the name listed in the Macintosh Chooser or Sharing Setup.
- Behavior analysis** Model behavior analysis is a type of sensitivity analysis in which you specify a set of alternative values for one or more inputs and

examine the effect on selected model output variables. It is also known as parametric analysis.

**Browse tool** The Browse tool is in the shape of a hand. With the Browse tool, you can examine the diagram but cannot make any changes, except to change the values in input nodes.

**Chance variable** A Chance variable is uncertain and cannot be directly controlled by the decision maker. Usually, it is defined by a probability distribution. A Chance variable is depicted as an oval node.

**Check** The check attribute contains an expression that checks the validity of the value of a variable. It displays a message when the variable's value is out of specified bounds.

**Class** The type of Analytica objects, such as decision, chance, objective or index variables, functions, and modules, libraries, form or 0.model.

**Conditional dependency** A Chance variable  $a$  is conditionally dependent on another variable  $b$  if the probability of a value of  $a$  depends on the value of  $b$ . If  $a$  is defined by a probability table,  $b$  may be an index of its probability table.

**Constant** A variable whose value is not probabilistic, and does not depend on other variables, such as the number of minutes in an hour.

**Continuous distribution** A probability distribution defined for a continuous variable, that is, for a real-valued variable. Example continuous distributions are beta, normal, and uniform. Compare to *discrete distribution*.

**Continuous variable** A variable whose value is a real number, that is, one of an infinite number of possible values. Its range can be bounded (for example, between 0 and 1) or unbounded. Compare to *discrete variable*.

**Created** The date and time at which the model was first created. This model attribute is entered automatically, and is not user-modifiable.

---

<b>Cumulative probability distribution</b>	A representation of a probability distribution that plots the cumulative probability that the actual value of the uncertain variable $x$ will be less than or equal to each possible value of $x$ . The cumulative probability distribution is a display option in the Uncertainty View popup menu.
<b>Cyclic dependency</b>	A cyclic dependency occurs when a variable depends on itself directly or indirectly so that the arrows form a directed circular path. The only cyclic dependencies allowed in Analytica are in variables using the <code>DYNAMIC()</code> function that contain a time lag on the cycle.
<b>Decision variable</b>	A variable that the decision maker can control directly. Decision variables are represented by rectangular nodes.
<b>Definition</b>	A formula that defines how to compute a variable's value. It can be a simple number, a mathematical expression, a list of values, a table, or a probability distribution. In text format it is limited in length to 32,000 characters.
<b>Description</b>	Text explaining what the node represents in the real system being modeled. It is limited in length to 32,000 characters.
<b>Deterministic table</b>	A deterministic function that gives the value of a variable $x$ conditional on the values of its input variables. The input must all be discrete variables. The table is indexed by each of its inputs, and gives the value of $x$ that corresponds to each combination of values of its inputs.
<b>Deterministic value</b>	A variable's deterministic value, or mid value, is a calculation of the variable's value assuming all uncertain inputs are fixed at their median values.
<b>Deterministic (determ) variable</b>	A variable that is a deterministic function of its inputs. Its definition does not contain a probability distribution. The value of a deterministic variable can be probabilistic if one or more of its inputs is uncertain. A deterministic variable is displayed as a double oval. You can also use a general variable (rounded rectangle) to depict a deterministic variable.

<b>Determtable</b>	See <i>deterministic table</i> .
<b>Diagram</b>	See <i>influence diagram</i> .
<b>Dimension</b>	An array has one or more dimensions. Each dimension is identified by an index variable. When an array is shown as a table, the row header (vertical) and column headers(horizontal) give the two dimensions of the table.
<b>Discrete distribution</b>	A probability distribution over a finite number of possible values. Example discrete distributions are Bernoulli and the Probtale function. Compare to <i>continuous distribution</i> .
<b>Discrete variable</b>	A variable whose value is one of a finite number of possible values. Examples are the number of days in a month (28, 29, 30, or 31), or a Boolean variable with possible values <code>True</code> and <code>False</code> . A variable that is defined as a list or list of labels is discrete. Compare to <i>continuous variable</i> .
<b>Domain</b>	The possible outcomes of a variable. The Domain has a type as well as value. The possible types are List of labels, List of numbers, or Continuous; the default type is Continuous, except for variables defined with the <code>Choice()</code> , <code>Probtale()</code> , and <code>Determtable()</code> functions.
<b>Dynamic variable</b>	A variable that depends on the system variable <i>Time</i> and is defined by the <code>Dynamic()</code> function. A dynamic variable can depend on itself at a previous time period, directly or indirectly, through other dynamic variables.
<b>Edit table</b>	A definition that is a table is also called an edit table because it can be edited.
<b>Edit tool</b>	The Edit tool is in the shape of the normal Macintosh pointer cursor. The Edit tool is used to create a new model or to change an existing model. It allows you to move, resize, and edit nodes, and exposes the Arrow tool and node palette.

---

<b>Expression</b>	A formula that can contain numbers, variables, functions, distributions, and operators, such as $0.5$ , $a-b$ , or $\text{Min}(x)$ , combined according to the Analytica language syntax. The definition of a variable must contain an expression.
<b>Expression type</b>	The Expression popup menu, which appears above the definition field, allows you to change the definition of a variable to one of several different kinds of expressions. Expression types include expression, list (of expressions or numbers), list of labels (text values), table, probability table, and distribution. Any definition, regardless of expression type, can be viewed as an expression.
<b>File Info</b>	The name of the file and folders in which the model was last saved.
<b>Filed library</b>	A library whose contents are saved in a file separate from the model that contains it. A filed library can be shared among several models without making a copy for each model.
<b>Filed module</b>	A module whose contents are saved in a file separate from the model that contains it. A filed module can be shared among several models without making a copy for each model.
<b>Fractile</b>	The median is the 0.5 fractile. More generally, there is probability $p$ that the value is less than or equal to the $p$ fractile. Quantile is a synonym for fractile. Fractal is something different! Compare to <i>percentile</i> .
<b>General variable</b>	A variable that can be certain or probabilistic. It is often convenient to define a variable as a general variable without worrying about what particular kind of variable it is. A general variable is depicted by a rounded rectangle node.
<b>Help</b>	The help attribute consists of text that is displayed in a balloon when you move the mouse over this node in a diagram, when Balloon Help is switched on (see page-xii or your Macintosh System 7 reference). If the help attribute is not displayed, Balloon Help uses the text in the description attribute.



- Identifier** A short name for a variable used in mathematical expressions in definitions. An identifier must start with a letter, have no more than 20 characters, and contain only letters, numbers, '.' (period), and '\_' (underscore, used instead of a space). Each identifier in a model must be unique. Compare to *title*.
- Importance analysis** Importance analysis lets you determine how much effect the uncertainty of one or more input variables have on the uncertainty of an output variable. Analytica defines importance as the rank order correlation between the sample of output values and the sample for each uncertain input. It is a robust measure of the uncertain contribution because it is insensitive to extreme values and skewed distributions.
- Unlike commonly used deterministic measures of sensitivity, this rank order correlation averages over the entire joint probability distribution. Therefore, it works well even for models where the sensitivity to one input depends strongly on the value of another.
- Index** An index of an array identifies a dimension of that array. An index is usually a variable defined as a list, list of labels, or sequence. An index is often, but not always, a variable with a node class of Index.
- Indexes** Plural of index, indicates a set of index variables that define the dimensions of a table (in an edit table or value).
- Index selection area** The top portion of a Result window, containing a description of the result and other information about the dimensions of the result.
- Index variable** A class of variable, defined as a list, list of labels, or sequence, that identifies the dimensions of an array, for example, in an edit table. An Index variable is depicted as a parallelogram node. Variables of other classes whose definition or domain consist of list, list of labels, or sequence can also be used to identify the dimensions of an array, and are sometimes referred to as index variables.
- Influence arrow** See *arrow*.

---

<b>Influence diagram</b>	An intuitive graphical view of the structure of a model, consisting of nodes and arrows. Influence diagrams provide a clear visual way to express uncertain knowledge about the state of the world, decisions, objectives, and their interrelationships.
<b>Innermost dimension</b>	The dimension of an array that varies most rapidly in the <code>Table()</code> function. The innermost dimension is the last index listed in a <code>Table()</code> or <code>Array()</code> function. Compare to <i>outermost dimension</i> .
<b>Input</b>	A variable that appears in the definition of the selected variable. See also <i>output</i> .
<b>Input arrowhead</b>	An arrowhead pointing into a node, indicating that the node has one or more inputs from outside its module. Press the arrowhead for a popup menu of the input variables.
<b>Inputs</b>	A list of the variables or functions on which this variable or function depends. The inputs are determined by the arrows drawn to and the variables or functions referred to in this variable's or function's definition or check attribute. See also <i>outputs</i> .
<b>Key</b>	In a results graph, the key shows the value of the key index variable that corresponds to each curve, indicated by pattern or color.
<b>Kurtosis</b>	A measure of the peakedness of a distribution. A distribution with long thin tails has a positive kurtosis. A distribution with short tails and high shoulders, such as the uniform distribution, has a negative kurtosis. A normal distribution has zero kurtosis.
<b>Last Saved</b>	The date and time at which the model was last saved. This model attribute is entered automatically, and is not user-modifiable. If the model is new, this field remains empty until the model is first saved.
<b>Library</b>	A model component that typically contains a collection of user-defined functions and/or variables to be shared.

- List** A type of expression available in the Expression popup menu consisting of an ordered set of numbers or expressions. A list is often used to define Index and Decision variables.
- List of labels** A type of expression available in the Expression popup menu consisting of an ordered set of text items. A list of labels is often used to define Index and Decision variables.
- Matrix** A two-dimensional array of numbers with indexes of equal length.
- Mean** The average of the population, weighted by the probability mass or density for each value. The mean is also called the *expected value*. The mean is the center of gravity of the probability density function.
- Median** The value that divides the range of possible values of a quantity into two equally probable parts. Thus, there is 0.5 probability that the uncertain quantity is less than or equal to the median, and 0.5 probability that it is greater than the median.
- Mid value** The result of evaluating a variable deterministically, holding probability distributions at their median value. Analytica calculates the mid value of a variable by using the mid value of each input. The mid value is a measure of central value, computed very quickly compared to uncertainty values. Compare *probvalue*.
- Mode** The most probable value of the quantity. The mode is at the highest peak of the probability density function. On the cumulative probability distribution, the mode is at the steepest slope, at the point of inflection.
- Model** A module, or a hierarchy of linked and/or embedded modules and libraries, on which you work during an Analytica session; the main, or root, module at the top of the module hierarchy. Between sessions, a model is stored in an Analytica document file.
- Module** A collection of related nodes, typically including variables, functions, and other modules, organized as a separate influence diagram. A module is depicted in a diagram as a node with a thick outline.

---

<b>Module hierarchy</b>	A model can contain several modules, each one containing details of the model. Each module can contain further modules, containing still more detail. This module hierarchy is organized as a tree with the model at the top. You can view the hierarchical structure in the Outline window.
<b>Multimodal distribution</b>	A probability distribution that has more than one mode.
<b>Node</b>	A shape, such as a rectangle, oval, or hexagon, that represents an object in an influence diagram. Different node shapes are used to represent different types of variables.
<b>Object</b>	A variable, function, or module in an Analytica model. Each object is depicted as a node in an influence diagram, and is described by a set of attributes. See also <i>class</i> , <i>node</i> , <i>attribute</i> , and <i>influence diagram</i> .
<b>Object Finder</b>	A dialog box used to browse and edit the functions and variables available in a model.
<b>Object window</b>	A view of the detailed information about a node. The object window shows the visible attributes, such as a node's type, identifier, and description.
<b>Objective variable</b>	A variable that evaluates the overall desirability of possible outcomes. The objective can be measured as cost, value, or utility. A purpose of most decision models is to find the decision or decisions that optimize the objective, for example, minimizing cost or maximizing expected utility. An objective variable is represented by a hexagonal node.
<b>Operator</b>	A symbol, such as a plus sign (+), that represents a computational process or action such as addition or comparison.
<b>Outermost dimension</b>	The dimension of an array that varies least rapidly in the <code>Table()</code> function. The outermost dimension is the first index listed in a <code>Table()</code> or <code>Array()</code> function. Compare to <i>innermost dimension</i> .

<b>Outline window</b>	A view of a model that lists the objects it contains as a hierarchical outline.
<b>Output</b>	A variable whose definition refers to the selected variable. See also <i>input</i> .
<b>Output Arrowhead</b>	An arrowhead pointing out of a node, indicating that the node has one or more outputs outside its module. Press the arrowhead for a popup menu of the output variables.
<b>Outputs</b>	A list of the variables or functions that depend on this variable or function. The outputs are determined by the arrows drawn from this variable or function and the variables or functions in whose definition or check attribute this variable or function appears. See also <i>inputs</i> .
<b>Parameters</b>	The arguments of a function.
<b>Parametric analysis</b>	See <i>behavior analysis</i> .
<b>Parent diagram</b>	The diagram for the module that contains this object.
<b>Percentile</b>	The median is the fiftieth percentile (also written as 50%ile). More generally, there is probability $p$ that the value is less than or equal to the $p$ th percentile. Compare to <i>fractile</i> .
<b>Probabilistic variable</b>	A variable that is uncertain, and is described by a probability distribution. A probabilistic variable is evaluated using simulation; its result is an array of sample values indexed by <i>Run</i> .
<b>Probability bands</b>	Probability bands are a way to display the uncertainty in a value by showing percentiles from its distribution, for example the 5%, 25%, 50%, 75%, and 95% percentiles. On a graph, these often appear as bands around the median (50%) line. Probability bands are also referred to as credible intervals.
<b>Probability density function</b>	A representation of a probability distribution that plots the probability density against the value of the variable. The probability density at each value of $X$ is the relative probability that $X$ will be at or near that value. The probability density

---

function can be displayed for continuous, but not discrete variables. It is a display option in the Uncertainty View popup menu. Compare to *Probability mass function*, which is used with discrete variables.

**Probability distribution** A probability distribution describes the relative likelihood of a variable having different possible values.

**Probability mass function** A probability mass function is a representation of a probability distribution for a discrete variable as a bar graph, showing the probability that the variable will take each possible value. The probability mass function can be displayed for discrete, but not continuous variables. It is a display option in the Uncertainty mode View menu. Compare to *Probability density function*, which is used with continuous variables.

**Probability table** A table for specifying a discrete probability distribution for a Chance variable. In a probability table, you specify the numerical probability for each value in the domain of the variable. If the variable depends on (that is, is conditioned by) other discrete variables, each of these conditioning variables gives an additional dimension to the table, so you can specify the probability distribution conditional on the value of each conditioning variable.

**Protable** See *probability table*.

**Probvalue** The probabilistic value of a variable, represented as a random sample of values from the probability distribution for the variable. The probvalue for a variable is based on the probvalue for the inputs to the variable. See also *probabilistic variable* and compare to *mid value*.

**Reducing function** A function that operates on an array over one of its indexes. The result of a reducing function has that dimension removed, and hence has one fewer dimension.

**Remote variable** A variable in another module, not shown in the active diagram. Typically a remote variable is an input or output of a node in the active diagram.

<b>Resize box</b>	The little square at the bottom right corner of a Macintosh window used to enlarge or shrink the window.
<b>Result view</b>	A window that shows the value of a variable as a table or graph.
<b>Sample</b>	An array of values selected at random from the underlying probability distribution for a quantity. Analytica represents uncertainty about a quantity as a sample, and estimates statistics, probability density function, and other representations of a probability distribution from the sample.
<b>Sampling method</b>	A method used to generate a random sample from the probability distributions in a model (for example, Monte Carlo and Latin Hypercube).
<b>Scalar</b>	A value that is a single number.
<b>Scatter plot</b>	A graph that plots the samples of two probabilistic variables against each other.
<b>Self</b>	A keyword used in two different ways: <ol style="list-style-type: none"><li>1. Refers to the index of a table that is indexed by itself. <code>Self</code> refers to the alternative values of the variable defined by the table.</li><li>2. Refers to the variable itself, as a substitute for the variable's identifier, in a <code>Check</code> attribute expression or a <code>Dynamic</code> expression.</li></ol>
<b>Sensitivity analysis</b>	A method to identify and compare the effects of various input variables to a model on a selected output. Example methods for sensitivity analysis are importance analysis and model behavior analysis.
<b>Skewness</b>	A measure of the asymmetry of the distribution. A positively skewed distribution has a thicker upper tail than lower tail, while a negatively skewed distribution has a thicker lower tail than upper tail. A normal distribution has a skewness of zero.

---

<b>Slice</b>	A slice of an array is an element or subarray selected along a specified dimension. A slice has one less dimension than the array from which it is sliced.
<b>Standard deviation</b>	The square root of the variance. The standard deviation of an uncertainty distribution reflects the amount of spread or dispersion in the distribution.
<b>Stationery</b>	A type of Macintosh document or file that is used as a template for creating new documents or files of the same type.
<b>Suffix</b>	Numbers, such as 10K, 123M, or 1.23u, are in suffix notation. The suffix letter denotes a power of ten, for example, K, M, and u denote $10^3$ , $10^6$ , and $10^{-6}$ , respectively.
<b>Symmetrical distribution</b>	A distribution, such as a normal distribution, that is symmetrical about its mean.
<b>System function</b>	A function available in the Analytica modeling language. See also <i>User-defined function</i> .
<b>System variable</b>	A variable that is part of the Analytica modeling language, such as <i>Samplesize</i> or <i>Time</i> .
<b>Table</b>	A two dimensional view of an array. The array can have more than two dimensions, but only two can be seen at one time. In the Result window, click on the Table button to select the table view of an array-valued result.
<b>Tail</b>	The upper and lower tails of a probability distribution contain the extreme high and low quantity, respectively. Typically, the lower and upper tails include the lower and upper ten percent of the probability, respectively.
<b>Title</b>	The full name of an Analytica object. A variable's or module's title is displayed in its node, in window titles, and in object lists. It is limited to 255 characters. It can contain any characters, including spaces and punctuation. Compare to <i>identifier</i> .
<b>Uncertain value</b>	See <i>Probvalue</i> .



**Units** The units of measurement for a variable. Units are used to annotate tables and graphs; they are not used in any calculation.

**User-defined function** A function that the user defines to augment the functions provided as part of the Analytica modeling language.

**Value** See *mid value*.

**Variable** An object that has a value, which may be text, a number, or an array.

**Variance** A measure of the uncertainty or dispersion of a distribution. The wider the distribution, the greater its variance.

**Zoom box** The little square at the top right corner of a Macintosh window. Click in the zoom box to enlarge the window to the full screen size or to reduce it.



# Index

## Symbols

- (subtraction) operator 10-4, 11-24
- < (less than) operator 10-5
- ≠ (not equal) operator 10-5
- \* (multiplication) operator 10-4, 11-24
- + (addition) operator 10-4, 11-24
- / (division) operator 10-4, 11-24
- = (equal) operator 10-5
- > (greater than) operator 10-5
- ^ (exponential) operator 10-4, 11-24
- ≤ (less than or equal to) operator 10-5
- ≥ (greater than or equal to) operator 10-5

## A

- Abs() function 10-7
- Accept button 8-3, 11-22
- Add Module** command 19-12, A-1
- Add Module dialog box 19-12
- Adjust Size** command 6-4, A-12
- aliases 4-12, GL-1
  - creating 4-9, 4-12
  - modifying 4-15
- Align to Grid** command 4-6, 6-6, A-12
- Arctan() function 10-7
- Area() function 12-12
- Argmax() function 12-13
- arithmetic operations 10-4, 11-24
- Array** command A-7
- Array library 12-2, A-7
- Array() function 12-6, 12-7, 12-10
- arrays 11-1–11-28, GL-1
  - arithmetic operations on 11-24
  - changing index of 12-9
  - conventions for 11-23
  - defining 12-7
  - functions that create 12-5
  - operating on 11-5–11-9, 11-24–11-28, 12-1–12-5
  - referencing an element of 12-21, 12-22
- Arraytype qualifier 20-3
- Arrow tool 4-7, GL-1
- Arrow tool button 1-4
- arrows 1-7, GL-1
  - and nodes 4-7
  - arranging 6-4
  - automatically drawn 4-9, 8-5
  - between modules 4-10
  - bold 19-13
  - changes to model when created 4-9
  - creating 4-8
  - deleting 4-8
  - drawing 4-7–4-9
  - drawing between 2 modules 4-14
  - dynamic 6-12, 17-10
  - hiding 6-6, 6-11, 6-13
  - removing 4-8
  - showing 6-11, 6-13
  - small arrow head 4-8
  - to and from indexes 11-4
- Ascending qualifier 20-3
- Attrib Of Ident function 19-8
- Attribute panel 1-14, GL-1
  - resizing 1-16
- Attribute** popup menu 1-14
- attributes 19-6–19-8, GL-1
  - creating new 19-8
  - displaying 8-12, 19-7
  - editing 4-15
  - in a definition 19-8
  - of functions 1-15, 19-6, 20-2
  - of modules 1-15, 19-6
  - of variables 1-15, 19-6
  - renaming 19-8
  - user-created 19-6
- Attributes** command 8-12, 19-7, A-5
- Attributes dialog box 8-12, 19-7, 19-8
- Author 19-6, GL-1

Average() function 12-13

## B

Balloon Help xii, 9-9, GL-5  
behavior analysis 3-1, GL-1  
Bernoulli() function 15-7  
Beta() function 14-1  
bevel, displaying 6-13  
Boolean number format 7-10  
Boolean operators 10-5  
Boolean values 10-4  
Boolean variables 13-2  
border, displaying 6-13  
**Bring to Front** command A-13  
Browse mode 1-5, 9-2  
Browse tool 1-5, GL-2  
Browse tool button 1-5

## C

Cancel button 11-22  
cancel button 8-3  
cells 11-15, 11-21  
    adding 11-21  
    copying 11-22  
    deleting 11-16, 11-21  
    editing 11-21  
    inserting 11-15  
    pasting 11-22  
Certain() function 15-8  
Chance variables 1-10, GL-2  
Chancedist() function 15-9  
Check 4-21, 8-12, 19-6, GL-2  
check value bounds 4-21  
check variable class 4-21  
Choice option 9-3  
Choice() function 12-20  
Class 1-9, 4-17, 19-6, GL-2  
**Class** popup menu 4-17  
**Clear** command A-3  
clipboard A-13  
**Close** command A-2  
**Close Model** command 1-2, A-2  
color  
    in influence diagrams 6-9  
    of input and output node 9-6  
color palette 6-10

column index 2-4  
columns 11-21  
comparing results 2-14  
comparison operators 10-5, 11-25  
computation time D-1  
Concat() function 12-26  
conditional dependencies 15-6, GL-2  
conditional deterministic table 15-10  
conditional operators 10-6, 11-26  
conditional probability tables 15-4  
confidence intervals D-2, D-3  
constants 1-10, GL-2  
Context qualifier 20-3  
continuous distributions 13-2, GL-2  
continuous variable GL-2  
conventions  
    for array examples 11-23  
    terminological xi  
    typographic xi  
**Copy** command 4-6, 11-22, 18-1–18-3, A-3  
**Copy Diagram/Table/Object** command 18-2, A-4  
Correlation() function 16-2  
Cos() function 10-7  
**Create Publisher** command 18-6  
Created 19-6, GL-2  
Cubicinterp() function 12-24  
Cumdist() function 14-2  
Cumproduct() function 12-17  
Cumulate() function 12-17  
**Cumulative Probability** command 2-13, A-11  
cumulative probability distributions GL-3  
Cumulative Probability options 13-17  
currency symbols 7-10  
**Cut** command 4-6, A-3  
cyclic dependencies 4-10, GL-3

## D

Date number format 7-10  
Decimal number format 10-1  
decimal points 7-10  
decision 15-6  
Decision variables 1-9, 5-2, 6-4, GL-3  
Decompose function 12-28  
default view 2-5  
Definition 19-6, 20-3, GL-3  
    editing 8-1–8-10  
    incomplete 19-9, A-6  
Definition button 1-4

**Definition** menu 8-10, A-5  
**Delete Columns** command 11-21, A-4  
**Delete Rows** command 11-16, 11-21, A-4  
 DeltaGraph options setup option 7-7  
 dependencies 1-4  
     conditional 15-6  
     cyclic 4-10  
     with the `Dynamic()` function 17-9  
 Description 19-6, 20-2, GL-3  
 Determ variables 1-11, GL-3  
`Determinant()` function 12-29  
 deterministic conditional tables 15-10–15-12, GL-3  
     defining 15-10  
 deterministic value GL-3  
 deterministic variables, *see* Determ variables  
`Determtable()` function 15-10, 15-12  
 determtables, *see* deterministic conditional tables  
`DetermType` qualifier 20-3  
 diagram  
     adding graphics 9-11  
     changing size 6-14  
     *see also* Diagram window  
**Diagram** menu A-11  
 Diagram Style dialog box 6-11  
 Diagram window 1-3, 1-7, 1-12  
     background color 6-10  
     customizing 6-11  
     maximum number of 19-18  
     printing 1-18  
     resizing 1-16  
 diagrams, *see* influence diagrams  
 dimensions 11-1–11-3, 11-10, GL-4  
     adding to a probability table 15-5  
     adding to an array 11-22  
     innermost 12-4  
     outermost 12-4  
     removing from an array 11-19, 11-22  
 discrete probability distributions 13-2, GL-4  
     creating 15-1  
     with label values 15-5  
 discrete variables GL-4  
**Distribution** command A-7  
 Distribution library A-7  
 distribution, defining a variable as 13-6  
 domain 9-4, 15-1, 15-3, 15-4, 15-7, 19-6, GL-4  
 dot product 12-30  
**Duplicate Nodes** command 4-6, A-3  
`Dydx()` function 16-9  
 dynamic arrows 17-10

    showing or hiding 6-12  
 dynamic models 6-4  
 dynamic simulation 17-1–17-11  
 dynamic variables GL-4  
`Dynamic()` function 17-2–17-11

## E

**Edit Definition** command 8-1, A-6  
**Edit Icon** command 9-10, A-12  
**Edit** menu A-3  
 Edit Table GL-4  
 Edit Table window 11-4, 11-20, 18-1  
     opening 11-20  
**Edit Time** command 17-1, A-6  
 Edit tool GL-4  
 Edit tool button 1-4  
`Elasticity()` function 16-10  
 electronic mail 18-8  
 error factor 14-4  
 error messages, *see* errors  
 errors E-1–E-4  
 evaluation errors E-3  
`Exp()` function 10-7  
 Exponent number format 7-10, 10-1  
**Export** command 18-4, A-2  
 export data format 18-9  
 Exporting data 18-1–18-12  
`Expr` keyword 4-10  
`Expr` qualifier 20-3  
**Expression** popup menu 8-5, 11-17  
 expression types 8-5, 8-6, GL-5  
 expressions 10-1–10-8, GL-5  
     syntax of 10-6

## F

`Factorial()` function 10-7  
**False** system variable 10-4, A-9  
 fatal errors E-4  
 File Info 19-6, GL-5  
**File** menu A-1  
 filed libraries 19-9, GL-5  
 Filed Library class 4-18  
 Filed Module class 4-18  
 filed modules 19-9, GL-5  
 fill color, displaying 6-13  
**Find** command 19-5, A-5

Find dialog box 19-5  
**Find Next** command 19-5, A-5  
**Find Selection** command 19-5, A-5  
Fixed Point number format 7-10  
fonts  
    in graphs 7-6  
    in nodes 6-12  
For...Do... function 12-31  
Form class 4-18  
form modules 9-7  
fractiles D-3, GL-5  
Fractiles() function 14-3  
frame 9-12  
Frequency() function 16-3  
FunctionOf() 4-10, 19-11  
functions 1-11, A-8  
    attributes of 20-2  
    creating 20-1–20-6

## G

general variable 1-10, GL-5  
**Get Info** command 4-22, C-3  
GetFract() function 16-4  
**Graph Setup** command 7-1, A-11  
Graph Setup dialog box 7-1  
Graph View button 2-7  
Graphing Tool setup option 7-2  
graphs  
    background grid 7-5  
    displaying 2-7  
    exporting 18-3  
    fonts 7-6  
    formatting 7-1  
    frames around 7-5  
    line style 7-6  
    origin 7-4  
    printing 1-18  
    ranges for 2-8  
    scatter 16-13  
    styles 2-8  
    tick marks 7-5  
    X-Y 16-11  
grid 4-6

## H

hardware specifications B-1  
Help 19-6, GL-5  
    *see also* Balloon Help

## I

Icon window 9-9  
icons 9-9  
Ident(I=U) function 12-21  
Ident(Time-n) function 17-4  
identifiers 8-3, 19-6, 20-2, GL-6  
    changing 4-20  
    naming E-3  
If...then...else... operator 10-6, 11-26  
Ifall...then...else... operator 11-28  
**Import** command 18-3, A-2  
import data format 18-9  
importance analysis 16-15, GL-6  
Importing data 18-1–18-12  
Index button 11-22  
index selection area 2-3, GL-6  
index variables, *see* indexes  
indexes 1-10, 11-1–11-3, 11-10, 11-17, GL-6  
    adding to a probability table 15-5  
    adding to an array 11-22  
    changing on arrays 12-9  
    column 2-4  
    creating 11-10, 11-19  
    interchanging 2-3  
    key 2-4  
    removing from an array 11-19, 11-22  
    row 2-4  
    selecting for tables 11-22  
    showing or hiding arrows 6-11, 11-4  
    x-axis 2-4  
Indexes dialog box 11-18  
IndexType qualifier 20-3  
INF 10-2  
infinity 10-2  
influence arrows, *see* arrows  
influence diagrams 1-7, 5-2, GL-7  
    copying 18-2  
    editing 4-3–4-6  
    guidelines for 6-1–6-16  
innermost dimension 12-4, GL-7  
input arrowhead 1-8, GL-7  
input nodes 1-6, 9-2–9-8

inputs 19-6, GL-7  
   displaying arrows 6-13  
   examining 1-13  
   remote 1-8  
**Inputs** popup menu 1-9, 8-2  
**Insert Columns** command 11-21, A-4  
**Insert Rows** command 11-15, 11-21, A-4  
 Integer number format 7-10, 10-1  
 Integrate() function 12-18  
 interpolation functions 12-23  
 Invert() function 12-30  
 iteration 12-33

## K

key 7-4, GL-7  
 key icon 1-14  
 key index 2-4  
 Knuth random number generator 13-15  
 kurtosis GL-7  
 kurtosis() function 16-5

## L

L'Ecuyer random number generator 13-15  
 labels, displaying 6-13  
 Last Saved 19-6, GL-7  
 lexical errors E-2  
 libraries GL-7  
   adding to a model 19-11  
   creating 20-6  
   filed 19-9  
   removing from a model 19-11  
   using 20-7  
 Library class 4-18  
**Library** popup menu 8-8  
 Linearinterp() function 12-24  
 list of labels 11-14, GL-8  
 lists 9-2, 11-10–11-16, GL-8  
   creating 3-2, 11-10  
   editing 11-15  
 Ln() function 10-7  
 logical operators 10-5, 11-25  
 logical values 10-4  
 logical variables 13-2  
 Lognormal() function 14-4  
 Logten() function 10-7

## M

Macintosh system software 7-10, B-1  
**Make Alias** command 4-12, A-5  
**Make Importance** command 16-15, A-5  
**Make Input Node** command 9-3, A-5  
**Make Output Node** command 9-6, A-5  
**Math** command A-6  
 math functions 10-7, 11-24  
 Math library A-6  
 matrices GL-8  
 matrix functions 12-28–12-30  
 Max() function 12-14  
 mean GL-8  
**Mean Value** command 2-10, A-10  
 Mean() function 16-5  
 median GL-8  
 Median Latin Hypercube sampling method 13-13  
 memory  
   increasing allocation C-2  
   Memory Usage window C-1  
   requirements B-1  
   usage A-13  
 menus A-1–A-13  
 mid value 1-16, GL-8  
**Mid Value** command 2-10, A-10  
 Mid() function 16-5  
 Min() function 12-14  
 Minimal Standard random number generator 13-15  
 mode 13-4, GL-8  
 Model class 4-18  
 models  
   building 5-1  
   changes when an arrow is created 4-9  
   changes when an arrow is removed 4-9  
   closing 1-2  
   combining 19-13  
   creating 4-1  
   defined 1-1  
   defining 4-2  
   documentation 5-9  
   dynamic 4-10, 6-4  
   editing 4-3–4-17, 19-5  
   expansion 5-10  
   integrated 19-13  
   modular 19-14  
   navigating 19-3  
   opening 1-1  
   saving 4-1, 19-11

- switching 1-2
- testing 5-7
- Module class 4-18
- module hierarchy 19-1, GL-9
  - organizing 6-8
- modules 1-10, GL-8
  - attributes of 1-15
  - filed 19-9
  - showing or hiding arrows 6-11
- Monte Carlo sampling method 13-15
- Move Into Parent** command A-12
- multidimensional array 11-1
- multimodal distribution GL-9

## N

- naming errors E-3
- NAN 10-3
- natural cubic spline 12-24
- New Model** command 4-1, A-1
- node palette 4-3
- Node Style dialog box 4-15, 6-12
- nodes GL-9
  - adding icons 9-9, 9-11, 9-12
  - aligning 4-6, 6-5
  - and arrows 4-7
  - arranging 6-4
  - changing size 4-5
  - color 6-10
  - creating 4-4
  - customizing 6-12
  - default size 6-12
  - deleting 4-6
  - deselecting 1-11
  - displaying arrows to/from 6-13
  - duplicating 4-6
  - editing title 4-4
  - fill colors 6-13
  - grouping related 6-8
  - in fonts 6-12
  - labels 6-2, 6-13
  - moving 4-5
  - selecting 1-11, 4-4
  - size 6-3
  - types 1-9
  - undefined 4-21, 6-15
- Normal() function 14-6
- Normalize() function 12-18
- Number Format** command 7-9, A-11

- number formats 7-9, 10-1
- Numeric qualifier 20-3

## O

- Object button 1-3
- Object Finder dialog box 8-7, GL-9
- Object** menu A-4
- Object window 1-3, 1-8, 1-12, 4-20, GL-9
  - maximum number of 19-18
  - opening 1-12
  - printing 1-18
- Objective variables 1-10, 5-3, 6-4, GL-9
- objects GL-9
- Open Model** command 1-2, A-1
- operators GL-9
- Operators** command A-8
- Operators library A-8
- order of precedence 10-6
- outermost dimension 12-4, GL-9
- Outline button 1-3
- Outline window 1-3, 19-2, GL-10
  - printing 1-18
- output arrowhead GL-10
- output nodes 1-6, 9-5
- outputs 19-6, GL-10
  - displaying arrows 6-13
  - examining 1-13
  - remote 1-8

## P

- page breaks A-13
- Page Setup** command 1-18, A-2
- palette
  - node 4-3
  - tool 1-3
- Parameters 19-6, 20-2, GL-10
- parametric analysis 3-1
- parent diagram 1-8, 1-13, GL-10
- Parent Diagram button 1-3, 1-8
- Paste** command 4-6, 11-22, 18-1, A-3
- Paste Identifier** command 8-7, A-6
- Percent number format 7-10
- percentile 16-4, GL-10
- Pi* system variable A-9
- picture nodes 9-11
- popup menu

creating 9-3  
 using 1-6  
 Positive qualifier 20-3  
 precedence, order of 10-6  
 precision 10-3, B-2  
**Preferences** command 4-19, 19-18, A-4  
 Preferences dialog box 1-4, 4-19, 8-14, 19-1, 19-18  
**Print** command 1-18, 18-7, A-2  
**Print Report** command 1-19, A-2  
 Prob qualifier 20-3  
 Prob Table button 15-2  
 probability bands GL-10  
   setting 13-17  
**Probability Bands** command 2-11, A-10  
**Probability Density** command 2-12, A-10  
 probability density function GL-10  
 Probability Density option 13-17  
 probability distribution functions 14-1–14-9  
 probability distributions 14-1–14-9, GL-11  
   beta 14-1  
   choosing 13-2–13-5  
   computing 13-9  
   continuous 13-2  
   discrete 13-2, 15-1  
   lognormal 14-4  
   normal 14-6  
   triangular 14-7  
   truncating 14-9  
   uniform 14-8  
**Probability Mass** command 2-12, A-10  
 probability mass function GL-11  
**Probability Table** command 15-2  
 probability tables 15-1–15-6, GL-11  
   conditional 15-4  
 Probability() function 16-6  
 Probbands() function 16-6  
 Probdist() function 14-6  
 Probtale() function 15-4  
 probtables, *see* probability tables  
 Probvalue 19-6, GL-11  
 Product() function 12-14  
 publish and subscribe 18-4  
**Publishing** commands 18-4, A-4

## Q

quantile 16-4  
**Quit** command 1-2, A-2

## R

Random Latin Hypercube sampling method 13-14  
 random number generators  
   Knuth 13-15  
   L'Ecuyer 13-15  
   Minimal Standard 13-15  
 random number methods 13-15  
 random seed 13-16  
 Rank() function 12-19  
 Rankcorrel() function 16-6  
 Recomputing results 2-5  
 reducing functions 12-11–12-16, GL-11  
 remote  
   inputs 1-8  
   outputs 1-8  
   variables 1-8  
 resampling 17-11  
 resize box GL-12  
**Resize Centered** command 4-5, 6-5, A-12  
 Result button 1-4, 1-17, 2-3  
**Result** command 2-2  
**Result** menu A-10  
 Result Tool palette 2-3  
 result view GL-12  
   *see also* Result window GL-12  
 Result window 1-4, 2-1–2-5  
   default view 2-5, 4-21  
   graph view 2-7  
   maximum number of 4-20, 19-18  
   opening 2-2  
   table view 2-5  
 results  
   comparing 2-14  
   printing 1-18  
   recomputing 2-5  
   viewing 1-9, 2-1–2-14  
 Round() function 10-8  
 row index 2-4  
**Run** system variable 11-20, 13-9, 16-7, A-9

## S

Samp qualifier 20-3  
 sample GL-12  
**Sample** command 2-13, A-11  
 sample size  
   selecting D-1  
   setting 13-11



`sample()` function 16-7  
**Samplesize** system variable 13-12, 16-7, A-9  
sampling methods GL-12  
    Median Latin Hypercube 13-13  
    Monte Carlo 13-15  
    Random Latin Hypercube 13-14  
    selecting 13-13  
**Save a Copy In** command 19-11, A-2  
**Save As** command 4-1, 19-11, A-2  
**Save** command 4-1, 19-11, A-2  
scalar 11-1, GL-12  
Scalar qualifier 20-3  
scatter plots 16-13, GL-12  
screen captures 6-15  
`sdeviation()` function 16-7  
**Select All** command 6-6, A-3  
**Self** 8-13, 11-18, 15-3, 17-3, GL-12  
sensitivity analysis 16-9–16-11, GL-12  
Sequence option 11-12  
`Sequence()` function 12-5  
**Set Diagram Size** command 6-14, A-12  
**Set Diagram Style...** command 6-11, A-11  
**Set Node Style** command 6-12, A-12  
shells, stand alone 19-16  
**Show By Identifier** command 8-3, A-5  
**Show Clipboard** command A-13  
**Show Color Palette** command 6-10, A-12  
**Show Invalid Variables** command 19-9, A-6  
**Show Memory Usage** command A-13, C-1  
show module hierarchy 19-1  
**Show Page Breaks** command A-13  
**Show Result** command A-10  
**Show With Values** command 1-17, 19-4, A-5  
`Sin()` function 10-8  
size box 1-16  
`Size()` function 12-27  
skewed distributions 13-5  
skewness GL-12  
`Skewness()` function 16-8  
slice GL-13  
`Slice()` function 12-21  
software specifications B-1  
`Sortindex()` function 12-27  
**Special** command A-7  
Special library 12-2, A-7  
specifications B-1  
`Sqr()` function 10-8  
`Sqrt()` function 10-8  
standard deviation GL-13

stationery 4-22, GL-13  
**Statistical** command A-8  
Statistical library A-8  
**Statistics** command 2-11, A-10  
`Statistics()` function 16-8  
Statistics, setting 13-16  
`Stepinterp ()` function 12-25  
string, *see* text values  
`Subindex()` function 12-15  
**Subscribe to** menu command 18-5  
**Subscript** function 12-22  
`Subset()` function 12-28  
Suffix number format 7-10, 10-1  
suffixes GL-13  
`Sum()` function 12-16  
symmetrical distributions 13-5, GL-13  
syntax errors E-2  
system functions GL-13  
system variables GL-13  
**System Variables** submenu A-8

## T

table lookup 12-25  
Table View button 2-3, 2-5  
`Table()` function 12-7, 12-10  
tables 11-1–11-28, GL-13  
    creating 11-16–11-19, 12-7  
    deterministic conditional 15-10–15-12  
    displaying 2-5  
    editing 11-20–11-22  
    fitting windows to 6-16  
    printing 1-18  
    saving 11-22  
    selecting cells 11-21  
    selecting indexes for 11-22  
tails GL-13  
text values 10-3  
TextTable keyword 18-9  
thousands separators 7-10  
**Time** system variable 11-20, 17-1, 17-4–17-7, A-9  
Title 1-12, 6-2, 19-6, 20-2, GL-13  
    editing 4-4, 4-15, 4-18  
transforming functions 12-16–12-20  
`Transpose()` function 12-30  
`Triangular()` function 14-7  
**True** system variable 10-4, A-9  
`Truncate()` function 14-9  
**Turn Grid Off** command 4-6, A-12

## U

uncertainty factor 14-4  
**Uncertainty Options** command 13-11  
 Uncertainty Sample option 13-11  
**Uncertainty Setup** command A-11  
 Uncertainty Setup dialog box 13-11  
**Uncertainty View** popup menu 2-3, 2-9  
 Uncumulate() function 12-19  
 undefined nodes 4-21  
**Undo** command 4-16, A-3  
 Uniform() function 14-8  
 Units 19-6, 20-2, GL-14  
 user libraries 20-6, A-5  
 user-defined functions GL-14  
 Using...Do... function 12-33

## V

values 19-6  
   checking bounds 4-21, 8-12–8-13  
   deterministic GL-3  
   showing 1-16  
   uncertain GL-13  
 variables 1-10  
   attributes of 1-15  
   automatic renaming 4-20  
   class checking 4-21  
   classes 1-9  
   finding 19-5  
   influencing each other 1-7  
   invalid 19-9  
   public 19-14  
   remote 1-8, GL-11  
   showing dependencies among 4-7  
   types 1-9  
 variance GL-14  
 Variance() function 16-8  
 Vector qualifier 20-4

## W

Warning icon 8-3, E-1  
 warnings, *see* errors  
 Whatif() function 16-10  
**Window** menu A-13  
 windows  
   fitting to tables 6-16

managing 19-18  
 numbers of 4-19  
*see also* Diagram window, Object window, Outline window

## X

x-axis index 2-4  
 XY button 16-11, 16-13  
 X-Y results 16-11

## Z

zoom box GL-14