IBM® SPSS® Amos™ 32

Programming Reference Guide

Table of Contents

1 What's new in Amos 32

There is a new Data focus check box.

Parameter values that are less than 1 in absolute value are now displayed on path diagrams without a leading zero, for example ".123" rather than "0.123".

The Stan IDE has been removed, but Export to Stan remains.

2 What was new in Amos 31

There is now a menu item that lets you view or analyze your data file using your system's default app. For example, if your Amos dataset is in a text file, the new menu item shows the data in a text editor. For more information, see File→Run Default App.

3 What was new in Amos 30

Amos 30 provides support for Python programming. The %amosexamples% folder now contains Python programs for the User's Guide examples. The Write a program dialog now contains an option for generating a Python program that fits the model specified by the path diagram.

4 Programming with Amos

You can extend the capabilities of Amos in two ways:

You can write programs that use Amos as a component. Your programs can make use of the Amos classes to incorporate the results of a structural modeling analysis into some larger data analysis project. (See Writing a Main Program that Uses Amos 28.)

You can add functionality to Amos by creating classes containing methods that are called by Amos. (See Writing Classes that are Used by Amos 31.)

4.1 Getting Started

Before you start writing programs that use Amos (or are used by Amos), it is a good idea to get some experience doing structural equation modeling with Amos. One way to do this is by working through the tutorial and some of the examples in the *User's Guide*.

The *User's Guide* contains many examples of programs that use Amos. The *Programming Reference Guide* contains many more. Although almost all of the examples in the Amos documentation use Visual Basic, C# is just as easy to use for Amos programming.

The Amos programming examples in the *Programming Reference Guide* and in the online help are installed with Amos. By default they are installed in the folder

C:\Program Files\IBM\SPSS\Amos\32\Programming

Most of the examples in the Amos documentation are small, intended to demonstrate the use of one or two methods or properties at a time. Some larger examples, such as the following, perform nontrivial tasks.

- Use the Amos Graphics classes to change the appearance of latent variables 492
- Use the Amos Graphics classes to resize all rectangles 499
- Use the Amos Graphics classes to draw a path diagram 494
- Use the Amos Graphics classes to calculate a new fit measure 490
- Use the Amos Graphics classes to draw double-headed arrows 496
- Use the Amos Graphics classes to name unobserved variables 497

For even larger examples of Amos programming, see the folder

C:\Program Files\IBM\SPSS\Amos\32\Programming\Plugins

which contains the source code for the plugins on the **Plugins** menu of Amos Graphics.

4.2 Programming Tools

Amos comes with a built-in program editor that can be used for writing and executing Amos programs.

- To write a main program, start the built-in program editor by opening the Windows **Start** menu and searching for **IBM SPSS Amos 32 Program Editor**.
- To write a plugin, start the built-in program editor from the Amos Graphics menu by clicking **Plugins→Plugins**.
- To write a class to compute Bayesian custom estimands (see Example 29 in the User's Guide), start the built-in program editor from the **Bayesian SEM** menu by clicking **View → Custom estimands**.
- To create a (non-Bayesian) user-defined estimand click the status bar at the bottom of the Amos Graphics window and select **Define new estimands** from the menu that pops up.

For main programs and plugins (but not for custom estimands) you can use the development tool of your choice. For examples using Visual Studio 2015, see

- Writing a Main Program with Visual Studio 2015 29
- Writing a Plugin with Visual Studio 2015 40

4.3 Writing a Main Program that Uses the AmosEngine Class

You can write programs in Visual Basic or C# that make use of the AmosEngine class. In this way you can specify and fit a model by writing Visual Basic or C# code, avoiding the use of Amos Graphics entirely. You can also write programs that use structural equation modeling as part of a larger data analysis project. Examples 1 through 21 in the *User's Guide* show how to write programs that specify and fit models using the AmosEngine class.

4.3.1 Using the Built-in Code Editor

The *User's Guide* contains a step-by-step tutorial on using Amos's built-in editor to write a main program. The tutorial is in the **Modeling in VB.NET** section of Example 1 in the *User's Guide*.

4.3.1.1 Open

Open an existing Amos program.

4.3.1.2 Print preview

Display onscreen a preview of the Amos program, showing how the program will look when it is printed after you press the **Print** button.

4.3.1.3 Print

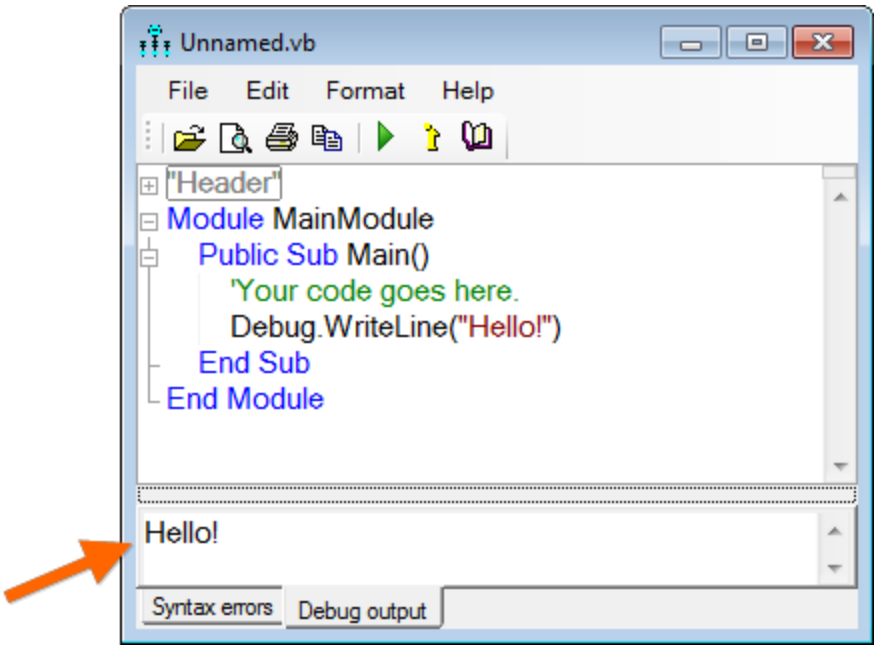
Print the Amos program. Press the **Print preview** button to see what the program will look like when it is printed.

4.3.1.4 Copy

Copy selected text to the clipboard.

4.3.1.5 Run

Run the Amos program. Any syntax errors will be displayed on the **Syntax errors** tab at the bottom of the window. Output written using the System.Diagnostics.Debug class will be displayed on the **Debug output** tab.



4.3.2 Using Visual Studio 2015

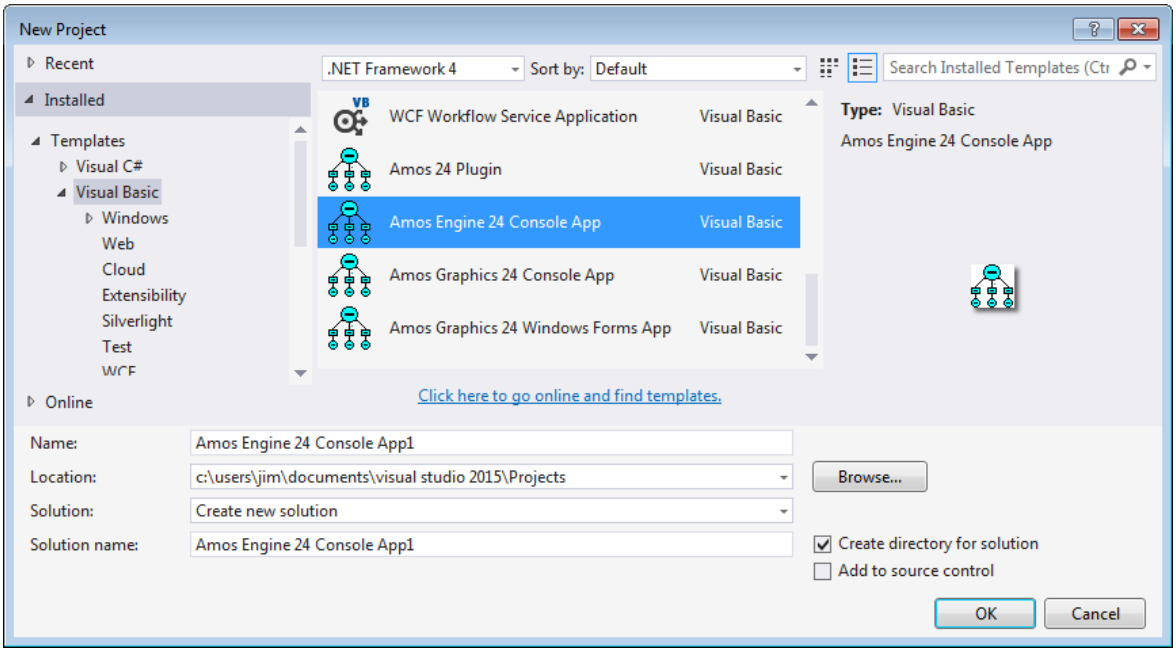
This section shows how to use Visual Studio 2015 to fit the model of Example 1 in the *User's Guide*.

1. (You only have to do this once.) Install Visual Studio 2015. The free Visual Studio Community 2015 can be used.

2. (You only have to do this once.) Double-click the file Amos.VSIX in the Amos program folder. This installs Visual Studio templates that give you a head start in writing Amos programs.

3. Open Visual Studio and click File > New Project.

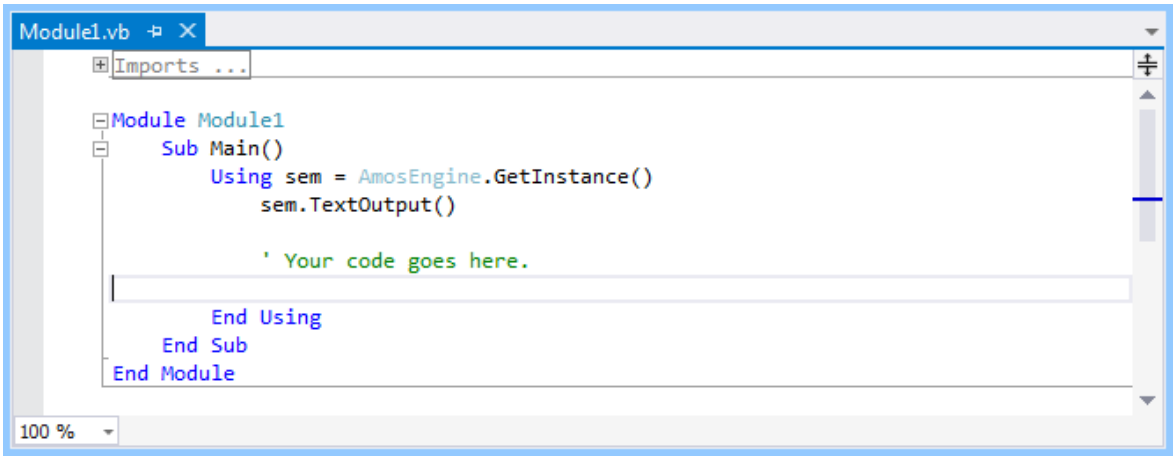
4. In the New Project dialog, select Visual Basic and then Amos Engine XX Console App, where XX is the version of Amos you want to use.



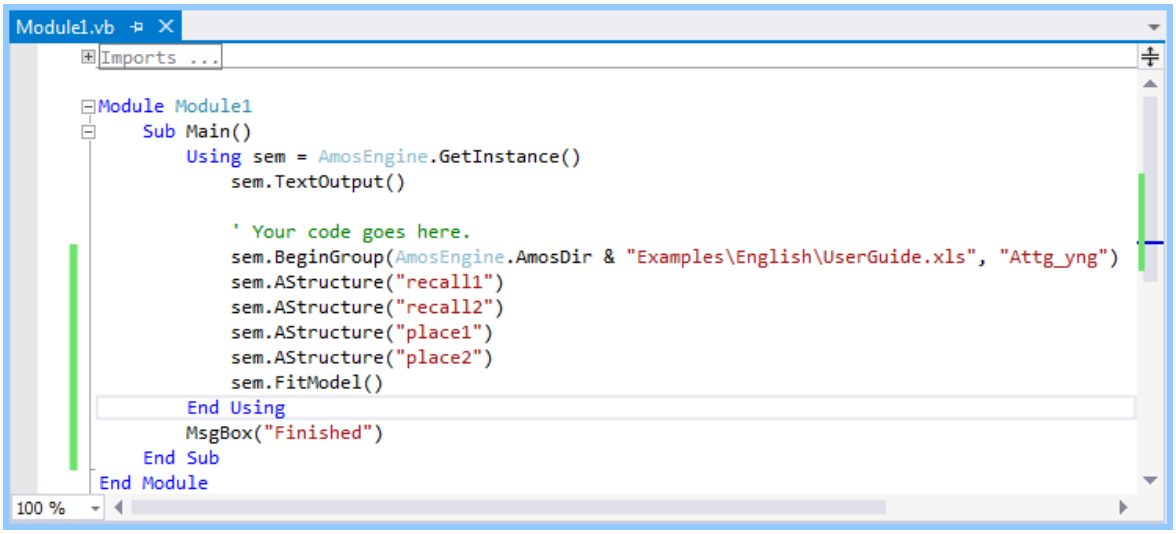
You can enter your own values for Name, Location and Solution name, or just accept the default values.

5. Click OK to close the New Project dialog.

6. View Module1 by clicking Module1 in Solution Explorer.



7. Enter the code for **Sub Main** as follows (just below the comment "Your code goes here".)



The *User's Guide* gives a line-by-line explanation of the code for Sub Main in the Modeling in VB.NET section of Example 1.

8. To run the program, click Debug > Start Debugging on the menu, or press the F5 key. After the program runs, the Amos Output window will open to display the program output.

4.4 Writing Classes that are Used by Amos

You can use Visual Basic or C# to create classes that Amos uses to perform analyses that it cannot perform "out of the box". There are ways in which you can enhance the capabilities of Amos by writing classes that contain methods that Amos calls.

You can write an Amos Graphics plugin. This is a class containing methods that Amos Graphics can call at key points during a structural equation modeling analysis. (See Writing a Plugin for Amos Graphics 31.)

When you are doing Bayesian estimation, you can estimate the posterior distribution of any function of the model parameters by creating a class that defines a *custom estimand*. (See Calculating Custom Estimands 43.)

4.4.1 Writing a Plugin for Amos Graphics

A plugin is a class that augments the capabilities of Amos Graphics. It contains methods that Amos Graphics can call at key points during an analysis. Plugins that you write have access to the Amos Graphics classes and to the AmosEngine class.

Amos comes with some pre-written plugins that appear on the Plugins menu. Source code for the pre-written plugins is in the folder %amosprogram%\Programming\Plugins

The following two sections each give a step-by-step demonstration of writing a plugin. The plugin that is created during the demonstration does not do any useful work. Its purpose is to show how to write a simple plugin that can respond to events that occur while Amos Graphics is running. After you go through

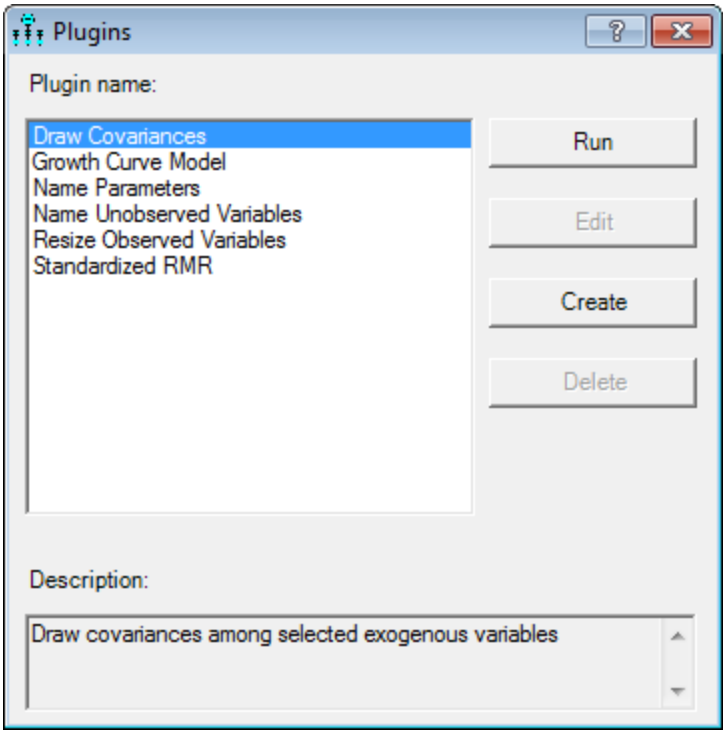
the steps of the demonstration, the plugin will appear as an item on the Amos Graphics Plugins menu with the name **A simple plugin**. When you click **A simple plugin**, a messagebox will display "Installing a simple plugin". After that, clicking any point on the path diagram will display another messagebox with the text "You released a mouse button.".

A mouse click is only one of many events 110 that your plugins can respond to.

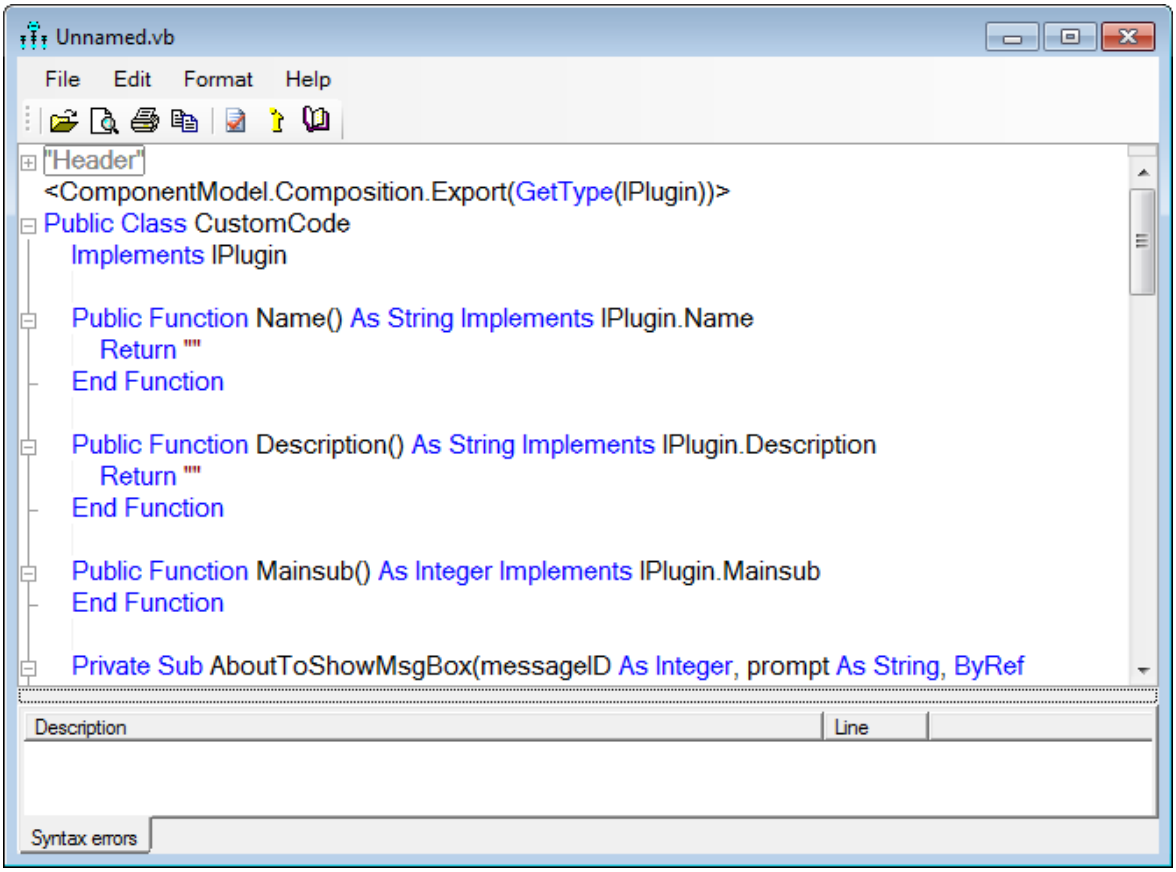
4.4.1.1 Writing a Plugin with the Built-in Code Editor

This section shows you how to use Amos's built-in editor to write a plugin in Visual Basic. Writing a plugin in C# is similar.

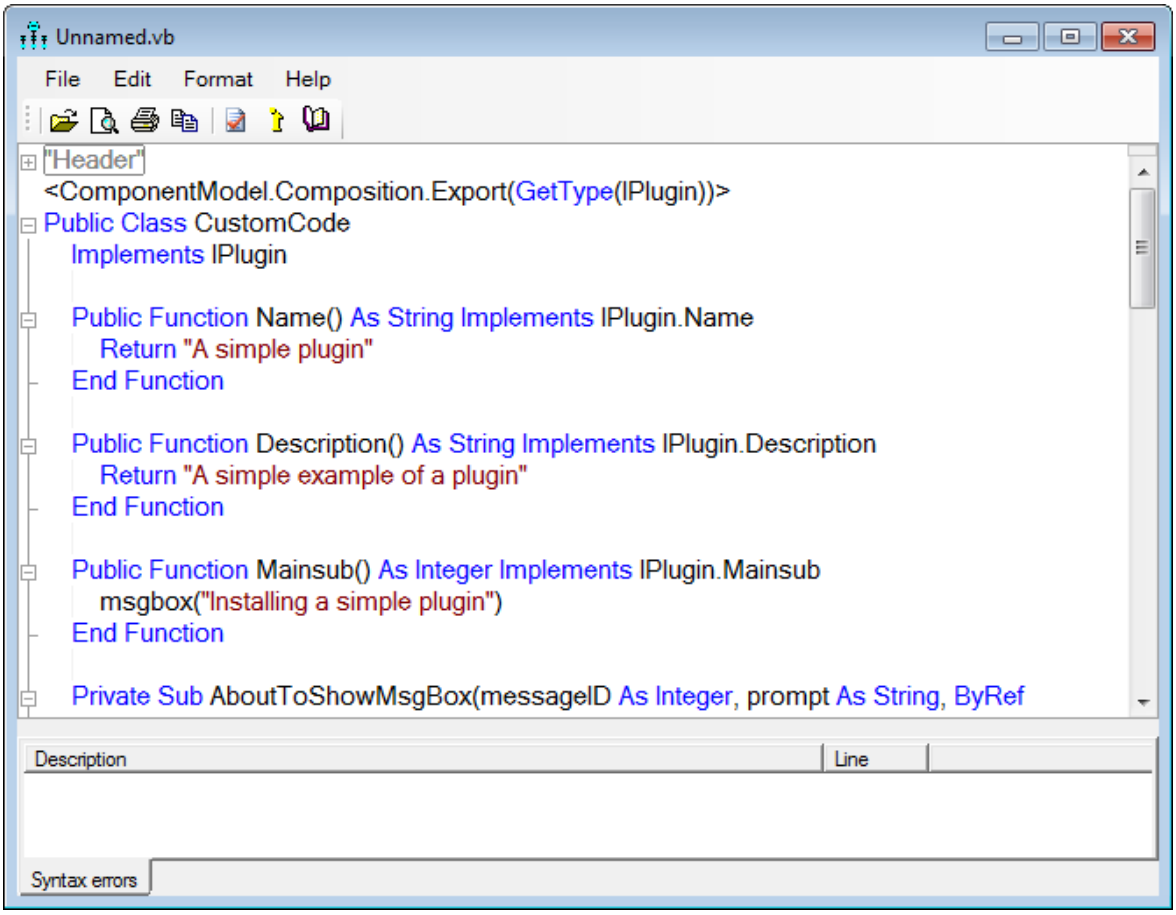
1. On the Amos Graphics menu, click Plugins -> Plugins. The Plugins dialog opens.



2. In the Plugins dialog, click Create. The plugin editor opens.



3. Add code for the Name, Description and Mainsub methods as shown in the following figure.

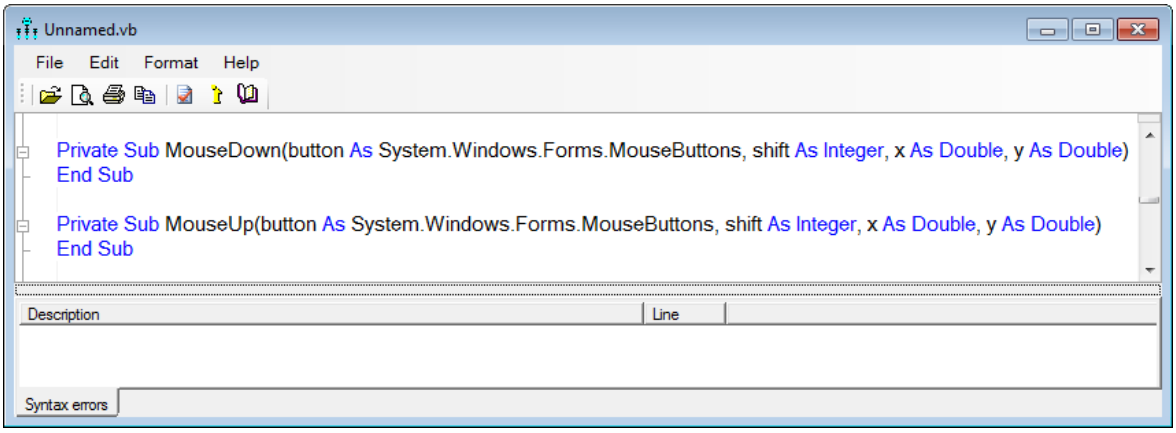


Amos calls the Name function to obtain the name of the plugin. The plugin's name will appear on the Amos Graphics Plugins menu. In the present example "A simple plugin" will be placed on the Plugins menu. It is OK if the Name function returns an empty string (""). In that case, the name of the file that contains the plugin will be placed on the Plugins menu.

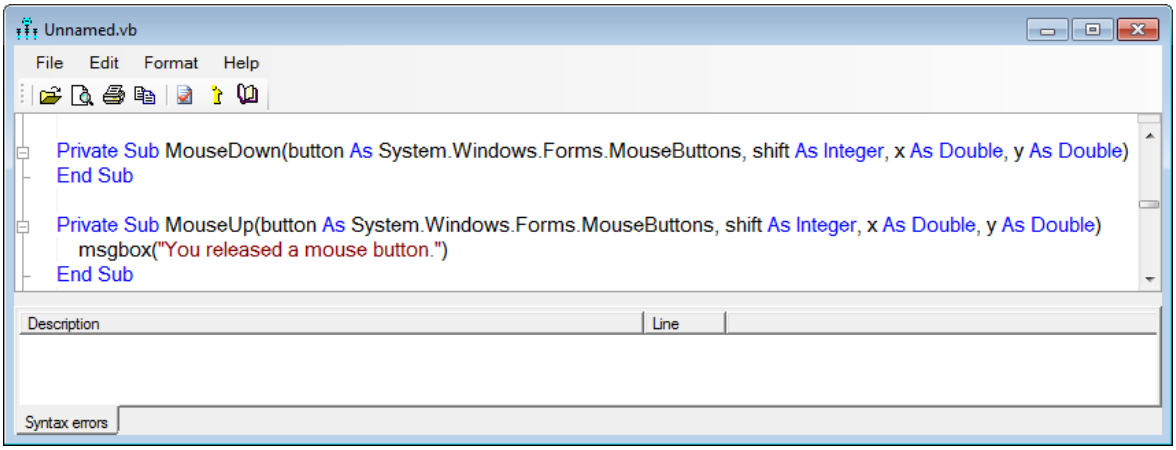
Amos calls the Description function to obtain a description of the plugin. A plugin's description is typically longer than its name. A plugin's description is displayed when the plugin is selected from the list in the Plugins dialog. It is OK for the Description function to return an empty string ("").

Amos calls the Mainsub function when you select A simple plugin from the Plugins menu. In this example, clicking A simple plugin on the Plugins menu will display the message "Installing a simple plugin".

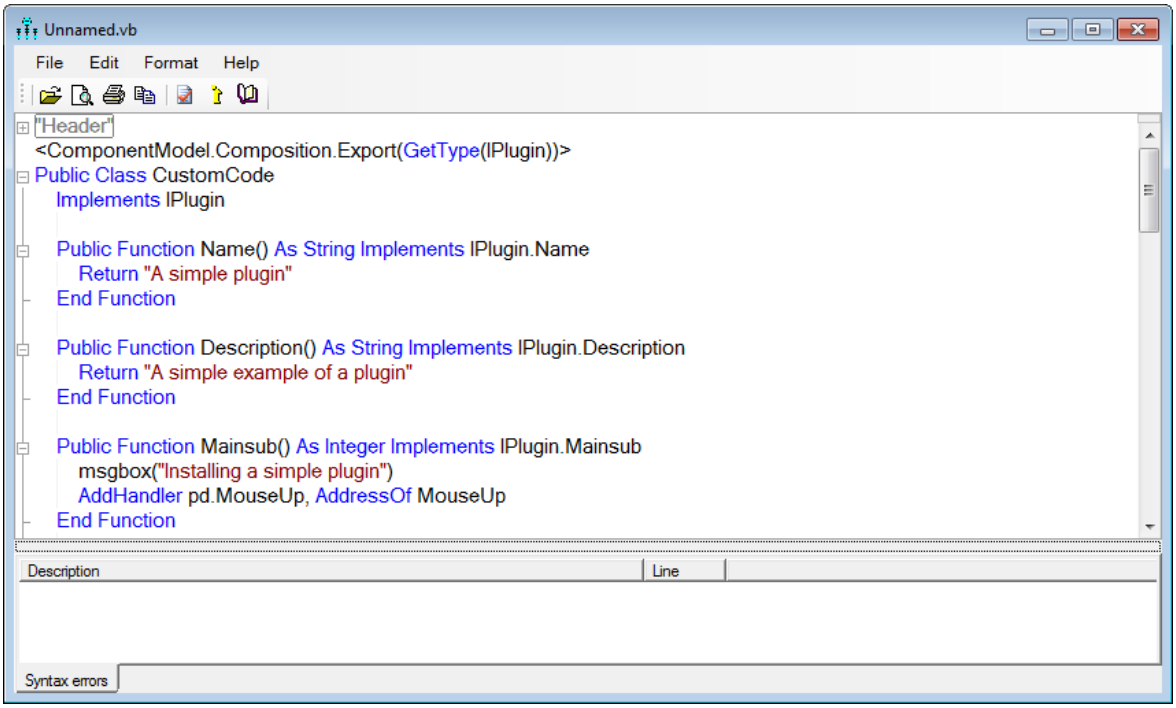
4. Scroll down through the source code until you locate the MouseUp method.



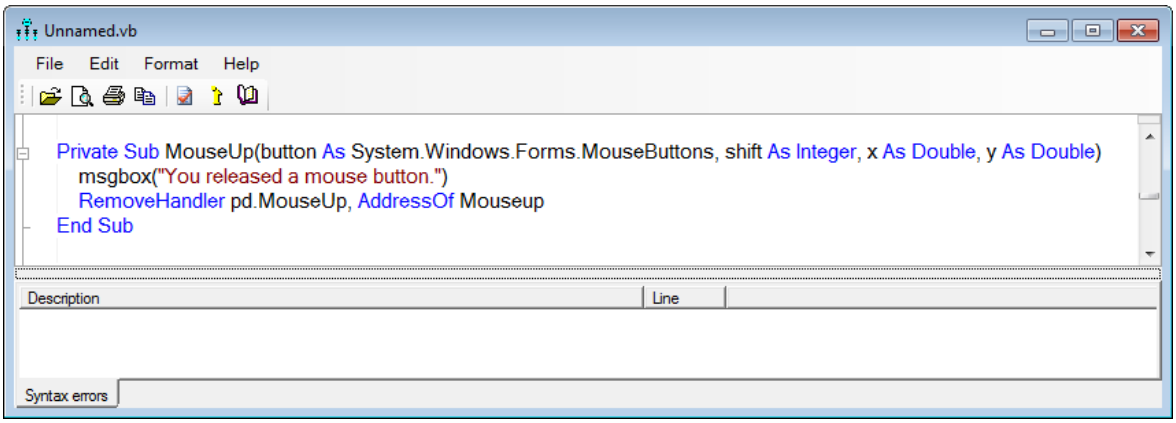
5. Insert code for the MouseUp method as follows.



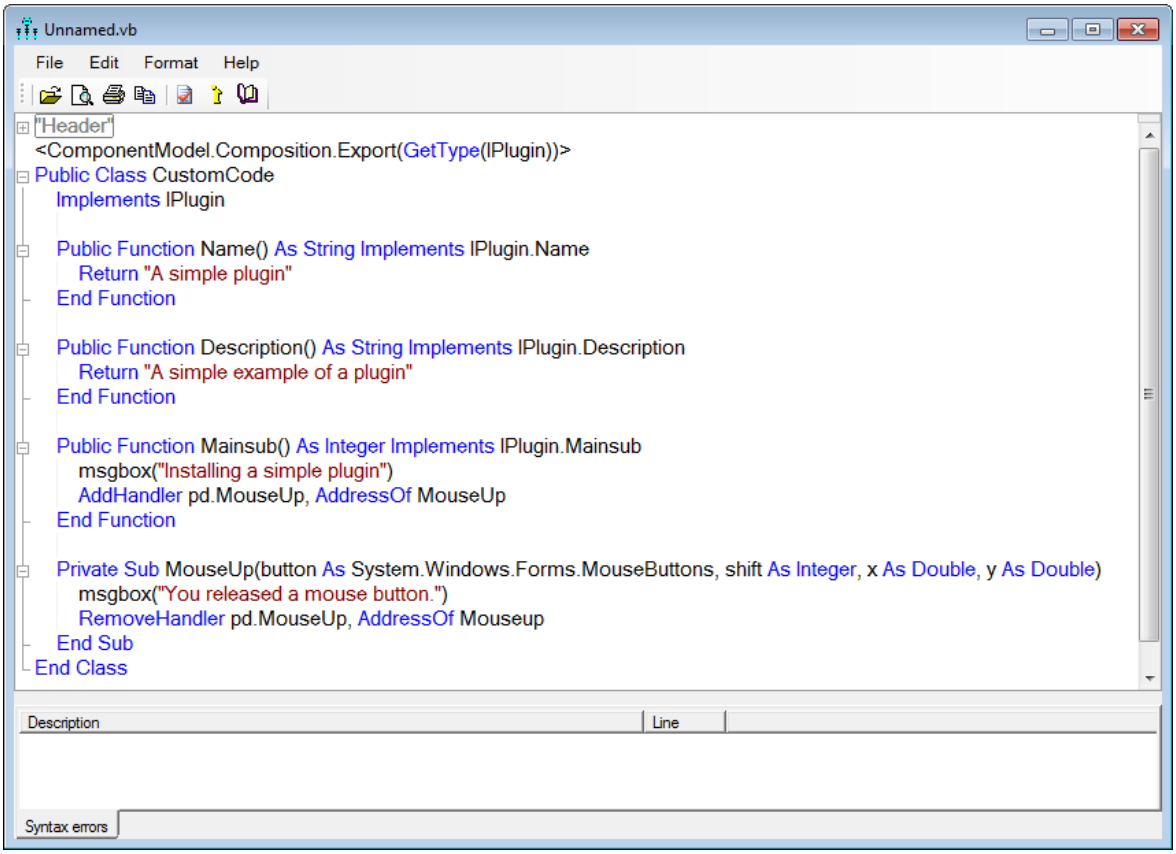
6. To let Amos know that it should call the MouseUp method when you release the mouse button, use the AddHandler keyword as shown in the following figure.



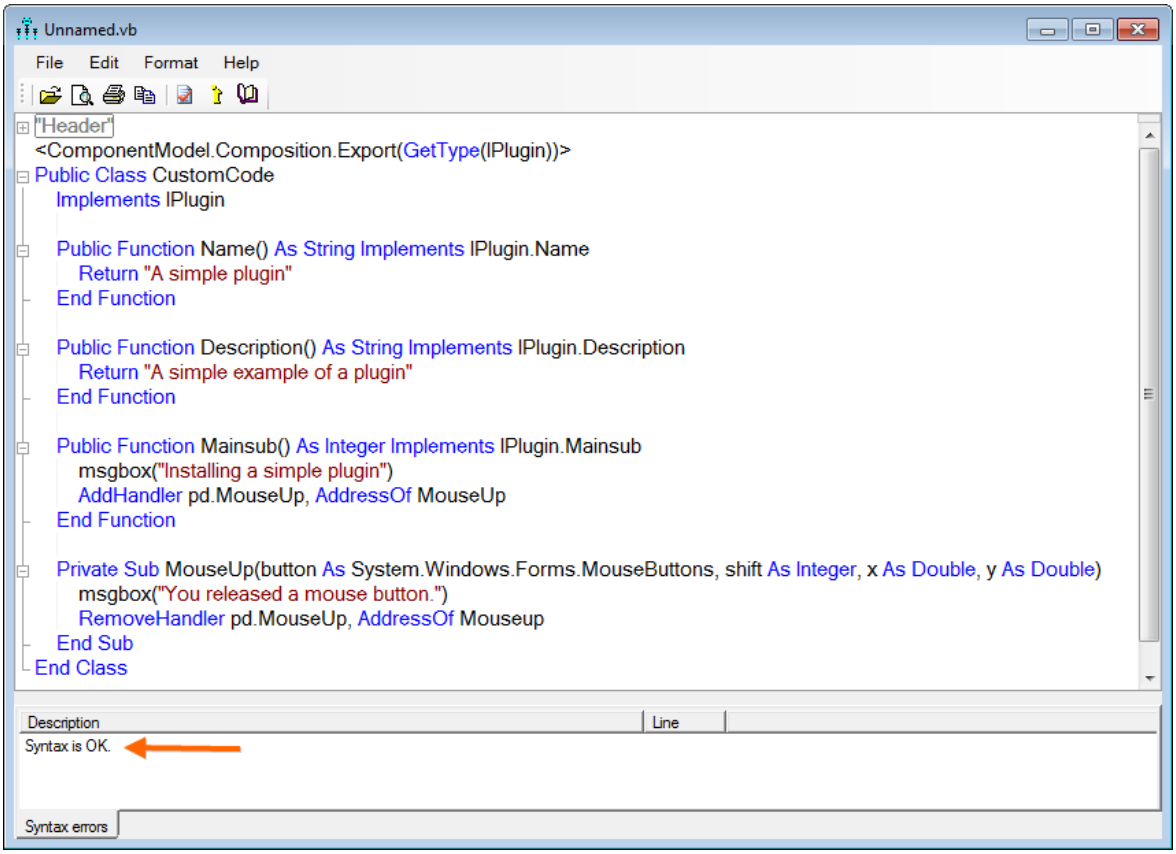
7. The plugin as specified up to this point displays a message every time you release the mouse button. You can stop this behavior only by closing Amos Graphics and re-opening it. However, you can cause the plugin to stop responding to mouse clicks by using the RemoveHandler keyword. Just as the AddHandler keyword causes Amos to call the Pd_MouseUp method for every mouse click, the RemoveHandler keyword can be used to cause Amos to stop calling the Pd_Mouseup method. In the following figure, RemoveHandler is executed immediately after the first "mouse up" message appears. As a result Amos calls the Pd_MouseUp method for a single mouse click only.



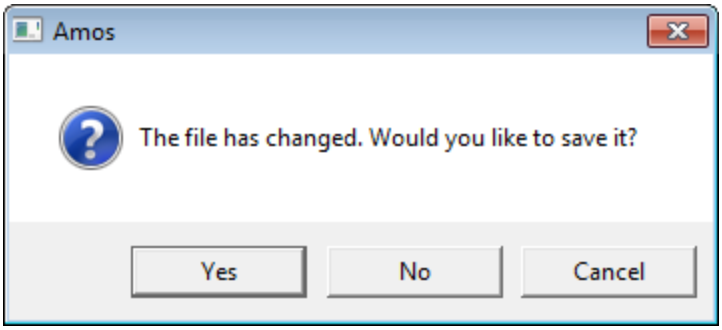
8. You may if you wish delete any method other than Name, Description, Mainsub and MouseUp. Amos always calls Name, Description and Mainsub, and so they must be present. In the present example, it is possible that MouseUp will be called because of the AddHandler line in Mainsub. The other methods will never be called and can be deleted, making the code for the plugin look like this:



9. Click the Check Syntax toolbar button to check for syntax errors. Any error messages will be displayed in the Syntax Errors area. If there are no syntax errors, you will see the message Syntax is OK.

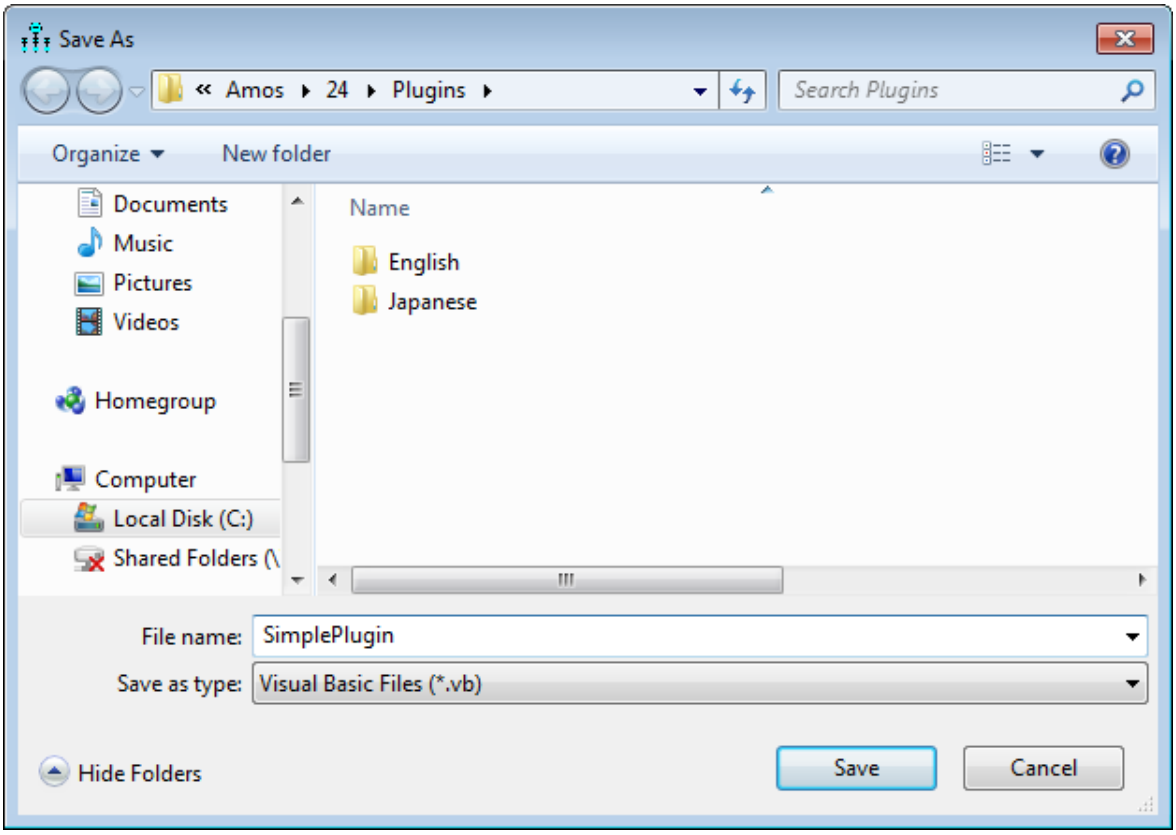


10. After correcting any typing mistakes so that pressing the Check Syntax toolbar button produces the message Syntax is OK., close the plugin editor window (by clicking the "x" in the upper-right corner). You will be asked if you want to save the plugin.



Click Yes.

11. The Save As dialog opens, with the Amos Plugins directory as the default directory. (Plugins must be saved in the Plugins directory.)



Enter a file name for the plugin and click Save. (The file name SimplePlugin was entered in the figure above.)

12. Close the Plugins dialog.
13. To test the new plugin, click A simple plugin on the Plugins menu.

4.4.1.1.1 Open

Open an existing plugin.

4.4.1.1.2 Print preview

Display onscreen a preview of the plugin as it will be printed if you press the **Print** button.

4.4.1.1.3 Print

Print the plugin. Press the **Print preview** button to see what the plugin will look like when it is printed.

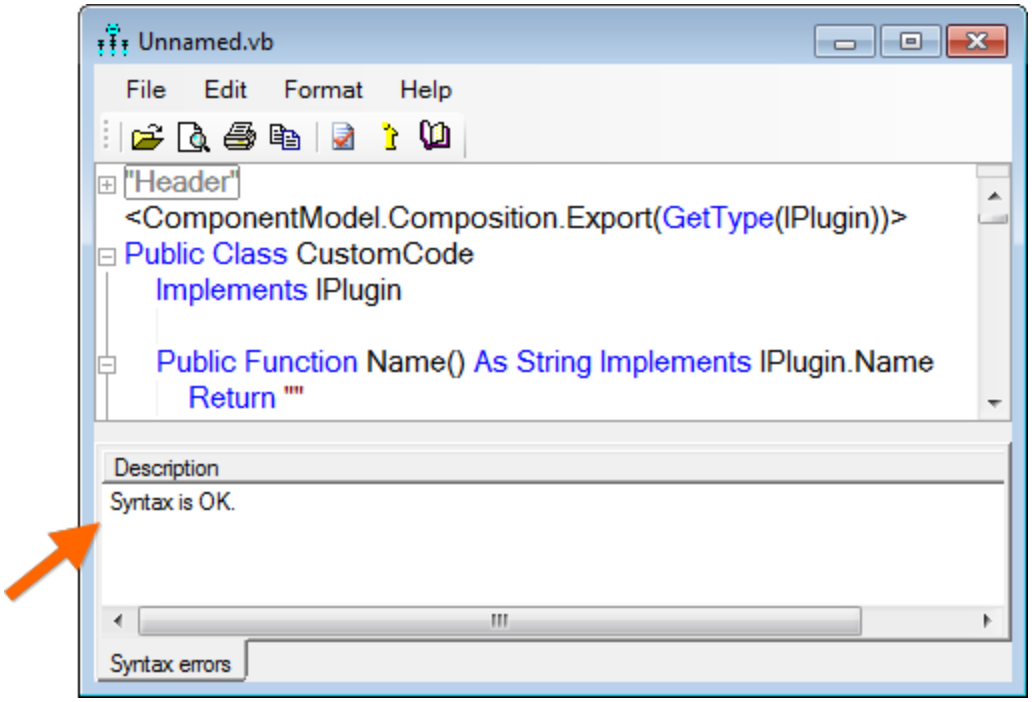
4.4.1.1.4 Copy

Copy selected text to the clipboard.

4.4.1.1.5 Check syntax

Check the plugin for syntax errors.

Any syntax errors will be displayed in the area labeled **Syntax errors** at the bottom of the window. If there are no syntax errors, the message **Syntax is OK** will be displayed.



4.4.1.2 Writing a Plugin with Visual Studio 2015

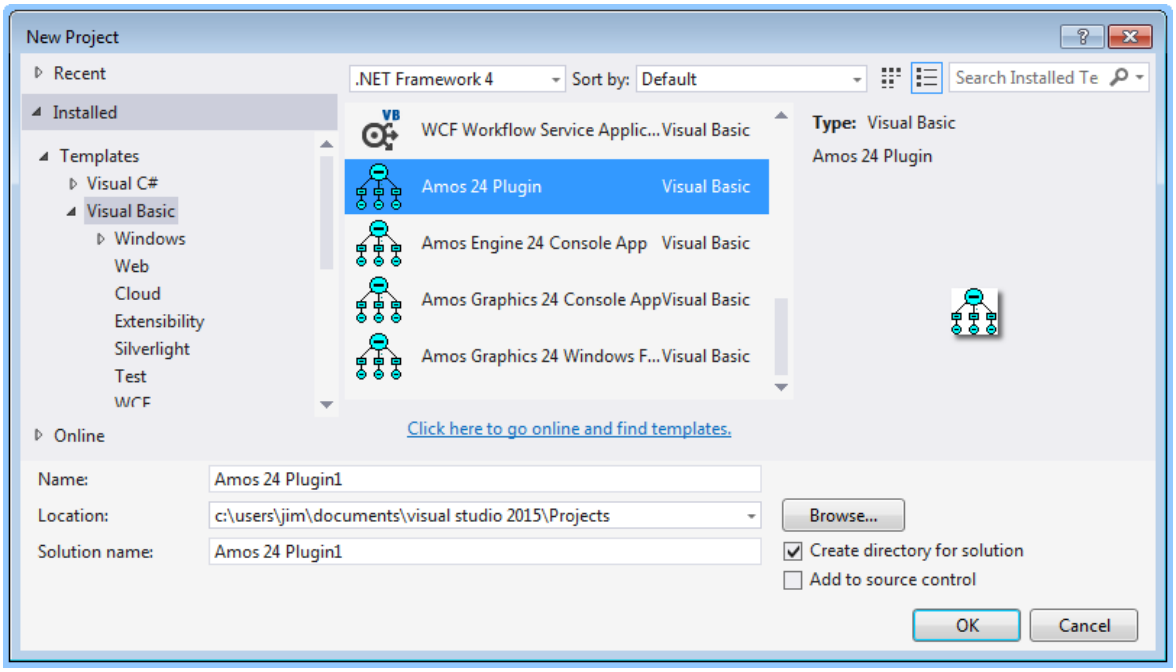
This section shows how to use Visual Studio 2015 to write an Amos plugin in Visual Basic. The plugin created here will draw two observed variables called ThingOne and ThingTwo.

1. (You only have to do this once.) Install Visual Studio 2015. You can use any edition of Visual Studio 2015, including the free Visual Studio Community 2015.

2. (You only have to do this once.) Double-click the file Amos.VSIX in the Amos program folder. This installs Visual Studio templates that give you a head start in writing Amos programs.

3. Open Visual Studio and click File > New Project.

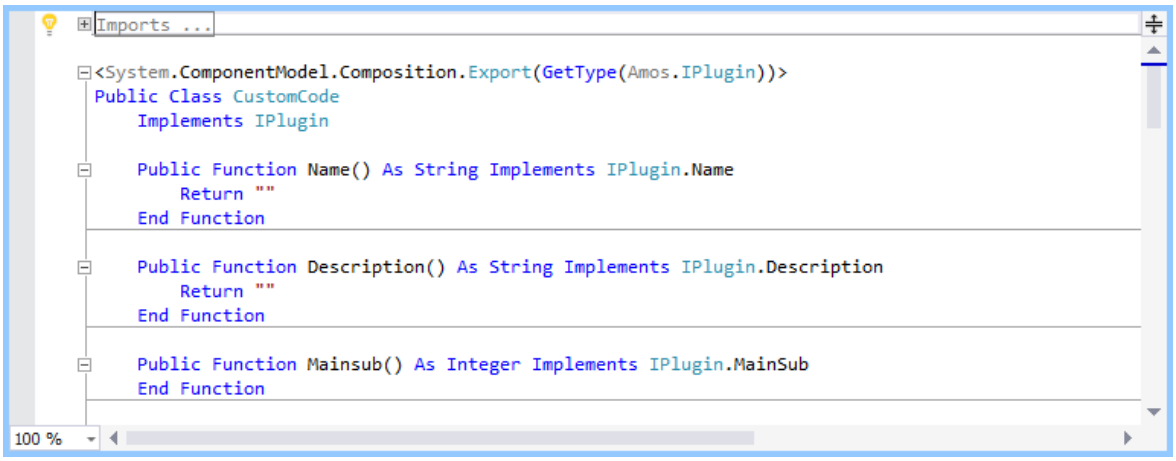
4. In the New Project dialog, select Visual Basic and then Amos XX Plugin, where XX is the version of Amos you want to use.



You can enter your own values for Name, Location and Solution name, or just accept the default values.

5. Click OK to close the New Project dialog.

6. View Module1 by clicking Module1 in Solution Explorer.



7. Add code to the functions Name, Description and MainSub as shown in the following figure.

```
Imports ...

<System.ComponentModel.Composition.Export(GetType(Amos.IPlugin))>
Public Class CustomCode
    Implements IPlugin

    Public Function Name() As String Implements IPlugin.Name
        Return "Two Observed Variables"
    End Function

    Public Function Description() As String Implements IPlugin.Description
        Return "Draws two observed variables called ThingOne and ThingTwo"
    End Function

    Public Function Mainsub() As Integer Implements IPlugin.MainSub
        pd.Observed("ThingOne")
        pd.Observed("ThingTwo")
    End Function
```

Amos calls the Name function to obtain the name of the plugin. Amos puts this name on the Amos Graphics Plugins menu. In the present example Two Observed Variables will be placed on the Plugins menu. It is OK for the Name function to return an empty string (""). In that case, the Plugins menu will display the name of the file in in which you save the plugin.

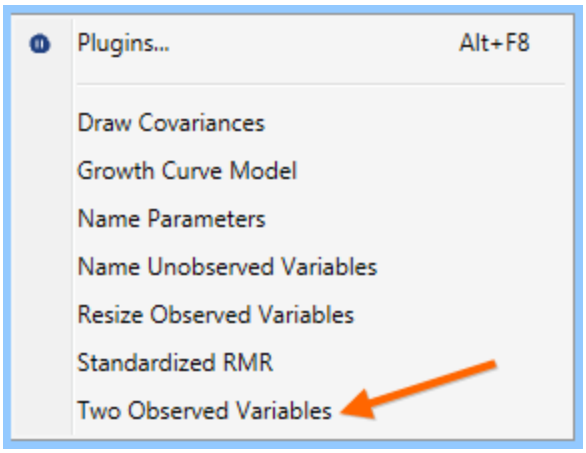
Amos calls the Description function to obtain a description of the plugin. A plugin's description is typically longer than its name. A plugin's description is displayed when the plugin is selected from the list in the Plugins dialog. It is OK if the Description function returns an empty string ("").

Amos will call the MainSub function (in other words, it will draw two variables called ThingOne and ThingTwo) when you select Two Observed Variables from the Plugins menu.

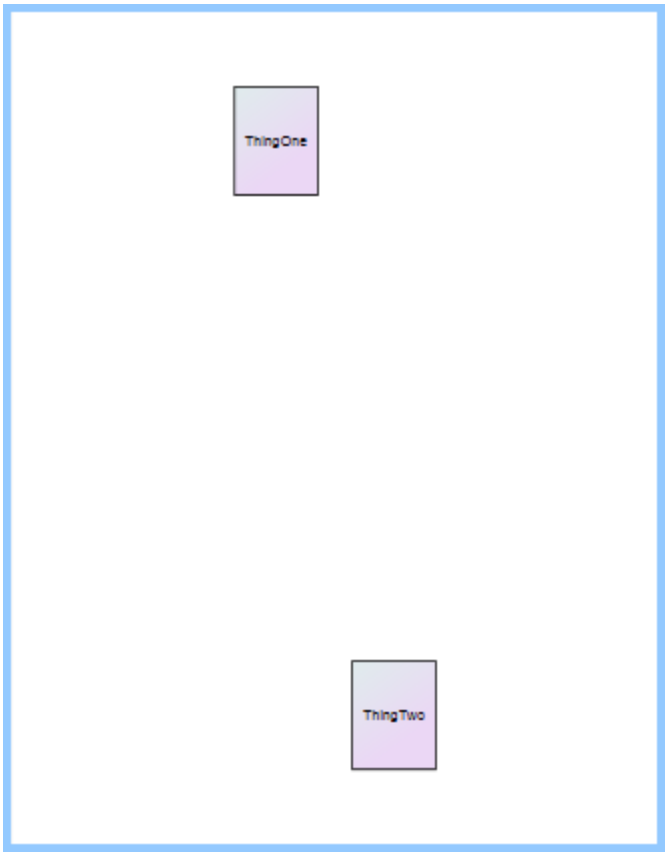
In this example, only the name of each new observed variable is specified. You can also specify the location and size of the rectangle that represents an observed variable, and you can place constraints on the variable's mean and variance. See the Observed Method 94 for more details.

8. Create the plugin by clicking Build > Rebuild Solution.

9. To test the plugin, click Plugins on the Amos Graphics main menu. The Plugins menu will contain the entry Draw Observed Variables.



10. Click Plugins > Two Observed Variables to draw the new variables



4.4.2 Calculating Custom Estimands

When doing Bayesian estimation, you can supply a class that calculates some function (that you specify) of the model parameters. Amos calls upon your class repeatedly to calculate the function, each time with a different set of parameter values. In the end, Amos uses the information gathered in this way to approximate the posterior distribution of your function. This function is referred to in the Amos documentation as a *custom estimand*.

Example 29 in the *User's Guide* shows how to specify a custom estimand in Visual Basic. Specifying a custom estimand in C# is similar.

4.4.2.1 Open

Open an existing file of Bayesian custom estimands.

4.4.2.2 Print preview

Display onscreen a preview of the file containing custom estimands, showing how the file will look when it is printed after you press the **Print** button.

4.4.2.3 Print

Print the file that contains custom estimands. Press the **Print preview** button to see what the file will look like when it is printed.

4.4.2.4 Copy

Copy selected text to the clipboard.

4.4.2.5 Run

Estimate the custom estimands. Calculation of the estimates is based on the MCMC samples that have already been collected. The calculation may take a while if the number of accumulated MCMC samples is large.

4.5 Class Reference

4.5.1 Amos Graphics Class Reference

Amos Graphics provides two classes:

1. The Pd class is used to draw path diagrams.
2. Objects of type **PDElement** make up the elements of a path diagram, such as rectangles, ellipses, arrows or figure captions.

If you are not using Amos's built-in program editor, you need to provide a reference to **Amos.dll** in order to use the **Amos Graphics** classes. In Visual Studio 2003:

- Click **Project -> Add Reference**.
- In the **Add Reference** dialog, click **Browse**.
- When the **Select Component** dialog opens, select Amos.dll from the Amos program directory and click **Open**.
- In the **Add Reference** dialog, click **OK**.

4.5.1.1 Pd Class Members

The **Pd** class is used to draw path diagrams.

4.5.1.1.1 Properties

This section documents the properties of the Pd class.

4.5.1.1.1.1 AmosDir Property

The Amos program directory.

Syntax

result = *Pd*.**AmosDir**

The **AmosDir** property syntax has the following parts:

Part	Description
result	The Amos program directory. *result* is a character string ending with a backslash character, for example, "C:\Program Files\IBM\SPSS\Amos\32\".

4.5.1.1.1.2 IsViewingPathDiagram Property

Gets or sets a value that is True if the **Path Diagram** view is selected and False otherwise. For example,

```
Pd.IsViewingPathDiagram=True
```

displays the **Path Diagram** view.

See IsViewingTables Property 45.

4.5.1.1.1.3 IsViewingTables Property

Gets or sets a value that is True if the **Tables** view is selected and False otherwise. For example,

```
Pd.IsViewingTables=True
```

displays the **Tables** view.

See IsViewingPathDiagram Property 45.

4.5.1.1.1.4 NGroups Property

Gets the number of groups.

Syntax

result = **Pd.NGroups**

The **NGroups** property syntax has the following parts:

Part	Description
result	The number of groups.

See example 46.

When you run this plugin, a message box displays the number of groups.

```
Imports Microsoft.VisualBasic
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      MsgBox("Number of groups = " & Pd.ngroups)
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.1.5 NotReady Property

The **NotReady** property always returns False. It is provided for compatibility with earlier versions of Amos.

Syntax

result = **Pd.NotReady**

The **NotReady** method syntax has the following parts:

Part	Description
result	True.

See DiagramDrawIndicatorVariable Method Example 58

4.5.1.1.1.6 PageHeight Property

Gets the page height in inches.

Syntax

result = **Pd.PageHeight**

The **PageHeight** method syntax has the following parts:

Part	Description
result	The height of the path diagram. If the path diagram's height is obtained from the Windows printer driver, *value* may be smaller than the height of a sheet of paper because the printer may be incapable of printing to the edges of the paper.

See example 47.

Running this plugin draws an ellipse in the center of the path diagram. The ellipse is one fourth as tall and one fourth as wide as the path diagram.

```
Imports Microsoft.VisualBasic
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim X As Single, Y As Single
      Dim Width As Single, Height As Single

      X = PageWidth / 2
      Y = PageHeight / 2
      Width = PageWidth / 4
      Height = PageHeight / 4

      DiagramDraw Unobserved(X, Y, Width, Height)
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.1.7 PageWidth Property

Gets the page width in inches.

Syntax

result = **Pd.PageWidth**

The **PageWidth** method syntax has the following parts:

Part	Description
result	The width of the path diagram. If the path diagram's width is obtained from the Windows printer driver, *value* may be smaller than the width of a sheet of paper because the printer may be incapable of printing to the edges of the paper.

See PageHeight Property Example 47

4.5.1.1.1.8 PDElements Property

Gets the collection of all rectangles, ellipses, arrows and figure captions in the path diagram for the current group.

Syntax

result = **Pd.PDElements**

The **PDElements** method syntax has the following parts:

Part	Description
result	The collection of all rectangles, ellipses, arrows and figure captions in the path diagram for the current group. The collection is 1-based. That is, the items in the collection are indexed starting with 1.

See IsCovariance Property Example 145

4.5.1.1.1.9 ProjectName Property

Gets the name of the file that contains the current path diagram, but without the ".amw" extension.

Syntax

result = **Pd.ProjectName**

The **ProjectName** method syntax has the following parts:

Part	Description
result	The file name, without the ".amw" extension.

4.5.1.1.2 Methods

This section documents the methods of the Pd class.

4.5.1.1.2.1 Amw FileName Method

Gets the name of the AMW file associated with the model.

Syntax

Pd.AmwFileName()

4.5.1.1.2.2 AnalyzeBayesianEstimation Method

Equivalent to the Amos Graphics menu item **Analyze** → **Bayesian Estimation**.

Syntax

```
Pd.AnalyzeBayesianEstimation ()
```

4.5.1.1.2.3 AnalyzeCalculateEstimates Method

Fits the specified models. This method is equivalent to the menu selection **Analyze** → **Calculate Estimates**.

Syntax

```
Pd.AnalyzeCalculateEstimates ()
```

4.5.1.1.2.4 AnalyzeDataImputation Method

Equivalent to the Amos Graphics menu item **Analyze** → **Data Imputation**.

Syntax

```
Pd.AnalyzeDataImputation ()
```

4.5.1.1.2.5 AnalyzeDegreesOfFreedom Method

Displays degrees of freedom. This method is equivalent to the menu selection **Analyze** → **Degrees Of Freedom**.

Syntax

```
Pd.AnalyzeDegreesOfFreedom ()
```

4.5.1.1.2.6 AnalyzeManageGroups Method

Opens a dialog for adding, deleting and renaming groups. This method is equivalent to the menu selection **Analyze** → **Manage Groups**.

Syntax

```
Pd.AnalyzeManageGroups ()
```

4.5.1.1.2.7 AnalyzeManageGroupsAdd Method

Adds a new group.

Syntax

```
Pd.AnalyzeManageGroupsAdd ()
```

4.5.1.1.2.8 AnalyzeManageGroupsDelete Method

Deletes the currently selected group. This method is equivalent to pressing the **Delete** button in the **Manage Groups** dialog.

Syntax

Pd.AnalyzeManageGroupsDelete ()

See GroupSelect Method Example 90

4.5.1.1.2.9 AnalyzeManageGroupsRename Method

Renames the currently selected group.

Syntax

Pd.AnalyzeManageGroupsRename (*newGroupName*)

The **AnalyzeManageGroupsRename** method syntax has the following parts:

Part	Description
newGroupName	New name for the currently selected group.

See GroupSelect Method Example 90

4.5.1.1.2.10 AnalyzeManageModels Method

Opens a dialog for creating, modifying and deleting models. This method is equivalent to the menu selection **Analyze** → **Manage Models**.

Syntax

Pd.AnalyzeManageModels ()

4.5.1.1.2.11 AnalyzeModelingLab Method

Runs the Modeling Lab. This method is equivalent to the menu selection **Analyze** → **Modeling Lab**.

Syntax

Pd.AnalyzeModelingLab ()

4.5.1.1.2.12 AnalyzeMultipleGroupAnalysis Method

Equivalent to the Amos Graphics menu item **Analyze** → **Multiple-Group Analysis**.

Syntax

Pd.AnalyzeMultipleGroupAnalysis ()

4.5.1.1.2.13 AnalyzeSpecificationSearch Method

Equivalent to the Amos Graphics menu item **Analyze** → **Specification Search**.

Syntax

Pd.AnalyzeSpecificationSearch ()

4.5.1.1.2.14 AnalyzeToggleObservedUnobserved Method

Changes rectangles to ellipses, and ellipses to rectangles.

Syntax

Pd.AnalyzeToggleObservedUnobserved ()
Pd.AnalyzeToggleObservedUnobserved (*theElement*)
Pd.AnalyzeToggleObservedUnobserved (*variableName*)
Pd.AnalyzeToggleObservedUnobserved (*elementNumber*)

The **AnalyzeToggleObservedUnobserved** method syntax has the following parts:

Part	Description
theElement	A rectangle or ellipse (of type **PDElement**).
variableName	(String) The name of a variable.
elementNumber	An integer that specifies a rectangle or ellipse in the path diagram. The objects in a path diagram are numbered starting with 1.

Calling **AnalyzeToggleObservedUnobserved** with no arguments is equivalent to the menu selection **Analyze** → **Toggle Observed/Unobserved**.

4.5.1.1.2.15 BuildNumber Method

Gets the build number that is displayed in the **About** box.

Syntax

result = **Pd.BuildNumber** ()

The **BuildNumber** method syntax has the following parts:

Part	Description
result	(Integer) The build number.

4.5.1.1.2.16 CanRespond Method

If Amos Graphics has an open modal dialog that may prevent it from responding correctly to calls to its methods and properties, the **CanRespond** method returns an explanatory string. Otherwise, it returns an empty string.

Syntax

result=**Pd.CanRespond** (*callerDisplaysErrorMessage*)

The **CanRespond** method syntax has the following parts:

Part	Description
result	(String) The empty string ("") if Amos Graphics can respond. Otherwise, a description of the condition that prevents Amos Graphics from responding.
callerDisplaysErrorMessage	(boolean) False if you want Amos Graphics to display a message box with a description of any condition that prevents it from responding. True if you want the calling program to assume responsibility for displaying such messages.

See example 52.

The following plugin checks to make sure that there are no open modal dialogs that may prevent Amos Graphics from responding correctly. Then the plugin specifies the data file for group number 1.

```
Imports System
Imports Microsoft.VisualBasic
Imports Amos
Imports PXMLPersist.CDataTable
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim S As String
      'Wait until Amos Graphics can respond
      Do
         S = Pd.CanRespond(True)
         If S = "" Then
            Exit Do
         End If
         If MsgBox(S, vbOKCancel) = vbCancel Then
            Return 0
         End If
      Loop

      ' Change the data file for group number 1
      If Pd.SetDataFile(1, cDatabaseFormat.mmEXCEL97, _
         Pd.AmosDir & "Examples\English\userguide.xls", _
         "grant", "", 0) = 0 Then
         MsgBox("Successful",, "CanRespond Example")
      Else
         MsgBox("Unsuccessful",, "CanRespond Example")
      End If
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.2.17 Caption Method

Creates a figure caption.

Syntax

result = **Pd.Caption** (*theCaption*)
result = **Pd.Caption** (*theCaption, x, y, position*)

The **Caption** method syntax has the following parts:

Part	Description
result	(*Object of type* **PDElement**) The newly created figure caption.
theCaption	(*String*) The figure caption text.

x	(*Double*) The horizontal position of the caption measured in units of 1/96 inch. (The horizontal position is also affected by the *position* argument.)
y	(*Double*) The vertical distance from the top edge of the path diagram to the center of the figure caption, measured in units of 1/96 inch.
position	(*Of type* **PDElement.TitlePositionEnum**) An enumeration that specifies one of the following ways of formatting the new figure caption: Centered at the caption's x coordinate Left Justified at the caption's x coordinate Right justified at the caption's x coordinate Centered horizontally on the path diagram

4.5.1.1.2.18 ClickMouse Method

Clicks the mouse on a rectangle, ellipse, arrow or figure caption.

Syntax

Pd.ClickMouse (*theElement*)
Pd.ClickMouse (*variableName*)
Pd.ClickMouse (*elementNumber*)

The **ClickMouse** method syntax has the following parts:

Part	Description
theElement	A path diagram object (of type **PDElement**).
variableName	(String) The name of a variable.
elementNumber	An integer that specifies an object in the path diagram. Objects are numbered starting with 1.

4.5.1.1.2.19 CopyAnalysisPropertiesTo Method

Copies the properties in the **Analysis Properties** window to an **AmosEngine** instance.

Syntax

result = **Pd.CopyAnalysisPropertiesTo** (*sem*)

The **CopyAnalysisPropertiesTo** method syntax has the following parts:

Part	Description
result	(Integer) 0 = no error.
sem	An object of type **AmosEngine**.

4.5.1.1.2.20 Cov Method

Draws a double-headed arrow.

Syntax

result = **Pd.Cov** (*variable1, variable2*)
result = **Pd.Cov** (*variable1, variable2, covarianceValue*)
result = **Pd.Cov** (*variable1, variable2, covarianceName*)
result = **Pd.Cov** (*variableName1, variableName2*)
result = **Pd.Cov** (*variableName1, variableName2, covarianceValue*)
result = **Pd.Cov** (*variableName1, variableName2, covarianceName*)

The **Cov** method syntax has the following parts:

Part	Description
result	(*Object of type* **PDElement**) The new double-headed arrow.
variable1, variable2	(*Objects of type* **PDElement**) The two variables to be connected by the new double-headed arrow.
variableName1, variableName2	*(Strings)* Names of the two variables to be connected by the new double-headed arrow.
covarianceValue	(*Double*) The value of the covariance represented by the new double-headed arrow.
covarianceName	(*String*) A name for the covariance represented by the new double-headed arrow.

Remarks

If you do not specify a name or value for the covariance represented by the double-headed arrow, the covariance will be unconstrained.

4.5.1.1.2.21 DiagramDraw Covariance Method

Draws a double-headed arrow.

Syntax

covariance = **Pd.DiagramDrawCovariance**()
covariance = **Pd.DiagramDrawCovariance**(*theElement1*, *theElement2*)
covariance = **Pd.DiagramDrawCovariance**(*variableName1*, *variableName2*)
covariance = **Pd.DiagramDrawCovariance**(*elementNumber1*, *elementNumber2*)
covariance = **Pd.DiagramDrawCovariance**(*x1, y1, x2, y2*)

The **DiagramDrawCovariance** method syntax has the following parts:

Part	Description
covariance	The newly drawn covariance. (An object of type **PDElement**.)
theElement1, theElement2	(Of type **PDElement**) The two variables to be connected by a double-headed arrow.
variableName1, variableName2	(String) The names of the two variables to be connected by a double-headed arrow.
elementNumber1, elementNumber2	*(Integer)* Numbers that identify the two variables to be connected by a double-headed arrow. Objects in a path diagram are arbitrarily numbered beginning with 1.
x1, y1	(*Single*) Coordinates of one of the variables to be connected by a double-headed arrow. *x1* is its distance in inches from the left edge of the path diagram. *y1* is its distance in inches from the top edge.
x2, y2	(Single) Coordinates of one of the variables to be connected by a double-headed arrow. *x2* is its distance in inches from the left edge of the path diagram. *y2* is its distance in inches from the top edge.

Calling **DiagramDrawCovariance** with no arguments is equivalent to the menu selection **Diagram** → **Draw Covariance**.

See Use the Amos Graphics classes to draw a path diagram 494

4.5.1.1.2.22 DiagramDraw IndicatorVariable Method

Draws an indicator variable and an associated residual variable for an existing unobserved variable.

Syntax

Pd.DiagramDrawIndicatorVariable ()
Pd.DiagramDrawIndicatorVariable (*theElement*)
Pd.DiagramDrawIndicatorVariable (*variableName*)
Pd.DiagramDrawIndicatorVariable (*elementNumber*)

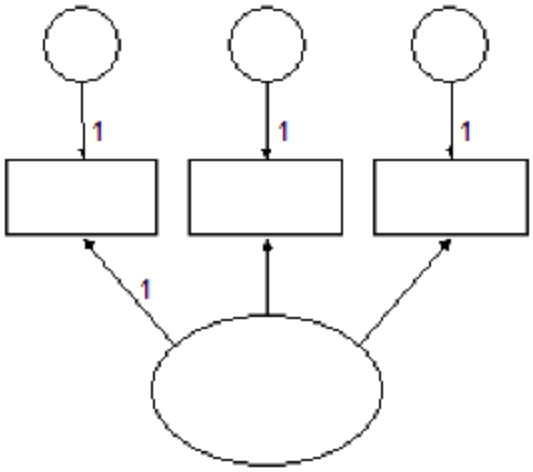
The **DiagramDrawIndicatorVariable** method syntax has the following parts:

Part	Description
theElement	(Of type **PDElement**) An unobserved variable (ellipse).
variableName	(String) The name of an unobserved variable (ellipse).
elementNumber	(Integer) A number that identifies an unobserved variable (ellipse). Objects in a path diagram are arbitrarily numbered beginning with 1.

Calling **DiagramDrawIndicatorVariable** with no arguments is equivalent to the menu selection **Diagram** → **Draw Indicator Variable**.

See example 58.

The plugin below draws the following path diagram.




```
Imports Microsoft.VisualBasic
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    Dim E As PDElement
    Pd.FileNew (Pd.PDSaveChangesEnum.pdPromptToSaveChanges)
    E = Pd.DiagramDraw Unobserved(3, 3, 2, 1)
    Pd.DiagramDrawIndicatorVariable(E)
    Pd.DiagramDrawIndicatorVariable(E)
    Pd.DiagramDrawIndicatorVariable(E)
    Pd.Refresh()
    Pd.EditFitToPage()
  End Function

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.5.1.1.2.23 DiagramDraw Observed Method

Draws observed variables (rectangles).

Syntax

variable = **Pd.DiagramDrawObserved** ()

variable = **Pd.DiagramDrawObserved** (*x, y, width, height*)

The **DiagramDrawObserved** method syntax has the following parts:

Part	Description
variable	The newly drawn observed variable. (An object of type **PDElement**.)
x	Horizontal coordinate of the center of the rectangle, expressed in inches from the left margin.
y	Vertical coordinate of the center of the rectangle, expressed in inches from the top margin.
width	Width of the rectangle, in inches.
height	Height of the rectangle, in inches

Calling **DiagramDrawObserved** with no arguments is equivalent to the menu selection **Diagram → Draw Observed**.

See Use the Amos Graphics classes to draw a path diagram 494

4.5.1.1.2.24 DiagramDraw Path Method

Draws regression weights (single-headed arrows).

Syntax

weight = **Pd.DiagramDrawPath**()
weight = **Pd.DiagramDrawPath**(*theElement1, theElement2*)
weight = **Pd.DiagramDrawPath**(*variableName1, variableName2*)
weight = **Pd.DiagramDrawPath**(*elementNumber1, elementNumber2*)
weight = **Pd.DiagramDrawPath**(*x1, y1, x2, y2*)

The **DiagramDrawPath** method syntax has the following parts:

Part	Description
weight	The newly drawn arrow (regression weight). (An object of type **PDElement**.)
theElement1, theElement2	(Of type **PDElement**) The new arrow points from *theElement1* to *theElement2*.

variableName1, variableName2	(String) The names of two variables. The new arrow points from *variableName1* to *variableName2*.
elementNumber1, elementNumber2	(Integer) Numbers that identify two variables. Objects in a path diagram are arbitrarily numbered beginning with 1. The new arrow points from the first variable to the second.
x1, y1	(Single) Coordinates of the variable that the new arrow points from. *x1* is its distance in inches from the left edge of the path diagram. *y1* is its distance in inches from the top edge.
x2, y2	(Single) Coordinates of the variable that the new arrow points to. *x2* is its distance in inches from the left edge of the path diagram. *y2* is its distance in inches from the top edge.

Calling **DiagramDrawPath** with no arguments is equivalent to the menu selection **Diagram** → **Draw Path**.

See Use the Amos Graphics classes to draw a path diagram 494

4.5.1.1.2.25 DiagramDraw UniqueVariable Method

Draws a unique (residual) variable for an existing variable.

Syntax

Pd.DiagramDrawUniqueVariable ()
Pd.DiagramDrawUniqueVariable (*theElement*)
Pd.DiagramDrawUniqueVariable (*variableName*)
Pd.DiagramDrawUniqueVariable (*elementNumber*)

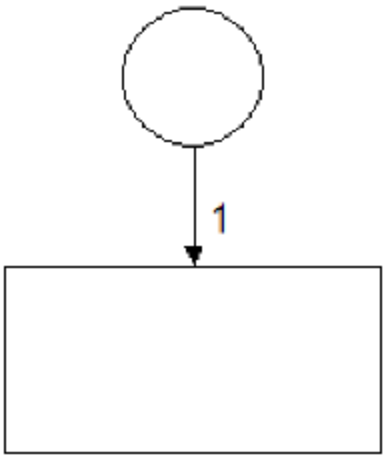
The **DiagramDrawIndicatorVariable** method syntax has the following parts:

Part	Description
theElement	(Of type **PDElement**) A variable (ellipse or rectangle).
variableName	(String) The name of a variable.
elementNumber	(Integer) A number that identifies a variable (ellipse or rectangle). Objects in a path diagram are arbitrarily numbered beginning with 1.

Calling **DiagramDrawUniqueVariable** with no arguments is equivalent to the menu selection **Diagram** → **Draw Unique Variable**.

See example [61].

The plugin below draws the following path diagram.




```
Imports Microsoft.VisualBasic
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim E As PDElement
      Pd.FileNew (Pd.PDSaveChangesEnum.pdPromptToSaveChanges)
      E = Pd.DiagramDraw Observed(3, 3, 2, 1)

      Pd.DiagramDrawUniqueVariable(E)
      Pd.Refresh()
      Pd.EditFitToPage()
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.2.26 DiagramDraw Unobserved Method

Draws unobserved variables (ellipses).

Syntax

variable = **Pd.DiagramDrawUnobserved** (*x*, *y*, *width*, *height*)

The **DiagramDrawUnobserved** method syntax has the following parts:

Part	Description
variable	The newly drawn unobserved variable (ellipse). (An object of type **PDElement**.)
x	Horizontal coordinate of the center of the ellipse, expressed in inches from the left margin.
y	Vertical coordinate of the center of the ellipse, expressed in inches from the top margin.
width	Width of the ellipse, in inches.
height	Height of the ellipse, in inches

Calling **DiagramDrawUnobserved** with no arguments is equivalent to the menu selection **Diagram** → **Draw Unobserved**.

See Use the Amos Graphics classes to draw a path diagram 494

4.5.1.1.2.27 DiagramFigureCaption Method

Draws new figure captions, and edits existing figure captions.

Syntax

Pd.DiagramFigureCaption ()
Pd.DiagramFigureCaption (*theElement*)
Pd.DiagramFigureCaption (*elementNumber*)

The **DiagramFigureCaption** method syntax has the following parts:

Part	Description
theElement	(Of type **PDElement**) An existing figure caption.
elementNumber	(Integer) A number that identifies a figure caption. Objects in a path diagram are arbitrarily numbered beginning with 1.

Calling **DiagramFigureCaption** with no arguments is equivalent to the menu selection **Diagram** → **Figure Caption**.

4.5.1.1.2.28 DiagramLoupe Method

Magnifies the region under the mouse pointer. This method is equivalent to the menu selection **Diagram** → **Loupe**.

Syntax

Pd.DiagramLoupe()

4.5.1.1.2.29 DiagramRedraw Diagram Method

Redraws the path diagram. This method is equivalent to the menu selection **Diagram** → **Redraw Diagram**.

Syntax

Pd.DiagramRedrawDiagram()

See IsObservedVariable Property Example 149

4.5.1.1.2.30 DiagramScroll Method

Scrolls the path diagram.

Syntax

Pd.DiagramScroll ()
Pd.DiagramScroll (*x*, *y*)

The **DiagramScroll** method syntax has the following parts:

Part	Description
x	The path diagram is scrolled x inches to the right.
y	The path diagram is scrolled y inches down.

Calling **DiagramScroll** with no arguments is equivalent to the menu selection **Diagram** → **Scroll**.

See example 64.

The following plugin scrolls the path diagram one inch to the right.

```
Imports System
Imports Amos
Imports Microsoft.VisualBasic
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Name() As String Implements IPlugin.Name
      Return "DiagramScroll Method Example"
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Amos.Pd.diagramscroll (1,0)
   End Function
End Class
```

4.5.1.1.2.31 DiagramZoom Method

Fills the Amos window with a selected portion of the path diagram.

Syntax

Pd.DiagramZoom ()
Pd.DiagramZoom (*x1*, *y1*, *x2*, *y2*)

The **DiagramZoom** method syntax has the following parts:

Part	Description
x1, *y1*, *x2*, *y2*	(x1, y1) and (x2, y2) specify two opposite corners of a rectangle. x1 and x2 are expressed in inches from the left margin. y1 and y2 are expressed in inches from the top margin. The **DiagramZoom** method resizes the path diagram so that the specified rectangle fills the Amos window.

Calling **DiagramZoom** with no arguments is equivalent to the menu selection **Diagram** → **Zoom**.

See example 64.

The following plugin zooms in or out so that the latent variables in the path diagram just fill the Amos Graphics window. If the path diagram contains no latent variables, no zoom is performed.

```
Imports Microsoft.VisualBasic
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
  Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    Dim ZTop As Object, ZBottom As Object
    Dim ZLeft As Object, ZRight As Object
    Dim E As PDElement
    Dim Temp As Single
    ' 999999 inches is an impossible value
    Const HugeNumber = 999999
    ZLeft = HugeNumber
    ZTop = HugeNumber
    ZRight = -HugeNumber
    ZBottom = -HugeNumber
    For Each E In Pd.PDElements
      If E.IsLatentVariable Then
        Temp = E.OriginX - E.Width / 2
        If Temp < ZLeft Then ZLeft = Temp
        Temp = E.OriginY - E.Height / 2
        If Temp < ZTop Then ZTop = Temp
        Temp = E.OriginX + E.Width / 2
        If Temp > ZRight Then ZRight = Temp
        Temp = E.OriginY + E.Height / 2
        If Temp > ZBottom Then ZBottom = Temp
      End If
    Next
    'If there are no latent variables, return without zooming
    If ZLeft = HugeNumber Then Return 0
    If ZTop = HugeNumber Then Return 0
    If ZRight = -HugeNumber Then Return 0
    If ZBottom = -HugeNumber Then Return 0
    Pd.DiagramZoom(ZLeft, ZTop, ZRight, ZBottom)
  End Function

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.5.1.1.2.32 DiagramZoomIn Method

Magnifies the screen image of the path diagram. This method is equivalent to the menu selection **Diagram → Zoom In**.

Syntax

Pd.DiagramZoomIn ()

See example 65.

The following plugin resizes the screen image of the path diagram so that one printed page just fits in the Amos Graphics window, and then enlarges the screen image of the path diagram.

```
Imports Microsoft.VisualBasic
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
  Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    DiagramZoomPage()
    DiagramZoomIn()
  End Function

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.5.1.1.2.33 DiagramZoomOut Method

Reduces the size of the screen image of the path diagram. This method is equivalent to the menu selection **Diagram** → **Zoom Out**.

Syntax

Pd.DiagramZoomOut ()

See example 66.

The following plugin resizes the screen image of the path diagram so that one printed page just fits in the Amos Graphics window, and then reduces the size of the screen image of the path diagram.

```
Imports Microsoft.VisualBasic
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
  Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    DiagramZoomPage()
    DiagramZoomOut()
  End Function

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.5.1.1.2.34 DiagramZoomPage Method

Resizes the screen image of the path diagram so that one printed page just fits the Amos window. This method is equivalent to the menu selection **Diagram** → **Zoom Page**.

Syntax

Pd.DiagramZoomPage ()

See DiagramZoomIn Method Example 65

4.5.1.1.2.35 DisplayInputPD Method

Displays the input path diagram. This method is equivalent to pressing  .

Syntax

Pd.DisplayInputPD ()

See DisplayOutputPD Method 67

4.5.1.1.2.36 DisplayOutputPD Method

Displays the input path diagram. This method is equivalent to pressing  .

Syntax

Pd.DisplayOutputPD ()

See DisplayInputPD Method 67

4.5.1.1.2.37 DoubleClickMouse Method

The **DoubleClickMouse** method is no longer supported.

4.5.1.1.2.38 DragMouse Method

Drags the mouse from one point to another.

Syntax

Pd.DragMouse (*theElement, x2, y2*)

Pd.DragMouse (*x1, y1, x2, y2*)

The **DragMouse** method syntax has the following parts:

Part	Description
theElement	An object of type **PDElement**. The drag operation begins at the point with coordinates (*theElement*.originX 135, *theElement*.originY 135).

x1, y1	Coordinates of the location where the drag operation begins (the left mouse button is pressed). x1 is its distance, in inches, from the left margin. y1 is its distance, in inches, from the top margin.
x2, y2	Coordinates of the location where the drag operation ends (the left mouse button is released). x2 is its distance, in inches, from the left margin. y2 is its distance, in inches, from the top margin.

4.5.1.1.2.39 EditCopy Method

Copies the path diagram to the Windows clipboard. This method is equivalent to the menu selection **Edit** → **Copy**.

Syntax

`Pd.EditCopy` ()

See example 68.

Running the following plugin has the same effect as selecting **Edit** → **Copy** on the Amos Graphics menu.

```
Imports System
Imports Microsoft.VisualBasic
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Pd.EditCopy
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.2.40 EditDeselectAll Method

Deselects all objects. This method is equivalent to the menu selection **Edit** → **Deselect All**.

Syntax

`Pd.EditDeselectAll` ()

See Use the Amos Graphics classes to draw a path diagram 494

4.5.1.1.2.41 EditDragProperties Method

Copies the properties of one object (the source object) to another object (the target object).

Syntax

Pd.EditDragProperties ()

Pd.EditDragProperties (*theElement1, theElement2, height, width, x, y, nameOrCaption, parameterConstraints, parameterPosition, font, parameterFont, penWidth, curvature, colors, visibility*)

Pd.EditDragProperties *(elementNumber1, elementNumber2, height, width, x, y, nameOrCaption, parameterConstraints, parameterPosition, font, parameterFont, penWidth, curvature, colors, visibility*)

Pd.EditDragProperties *(variableName1, variableName2, height, width, x, y, nameOrCaption, parameterConstraints, parameterPosition, font, parameterFont, penWidth, curvature, colors, visibility*)

The **EditDragProperties** method syntax has the following parts:

Part	Description
theElement1 *theElement2*	(Objects of type **PDElement**.) Properties are copied from *theElement1 to theElement2*.
elementNumber1 *elementNumber2*	(Integers) Properties are copied from object number *elementNumber1* to object number *elementNumber2*. (Objects in the path diagram are numbered starting with 1.)
variableName1 *variableName2*	(Strings) Properties are copied from the variable named *variableName1* to the variable named *variableName1*.
height	Optional. True if the source object's height is to be copied.
width	Optional. True if the source object's width is to be copied.
x	Optional. True if the source object's x (horizontal) coordinate is to be copied.
y	Optional. True if the source object's y (vertical) coordinate is to be copied.

nameOrCaption	Optional. True if the source object's name (in the case of a rectangle or ellipse) or caption (in the case of a figure caption) is to be copied.
parameterConstraints	Optional. True if the source object's parameter constraints are to be copied.
parameterPosition	Optional. True if the source object's parameter position is to be copied.
font	Optional. True if the source object's name or caption font is to be copied.
parameterFont	Optional. True if the source object's parameter font is to be copied.
penWidth	Optional. True if the source object's pen width is to be copied.
curvature	Optional. True if the source object's curvature is to be copied.
colors	Optional. True if the source object's colors are to be copied.
visibility	Optional. True if the source object's visibility properties are to be copied.

Calling **`EditDragProperties`** with no arguments is equivalent to the menu selection **Edit** → **Drag Properties**.

4.5.1.1.2.42 EditDuplicate Method

Makes additional copies of rectangles, ellipses and captions.

Syntax

`Pd.EditDuplicate` ()
`Pd.EditDuplicate` (*theElement, x2, y2*)
`Pd.EditDuplicate` (*variableName, x2, y2*)
`Pd.EditDuplicate` (*elementNumber, x2, y2*)

The **`EditDuplicate`** method syntax has the following parts:

Part	Description
theElement	A path diagram object (of type **PDElement**) to be copied.
variableName	(String) The name of a variable to be copied.
elementNumber	An integer that specifies an object to be copied. Objects are numbered starting with 1.
x2, *y2*	(Single) The coordinates of the newly created object. x2 is its distance, in inches, from the left margin. y2 is its distance, in inches, from the top margin.

Calling **EditDuplicate** with no arguments is equivalent to the menu selection **Edit** → **Duplicate**.

4.5.1.1.2.43 EditErase Method

Erases objects.

Syntax

Pd.EditErase ()
Pd.EditErase (*theElement*)
Pd.EditErase (*variableName*)
Pd.EditErase (*elementNumber*)

The **EditErase** method syntax has the following parts:

Part	Description
theElement	A path diagram object (of type **PDElement**) to be erased.
variableName	(String) The name of a variable to be erased.
elementNumber	An integer that specifies an object to be erased. Objects are numbered starting with 1.

Calling **EditErase** with no arguments is equivalent to the menu selection **Edit** → **Erase**.

4.5.1.1.2.44 EditFitToPage Method

Resizes the path diagram (not just its screen image) so that it fits on a page. This method is equivalent to the menu selection **Edit** → **Fit to Page**.

Syntax

Pd.EditFitToPage ()

See Use the Amos Graphics classes to draw a path diagram 494

4.5.1.1.2.45 EditLink Method

Forms a group of objects that will be treated as a unit in future operations. This method is equivalent to the menu selection **Edit** → **Link**.

Syntax

Pd.EditLink ()

4.5.1.1.2.46 EditMove Method

Moves rectangles, ellipses and captions.

Syntax

Pd.EditMove ()
Pd.EditMove (*theElement, x2, y2*)
Pd.EditMove (*variableName, x2, y2*)
Pd.EditMove (*elementNumber, x2, y2*)

The **EditMove** method syntax has the following parts:

Part	Description
theElement	A path diagram object (of type **PDElement**) to be moved.
variableName	(String) The name of a variable to be moved.
elementNumber	An integer that specifies an object to be moved. Objects are numbered starting with 1.
x2, y2	(Single) The new coordinates of the object. x2 is its distance, in inches, from the left margin. y2 is its distance, in inches, from the top margin.

Calling **EditMove** with no arguments is equivalent to the menu selection **Edit** → **Move**.

4.5.1.1.2.47 EditMoveParameter Method

Allows parameter constraints and estimates to be moved. This method is equivalent to the menu selection **Edit** → **Move Parameter**.

Syntax

Pd.EditMoveParameter ()

4.5.1.1.2.48 EditPaste Method

Pastes a path diagram, or a part of a path diagram, from the Windows clipboard into the Amos Graphics window. This method is equivalent to the menu selection **Edit** → **Paste**.

Syntax

Pd.EditPaste ()

4.5.1.1.2.49 EditRedo Method

Undoes the effect of the most recent use of the EditUndo 77 method. This method is equivalent to the menu selection **Edit** → **Redo**.

Syntax

Pd.EditRedo()

4.5.1.1.2.50 EditReflect Method

Reflects the indicators of a latent variable.

Syntax

Pd.EditReflect ()
Pd.EditReflect (*theElement*)
Pd.EditReflect (*variableName*)
Pd.EditReflect (*elementNumber*)

The **EditReflect** method syntax has the following parts:

Part	Description
theElement	A latent variable (of type **PDElement**).
variableName	(String) The name of a latent variable.
elementNumber	An integer that specifies an object that is a latent variable. Objects are numbered starting with 1.

Calling **EditReflect** with no arguments is equivalent to the menu selection **Edit** → **Reflect**.

4.5.1.1.2.51 EditRotate Method

Rotates the indicators of a latent variable.

Syntax

Pd.EditRotate ()
Pd.EditRotate (*theElement*)
Pd.EditRotate (*variableName*)
Pd.EditRotate (*elementNumber*)

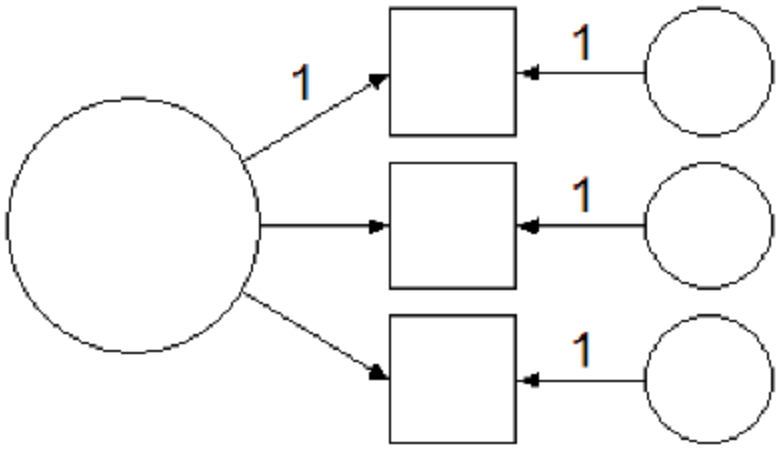
The **EditRotate** method syntax has the following parts:

Part	Description
theElement	A latent variable (of type **PDElement**).
variableName	(String) The name of a latent variable.
elementNumber	An integer that specifies an object that is a latent variable. Objects are numbered starting with 1.

Calling **EditRotate** with no arguments is equivalent to the menu selection **Edit** → **Rotate**.

See example 74.

The plugin below draws the following path diagram.



```
Imports Microsoft.VisualBasic
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim E As PDElement
      If FileNew(PDSaveChangesEnum.pdPromptToSaveChanges) Then
         Return 0
      End If
      E = DiagramDraw.Unobserved(3, 3, 2, 2)
      DiagramDraw.IndicatorVariable(E)
      DiagramDraw.IndicatorVariable(E)
      DiagramDraw.IndicatorVariable(E)

      EditRotate(E)
      Pd.Refresh()
      EditFitToPage()
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.2.52 EditSelect Method

Selects one object at a time.

Syntax

Pd.EditSelect ()
Pd.EditSelect (*theElement*)
Pd.EditSelect (*variableName*)
Pd.EditSelect (*elementNumber*)

The **EditSelect** method syntax has the following parts:

Part	Description
theElement	An object (of type **PDElement**) to be selected.
variableName	(String) The name of a variable to be selected.
elementNumber	An integer that specifies an object to be selected. Objects are numbered starting with 1.

Calling **EditSelect** with no arguments is equivalent to the menu selection **Edit** → **Select**.

4.5.1.1.2.53 EditSelectAll Method

Selects all objects in the path diagram. This method is equivalent to the menu selection **Edit** → **Select All**.

Syntax

Pd.EditSelectAll ()

See Use the Amos Graphics classes to draw a path diagram 494

4.5.1.1.2.54 EditShapeOfObject Method

Allows the user to change the size and shape of variables (rectangles and ellipses). This method is equivalent to the menu selection **Edit** → **Shape Of Object**.

Syntax

Pd.EditShapeOfObject ()

See Use the Amos Graphics classes to draw double-headed arrows 496

4.5.1.1.2.55 EditSpaceHorizontally Method

Arranges selected objects so that they are equally spaced horizontally. This method is equivalent to the menu selection **Edit** → **Space Horizontally**.

Syntax

Pd.EditSpaceHorizontally ()

4.5.1.1.2.56 EditSpaceVertically Method

Arranges selected objects so that they are equally spaced vertically. This method is equivalent to the menu selection **Edit** → **Space Vertically**.

Syntax

Pd.EditSpaceVertically ()

4.5.1.1.2.57 EditTouchUp Method

Rearranges the arrows in a path diagram in a way intended to be aesthetically pleasing.

Syntax

Pd.EditTouchUp ()
Pd.EditTouchUp (*theElement*)
Pd.EditTouchUp (*variableName*)
Pd.EditTouchUp (*elementNumber*)

The **EditTouchUp** method syntax has the following parts:

Part	Description
theElement	A rectangle or ellipse (of type **PDElement**). Arrows that touch the rectangle or ellipse will be repositioned.
variableName	(String) The name of a variable. Arrows that touch the variable's rectangle or ellipse will be repositioned.
elementNumber	An integer that specifies a rectangle or ellipse. Objects in a path diagram are numbered starting with 1. Arrows that touch the rectangle or ellipse will be repositioned.

Calling **EditTouchUp** with no arguments is equivalent to the menu selection **Edit** → **Touch Up**.

See Use the Amos Graphics classes to draw a path diagram 494

4.5.1.1.2.58 EditUndo Method

Undoes the most recent change to the path diagram. This method is equivalent to the menu selection **Edit** → **Undo**.

Syntax

Pd.EditUndo ()

4.5.1.1.2.59 EnableUserInteraction Method

Enable or disable toolbox buttons and menus.

Syntax

Pd.EnableUserInteraction (*tf*)

The **EnableUserInteraction** method syntax has the following parts:

Part	Description
tf	False to disable toolbox buttons and menus. True to enable them.

4.5.1.1.2.60 EnableUserInteraction2 Method

Like EnableUserInteraction, EnableUserInteraction2 enables or disables toolbox buttons and menus. However EnableUserInteraction2 always leaves the following toolbar buttons and menu items enabled.

-  Print a path diagram

-  Duplicate objects
-  Save a path diagram
-  Save a path diagram with a new name

Syntax

Pd.EnableUserInteraction2 (*tf. showHandCursorOnCanvas*)

The **EnableUserInteraction2** method syntax has the following parts:

Part	Description
tf	True to enable toolbox buttons and menus. False to disable them.
showHandCursorOnCanvas	True to display the 'hand' cursor on the path diagram area of the Amos Graphics window and display the 'wait' cursor on the remainder of the Amos Graphics window. False to display the 'wait' cursor on the entire Amos Graphics window.

See example 78.

The following plugin asks the user if it is ok to change the border colors of objects in the path diagram. If the user responds "yes", the program sets the border colors to magenta. The **EnableUserInteraction2** method is used to prevent the user from interacting with the Amos Graphics window while the program is waiting for a user response.

```
Imports Microsoft.VisualBasic
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
  Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    Dim x As PDElement
    Dim Result As MsgBoxResult

    EnableUserInteraction2(False, False)
    Result = MsgBox("Do you want to change the border colors?", vbYesNo)
    EnableUserInteraction2(True, False)

    If Result = vbYes Then
      For Each x In Pd.PDElements
        x.BorderColor = System.Drawing.Color.Magenta.ToArgb
        Pd.refresh()
      Next
    End If
  End Function

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.5.1.1.2.61 FileDataFiles Method

Opens a dialog to allow the user to specify the database file (or files) to be analyzed. This method is equivalent to the menu selection **File** → **Data Files**.

Syntax

```
Pd.FileDataFiles ()
```

4.5.1.1.2.62 FileExit Method

Closes the Amos Graphics window. This method is equivalent to the menu selection **File** → **Exit**.

Syntax

```
Pd.FileExit ()
```

See MouseDown and MouseUp Events Example 121.

4.5.1.1.2.63 FileNew Method

Starts a new path diagram.

Syntax

```
result = Pd.FileNew ()
```

result = **Pd.FileNew** (*saveOptions*)

The **FileNew** method syntax has the following parts:

Part	Description
result	True if an error occurs.
saveOptions	Optional. A constant, as specified in Settings, that tells Amos what to do if there is currently an unsaved path diagram in the Amos window.

If *saveOptions* is omitted, the **FileNew** method is equivalent to the menu selection **File** → **New**.

Settings

The settings for *saveOptions* are:

Constant	Value	Description
pdPromptToSaveChanges (default)	0	Prompt the user to save or discard the current path diagram
pdDoNotSaveChanges	1	Discard the current path diagram.
pdSaveChanges	2	Save the current path diagram.

See Use the Amos Graphics classes to draw a path diagram 494

4.5.1.1.2.64 FileNew WithTemplate Method

Starts a new path diagram using a template.

Syntax

result = **Pd.FileNewWithTemplate** ()
result = **Pd.FileNewWithTemplate** (*templateFileName*)
result = **Pd.FileNewWithTemplate** (*templateFileName*,*saveOptions*)

The **FileNewWithTemplate** method syntax has the following parts:

Part	Description
result	True if an error occurs or if the user cancels the operation.
templateFileName	The name of a template file.

saveOptions	Optional. A constant, as specified in Settings, that tells Amos what to do if there is currently an unsaved path diagram in the Amos window.

Calling **FileNewWithTemplate** with no arguments is equivalent to the menu selection **File** → **New With Template**.

Settings

The settings for *saveOptions* are:

Constant	Value	Description
pdPromptToSaveChanges (default)	0	Prompt the user to save or discard the current path diagram
pdDoNotSaveChanges	1	Discard the current path diagram.
pdSaveChanges	2	Save the current path diagram.

4.5.1.1.2.65 FileOpen Method

Retrieves a path diagram that was saved previously.

Syntax

result = **Pd.FileOpen** ()
result = **Pd.FileOpen** (*fileName*)
result = **Pd.FileOpen** (*fileName*, *saveOptions*)

The **FileOpen** method syntax has the following parts:

Part	Description
result	True if an error occurs or if the user cancels the operation.
fileName	The name of a path diagram file (a ***.amw** file).
saveOptions	Optional. A constant, as specified in Settings, that tells Amos what to do if there is currently an unsaved path diagram in the Amos window.

Calling **FileOpen** with no arguments is equivalent to the menu selection **File** → **Open**.

Settings

The settings for *saveOptions* are:

Constant	Value	Description
PdPromptToSaveChanges (default)	0	Prompt the user to save or discard the current path diagram
PdDoNotSaveChanges	1	Discard the current path diagram.
PdSaveChanges	2	Save the current path diagram.

See DiagramScroll Method Example 64

4.5.1.1.2.66 FilePrint Method

Opens a dialog box for printing path diagrams. This method is equivalent to the menu selection **File** → **Print**.

Syntax

```
Pd.FilePrint ()
```

4.5.1.1.2.67 FileRetrieveBackup Method

This method is equivalent to the menu selection **File** → **Retrieve Backup**.

Syntax

```
Pd.FileRetrieveBackup ()
```

4.5.1.1.2.68 FileSave Method

Saves the path diagram as a disk file. This method is equivalent to the menu selection **File** → **Save**.

Syntax

result = `Pd.FileSave ()`

The `FileSave` method syntax has the following parts:

Part	Description
result	True if an error occurs or if the user cancels the operation.

4.5.1.1.2.69 FileSaveAs Method

Saves a path diagram with a new name.

Syntax

result = `Pd.FileSaveAs ()`

result = **Pd.FileSaveAs** (*fileName*)

The **FileSaveAs** method syntax has the following parts:

Part	Description
result	True if an error occurs or if the user cancels the operation.
fileName	Optional new file name. If a file with this name already exists, it is overwritten.

Calling **FileSaveAs** with no arguments is equivalent to the menu selection **File → Save As**.

See GroupSelect Method Example 90

4.5.1.1.2.70 FileSaveAsTemplate Method

Saves your path diagram as a template.

Syntax

result = **Pd.FileSaveAsTemplate** ()
result = **Pd.FileSaveAsTemplate** (*templateFileName*)

The **FileSaveAsTemplate** method syntax has the following parts:

Part	Description
result	True if an error occurs or if the user cancels the operation.
templateFileName	Optional. Name for the template file.

Calling **FileSaveAsTemplate** with no arguments is equivalent to the menu selection **File → Save As Template.**

4.5.1.1.2.71 GetButton Method

Gets a **Button** control in an Amos Graphics window.

Syntax

result = **Pd.GetButton** (*formName*, *controlName*)

The **GetButton** method syntax has the following parts:

Part	Description
result	*An object of type* **System.Windows.Forms.Button**.

formName	(String) The name of an Amos Graphics form.
controlName	(String) The name of a Button on an Amos Graphics form.

To obtain *formName* and *controlName* values for a Button, hold the mouse pointer over the Button and press F1.

4.5.1.1.2.72 GetCheckBox Method

Gets a **CheckBox** control in an Amos Graphics window.

Syntax

result = **Pd.GetCheckBox** (*formName*, *controlName*)

The **GetCheckBox** method syntax has the following parts:

Part	Description
result	*An object of type* **System.Windows.Forms.CheckBox**.
formName	(String) The name of an Amos Graphics form.
controlName	(String) The name of a CheckBox on an Amos Graphics form.

To obtain *formName* and *controlName* values for a CheckBox, hold the mouse pointer over the CheckBox and press F1.

The following plugin displays a messagebox that tells whether there is a check mark next to **Estimate means and intercepts** in the **Analysis Properties** window.

```
Imports Microsoft.VisualBasic
```

```
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      If Pd.GetCheckBox("AnalysisPropertiesForm", "MeansInterceptsCheck").Checked Then
         MsgBox("The checkbox is checked.")
      Else
         MsgBox("The checkbox is not checked.")
      End If
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

The following plugin puts a check mark next to **Estimate means and intercepts** in the **Analysis Properties** window.
Imports Microsoft.VisualBasic

```
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Pd.GetCheckBox("AnalysisPropertiesForm", "MeansInterceptsCheck").Checked = True
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.2.73 GetComboBox Method

Gets a **ComboBox** control in an Amos Graphics window.

Syntax

result = **Pd.GetComboBox** (*formName*, *controlName*)

The **GetComboBox** method syntax has the following parts:

Part	Description
result	*An object of type* **System.Windows.Forms.ComboBox**.

formName	(String) The name of an Amos Graphics form.
controlName	(String) The name of a ComboBox on an Amos Graphics form.

To obtain *formName* and *controlName* values for a ComboBox, hold the mouse pointer over the ComboBox and press F1.

4.5.1.1.2.74 GetControl Method

The **GetControl** method is no longer supported. Use one of the following methods instead.

- GetButton Method 83
- GetCheckBox Method 84
- GetComboBox Method 85
- GetNumericUpDown Method 88
- GetRadioButton Method 88
- GetTextBox Method 89

4.5.1.1.2.75 GetDataFile Method of the Pd class

Gets information about the data file for a single group.

See example 86.

```
Imports System
Imports Microsoft.VisualBasic
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim dbformat As PXMLPersist.CDataTable.cDatabaseFormat
      Dim FileName As String
      Dim TableName As String
      Dim GroupingVariable As String
      Dim GroupingValue As Object
      Dim igroup As Integer
      Dim message As String
      For igroup = 1 To Pd.ngroups
         Pd.GetDataFile(igroup, dbformat, FileName, TableName, GroupingVariable, GroupingValue)
         message &= vbcrlf & "The data file for group " & igroup & " is " & FileName
         message &= vbcrlf & "The table name is " & TableName & ""
      Next
      MsgBox(message,, "GetDataFile Example")
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.2.76 GetModels Method

Gets information about the models specified in Amos Graphics.

Syntax

Pd.GetModels (*sNames*, *sConstraints*)

The **GetModels** method syntax has the following parts:

Part	Description
sNames	A string array that contains the model names.
sConstraints	A string array. Each element of the array contains the constraints of a single model.

See example 87.

The following plugin displays all model names and the constraints imposed by each model.

```
Imports System.Diagnostics
Imports Microsoft.VisualBasic
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim i As Long
      Dim NModels As Long
      Dim SNames() As String
      Dim SConstraints() As String
      Pd.GetModels(SNames, SConstraints)
      NModels = UBound(SNames)

      Dim message As String
      Debug.WriteLine("Number of models = " & NModels)
      For i = 1 To NModels
         message &= "Name: " & SNames(i)
         message &= vbCrLf
         message &= "Constraints: " & SConstraints(i)
         message &= vbCrLf & vbCrLf
      Next
      MsgBox(message)
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.2.77 GetNGroups Method

Gets the number of groups.

Syntax

result = **Pd.GetNGroups** ()

The **GetNGroups** method syntax has the following parts:

Part	Description
result	The number of groups.

4.5.1.1.2.78 GetNumericUpDow n Method

Gets a **NumericUpDown** control in an Amos Graphics window.

Syntax

result = **Pd.GetNumericUpDown** (*formName*, *controlName*)

The **GetNumericUpDown** method syntax has the following parts:

Part	Description
result	*An object of type* **System.Windows.Forms.NumericUpDown**.
formName	(String) The name of an Amos Graphics form.
controlName	(String) The name of a NumericUpDown control on an Amos Graphics form.

To obtain *formName* and *controlName* values for a NumericUpDown control, hold the mouse pointer over the NumericUpDown control and press F1.

4.5.1.1.2.79 GetRadioButton Method

Gets a **RadioButton** control in an Amos Graphics window.

Syntax

result = **Pd.GetRadioButton** (*formName*, *controlName*)

The **GetRadioButton** method syntax has the following parts:

Part	Description

result	*An object of type* **System.Windows.Forms.RadioButton**.
formName	(String) The name of an Amos Graphics form.
controlName	(String) The name of a RadioButton on an Amos Graphics form.

To obtain *formName* and *controlName* values for a RadioButton, hold the mouse pointer over the RadioButton and press F1.

4.5.1.1.2.80 GetTextBox Method

Gets a **TextBox** control in an Amos Graphics window.

Syntax

result = **Pd.GetTextBox** (*formName*, *controlName*)

The **GetTextBox** method syntax has the following parts:

Part	Description
result	*An object of type* **System.Windows.Forms.TextBox**.
formName	(String) The name of an Amos Graphics form.
controlName	(String) The name of a TextBox on an Amos Graphics form.

To obtain *formName* and *controlName* values for a TextBox, hold the mouse pointer over the TextBox and press F1.

4.5.1.1.2.81 GlobalShow Menu Method

This method is no longer supported.

4.5.1.1.2.82 GlobalShow Tools Method

This method is no longer supported.

4.5.1.1.2.83 GroupSelect Method

Selects a group. This method is equivalent to selecting a group in the group list of the Amos Graphics window.

Syntax

Pd.GroupSelect (*groupNameOrNumber*)

The **GroupSelect** method syntax has the following parts:

Part	Description
groupNameOrNumber	A group name, or a group number (where group number 1 is the first group).

See example 90.

The following plugin opens the file **Ex11-ab.amw**, renames the group **Girls** to **Female**, deletes the group **Boys**, and saves the result in the file **Temp.amw**. (If the file **Temp.amw** already exists in the examples directory, it is overwritten.)

```
Imports Microsoft.VisualBasic
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      If FileOpen(AmosDir & "Examples\English\Ex11-ab") Then
         Return 0
      End If

      GroupSelect("Girls")

      ModelFitManageGroupsRename("Female")

      GroupSelect("Boys")

      ModelFitManageGroupsDelete()
      FileSaveAs(AmosDir & "Examples\English\Temp.amw")
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.2.84 HelpAmosOnTheWeb Method

Uses your web browser to visit the Amos website. This method is equivalent to the menu selection **Help → Amos on the Web**.

Syntax

```
Pd.HelpAmosOnTheWeb ()
```

4.5.1.1.2.85 HelpContents Method

Opens a help window and displays the table of contents for Amos's online help system. This method is equivalent to the menu selection **Help → Contents**.

Syntax

Pd.HelpContents ()

4.5.1.1.2.86 HighlightArrows Method

Makes the arrows in the path diagram change color when the mouse pointer approaches. (Other path diagram objects do not change color.) Use this method to let the user click an arrow.

Syntax

Pd.HighlightArrows ()

4.5.1.1.2.87 HighlightNothing Method

Prevents the objects in the path diagram from changing color when the mouse pointer approaches.

Syntax

Pd.HighlightNothing ()

4.5.1.1.2.88 InterfacePropertiesApplyClick Method

This method has no effect and is deprecated.

Syntax

result = **Pd.InterfacePropertiesApplyClick**

The **InterfacePropertiesApplyClick** method syntax has the following parts:

Part	Description
result	(Integer) 0 if no error.

4.5.1.1.2.89 InvalidateOutput Method

Disables the toolbox buttons and menu items that are used for displaying results. If an output path diagram is visible, it is replaced by the input path diagram. Use this method after your program has made a change that makes previously generated output invalid.

Syntax

Pd.InvalidateOutput ()

4.5.1.1.2.90 IsDirtyAmp Method

Gets a value that tells how much the model has changed since the last time it was fitted.

Syntax

result = **Pd.IsDirtyAmp** ()

The **IsDirtyAmp** method syntax has the following parts:

Part	Description
result	0 = No change. 1 = There have been cosmetic changes (e.g., an object has been moved). 2 = Parameter constraints have changed. 3 = The model has change structurally (e.g., an object has been added)

4.5.1.1.2.91 IsDirtyAmw Method

Gets a value that tells how much the model has changed since the last time it was saved as an AMW file.

Syntax

result = **Pd.IsDirtyAmw** ()

The **IsDirtyAmw** method syntax has the following parts:

Part	Description
result	0 = No change. 1 = There have been cosmetic changes (e.g., an object has been moved). 2 = Parameter constraints have changed. 3 = The model has change structurally (e.g., an object has been added)

4.5.1.1.2.92 ModelAdd Method

Adds a new model.

Syntax

Pd.ModelAdd (*modelName*, *modelConstraints*)

The **ModelAdd** method syntax has the following parts:

Part	Description
modelName	A name for the new model.

modelConstraints	Equality constraints on parameters, separated by semicolons. For example, "a=b=c" or "a=1;b=0;c=d=f".

4.5.1.1.2.93 ModelDelete Method

Deletes the model that is currently selected in the Amos Graphics window. This method is equivalent to pressing the **Delete** button in the **Manage Models** dialog.

Syntax

Pd.ModelDelete ()

See ModelSelect Method Example 94

4.5.1.1.2.94 ModelRedefine Method

Changes the name and the definition of an existing model.

Syntax

Pd.ModelRedefine (*modelNumber, modelName, modelConstraints*)

The **ModelRedefine** method syntax has the following parts:

Part	Description
modelNumber	A model number. Model number 1 is the first model.
modelName	A new name for the model specified by *modelNumber*.
modelConstraints	New parameter constraints for the model specified by *modelNumber*. For example, "a=b=c" or "a=1;b=0;c=d=f".

4.5.1.1.2.95 ModelSelect Method

Selects a model in the panel at the left side of the Amos Graphics window.

Syntax

Pd.ModelSelect (*modelNumber*)
Pd.ModelSelect (*modelName*)

The **ModelSelect** method syntax has the following parts:

Part	Description
modelNumber	A model number. Model number 1 is the first model.

modelName	A model name.

See example 94.

The following plugin opens the file **Ex11-ab.amw**, deletes **Model A**, and saves the result in the file **Temp.amw**. (If the file **Temp.amw** already exists in the examples directory, it is overwritten.)

```
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      FileOpen(AmosDir & "Examples\English\Ex11-ab")

      ModelSelect("Model A")
      ModelFitManageModelsDelete()
      popallbuttons()

      FileSaveAs(AmosDir & "Examples\English\Temp.amw")
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.2.96 Observed Method

Draws a rectangle to represent an observed variable.

Syntax

result = **Pd.Observed** (*name*)
result = **Pd.Observed** (*name, meanOrInterceptValue, varianceValue*)
result = **Pd.Observed** (*name, meanOrInterceptName, varianceValue*)
result = **Pd.Observed** (*name, meanOrInterceptValue, varianceName*)
result = **Pd.Observed** (*name, meanOrInterceptName, varianceName*)
result = **Pd.Observed** (*name, x, y, width, height*)
result = **Pd.Observed** (*name, meanOrInterceptValue, varianceValue, x, y, width, height*)
result = **Pd.Observed** (*name, meanOrInterceptName, varianceValue, x, y, width, height*)
result = **Pd.Observed** (*name, meanOrInterceptValue, varianceName, x, y, width, height*)
result = **Pd.Observed** (*name, meanOrInterceptName, varianceName, x, y, width, height*)

The **Observed** method syntax has the following parts:

Part	Description

result	(*Object of type* **PDElement**) A rectangle in the path diagram that represents an observed variable in the model.
name	(*String*) The name of the observed variable.
meanOrInterceptValue	(*Double*) The value of the variable's mean (if the variable is exogenous) or intercept (if the variable is endogenous). This argument should be supplied only when means and intercepts are explicit model parameters.
meanOrInterceptName	(*String*) The name of the variable's mean (if the variable is exogenous) or intercept (if the variable is endogenous). If means and intercepts are not explicit model parameters, *meanOrInterceptName* should be the empty string.
varianceValue	(*Double*) The value of the variable's variance. *varianceValue* should be supplied as an argument to the **Observed** method only when creating an exogenous variable.
varianceName	(*String*) A name for the variable's variance. *varianceName* should be supplied as an argument to the **Observed** method only when creating an exogenous variable.
x	(*Double*) The distance from the left edge of the path diagram to the center of the rectangle, measured in units of 1/96 inch.
y	(*Double*) The distance from the top edge of the path diagram to the center of the rectangle, measured in units of 1/96 inch.
width	(*Double*) The rectangle's width, measured in units of 1/96 inch.
height	(*Double*) The rectangle's height, measured in units of 1/96 inch.

Remarks

If you do not specify a name or value for a parameter, the parameter will be unconstrained. If you do not specify a height, width, x-coordinate and y-coordinate, the rectangle will be assigned an arbitrary height and width and a random x-coordinate and y-coordinate.

4.5.1.1.2.97 Path Method

Draws a single-headed arrow.

Syntax

result = **Pd.Path** (*toVariable, fromVariable*)

result = **Pd.Path** (*toVariable, fromVariable, regressionWeightValue*)

result = **Pd.Path** (*toVariable, fromVariable, regressionWeightName*)

result = **Pd.Path** (*toVariableName, fromVariableName*)

result = **Pd.Path** (*toVariableName, fromVariableName, regressionWeightValue*)

result = **Pd.Path** (*toVariableName, fromVariableName, regressionWeightName*)

The **Path** method syntax has the following parts:

Part	Description
result	(*Object of type* **PDElement**) The new single-headed arrow.
toVariable	(*Object of type* **PDElement**) The variable that the new single-headed arrow points to.
fromVariable	(*Object of type* **PDElement**) The variable that the new single-headed arrow points away from.
toVariableName	*(String)* The name of the variable that the new single-headed arrow points to.
fromVariableName	*(String)* The name of the variable that the new single-headed arrow points away from.
regressionWeightValue	(*Double*) The value of the regression weight represented by the new single-headed arrow.
regressionWeightName	(*String*) A name for the regression weight represented by the new single-headed arrow.

Remarks

If you do not specify a name or value for the regression weigh trepresented by the single-headed arrow, the regression weight will be unconstrained.

4.5.1.1.2.98 PDE Method

Gets an element of a path diagram — a rectangle, ellipse, arrow or figure caption.

Syntax

result = **Pd.PDE** (*theElement*)
result = **Pd.PDE** (*elementNumber*)
result = **Pd.PDE** (*variableName*)

Syntax 2

result = **Pd.PDE** (*theElement1, theElement2*)
result = **Pd.PDE** (*elementNumber1, elementNumber2*)
result = **Pd.PDE** (*variableName1, variableName2*)

The **PDE** method syntax 1 has the following parts:

Part	Description
result	An object of type **PDElement**.
theElement	An object of type **PDElement**.
elementNumber	(Integer) An object number. Objects in a path diagram are numbered starting with 1.
variableName	(String) A variable name.

The **PDE** method syntax 2 has the following parts:

Part	Description
result	A single-headed or double-headed arrow (an object of type **PDElement**.) If the two variables (rectangles or ellipses) specified as function arguments are connected by a double-headed arrow, *result* is set equal to that double-headed arrow. If there is a single-headed arrow that points from the first variable to the second variable, *result* is set equal to that single-headed arrow. Otherwise, *result* is set equal to nothing (null).

theElement1 *theElement2*	Objects of type **PDElement** that are both variables (rectangles or ellipses).
elementNumber1 *elementNumber2*	Two integers that specify variables (rectangles or ellipses). Objects in a path diagram are numbered starting with 1.
variableName1 *variableName2*	The names of two variables.

See example 98.

The following plugin opens the path diagram, Ex08.amw, and changes the color of two variables and one path to yellow. The UndoToHere 106 / UndoResume 105 method pair allows you to undo the changes by pressing  .

```
Imports Microsoft.VisualBasic
Imports Amos
Imports Amos.Pd
Imports System.drawing
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim V1 As PDElement
      Dim V2 As PDElement
      Dim P As PDElement
      If FileOpen(AmosDir & "Examples\English\Ex08") Then
         Return 0
      End If
      UndoToHere()

      V1 = PDE("spatial")
      V2 = PDE("visperc")
      P = PDE(V1, V2)

      V1.BorderColor = Color.Yellow.ToArgb
      V2.BorderColor = Color.Yellow.ToArgb
      P.BorderColor = Color.Yellow.ToArgb

      DiagramRedraw Diagram()
      UndoResume()
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.2.99 PluginsPlugins Method

Opens the **Plugins** dialog for creating, running, editing and deleting plugins. This method is equivalent to the menu selection **Plugins** → **Plugins**.

Syntax

Pd.PluginsPlugins ()

4.5.1.1.2.100 PopAllButtons Method

Pops all the buttons in the Amos Graphics toolbox.

Syntax

Pd.PopAllButtons ()

4.5.1.1.2.101 PropertyGet Method

Retrieves the value of a property that was previously saved using the PropertySave [100] method.

Syntax

result = **Pd.PropertyGet** (*propertyName*, *defaultValue*)

The **PropertyGet** method syntax has the following parts:

Part	Description
result	(String) The value of the property called *propertyName*.
propertyName	(String) The name of a property.
defaultValue	A string that is returned as the value of **PropertyGet** when no property with the name *propertyName* exists.

4.5.1.1.2.102 PropertyRemove Method

Removes a property associated with a path diagram or with an element of a path diagram.

Syntax

Pd.PropertyRemove (*propertyName*)

The **PropertyRemove** method syntax has the following parts:

Part	Description
propertyName	A string containing the name of a property.

4.5.1.1.2.103 PropertySave Method

Saves a string that can later be retrieved by name using the PropertyGet[99] method.

Syntax

Pd.PropertySave (*propertyName*, *value*)

The **PropertySave** method syntax has the following parts:

Part	Description
propertyName	A string.
value	A string. A subsequent use of the PropertyGet[99] method that specifies a property name of *propertyName* will return *value*.

4.5.1.1.2.104 Refresh Method

Refreshes the Amos Graphics window. This method can be used after other methods that change the path diagram in order to make sure that the changes are made visible immediately.

Syntax

Pd.Refresh ()

For an example, see DiagramDrawIndicatorVariable Method Example[58].

4.5.1.1.2.105 Reposition Method

The simplest overloads of the Observed[94], Unobserved[106] and Caption methods place objects at random positions in the path diagram. The **Reposition** method attempts to improve the appearance of the path diagram by rearranging objects.

Syntax

Pd.Reposition ()

Remarks

The **Reposition** method does not produce path diagrams of presentation quality. Far from it, in fact. On the other hand, **Reposition** usually improves a path diagram's appearance substantially.

In order to get objects in the path diagram sized and positioned exactly the way you want, you can use one of the following approaches.

Specify a height, width and location each time you use one of the Observed[94], Unobserved[106] or Caption methods.

or

In your plugin use the **Reposition** method to improve the positioning of objects. Then after running your plugin use the drawing tools in the Amos Graphics toolbox to interactively move and resize the objects in the path diagram.

4.5.1.1.2.106 SetControl Method

The **SetControl** method is no longer supported. Use one of the following methods instead.

- GetButton Method 83
- GetCheckBox Method 84
- GetComboBox Method 85
- GetNumericUpDown Method 88
- GetRadioButton Method 88
- GetTextBox Method 89

4.5.1.1.2.107 SetDataFile Method

Specifies the data file for a single group.

Syntax

result = **Pd.SetDataFile** (*groupNumber*, *dbFormat*, *fileName*, *tableName*, *groupingVariable*, *groupingValue*)

The **SetDataFile** method syntax has the following parts:

Part	Description
result	0 if successful.
groupNumber	Group number of the group for which the data file information is specified. The first group is group number 1.
dbFormat	(Integer) A database format specifier, as described in Settings.
fileName	(String) The name of the data file.
tableName	(String) The name of the data table within the data file (for data files that contain multiple data tables).
groupingVariable	(String) The name of the grouping variable. The empty string if there is no grouping variable. AT PRESENT, THIS ARGUMENT IS IGNORED. (Grouping variables are not currently implemented for the **SetDataFile** method.)

groupingValue	(Object) The value of the grouping variable for this group. At present, *groupingValue* is ignored.

Settings

The settings for *dbFormat* are:

Constant	Value	Description
mmDBASE3	0	Dbase III
mmDBASE4	1	Dbase IV
mmDBASE5	2	Dbase V
mmEXCEL3	3	Excel 3
mmEXCEL4	4	Excel 4
mmEXCEL5	5	Excel 5, Excel 7
mmEXCEL97	6	Excel 97, Excel 8
mmFOXPRO20	7	Foxpro 2.0
mmFOXPRO25	8	Foxpro 2.5
mmFOXPRO26	9	Foxpro 2.6
mmLOTUSWK1	11	Lotus *.wk1
mmLOTUSWK3	12	Lotus *.wk3
mmLOTUSWK4	13	Lotus *.wk4
mmAccess	14	Microsoft Access
mmSPSS	18	SPSS Statistics
mmText	19	Text

See example 102.

The following plugin specifies the data file for group number 1. First, it checks to make sure that there are no open modal dialogs that could prevent Amos Graphics from responding correctly.

```
Imports Microsoft.VisualBasic
Imports Amos
Imports PXMLPersist.CDataTable
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim S As String

      'Wait until Amos Graphics can respond
      Do
         S = Pd.CanRespond(True)
         If S = "" Then
            Exit Do
         End If
         If MsgBox(S, vbOKCancel) = vbCancel Then
            Exit Function
         End If
      Loop

      'Change the data file for group number 1
      If Pd.SetDataFile(1, cDatabaseFormat.mmEXCEL97, _
         Pd.AmosDir & "Examples\English\userguide.xls", _
         "grant", "", 0) = 0 Then
         MsgBox("Successful")
      Else
         MsgBox("Unsuccessful")
      End If
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.2.108 SpecifyModel Method

Obtain a model specification and data file name(s) from Amos Graphics.

Syntax

`Pd.SpecifyModel` (*sem*)

The **`SpecifyModel`** property syntax has the following parts:

Part	Description
sem	An object of type **`AmosEngine`**.

Use this method if you want to use Amos Graphics for model specification, but want to perform a nonstandard analysis that cannot be performed in Amos Graphics.

See example 104.

The following plugin creates an AmosEngine instance (called Sem) and uses it to fit the model that is specified in Amos Graphics. Then the plugin displays the minimized value of the discrepancy function.

```
Imports Microsoft.VisualBasic
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim Sem As New AmosEngineLib.AmosEngine

      Pd.SpecifyModel(Sem)

      'The Amos engine now has the model specification,
      'but the model has not yet been fitted.
      'Fit it now using the Amos Engine (not Amos Graphics)
      'and display the discrepancy function.
      Sem.FitModel()
      MsgBox(Sem.Cmin)
      Sem.Dispose()
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.2.109 ToolsGolden Method

Turns on or off the drawing of subsequently drawn rectangles as golden sections. This method is equivalent to the menu selection **Tools** → **Golden**.

Syntax

Pd.ToolsGolden ()

4.5.1.1.2.110 ToolsListFont Method

This method is equivalent to the menu selection **Tools** → **List Font**.

Syntax

Pd.ToolsListFont ()

4.5.1.1.2.111 ToolsOutline Method

Turns on or off the display of variable names, parameter constraints and arrow heads. This method is equivalent to the menu selection **Tools** → **Outline**.

Syntax

Pd.ToolsOutline ()

4.5.1.1.2.112 ToolsSeedManager Method

Opens the **Seed Manager** window. This method is equivalent to the menu selection **Tools** → **Seed Manager**.

Syntax

```
Pd.ToolsSeedManager ()
```

4.5.1.1.2.113 ToolsSmart Method

Turns on or off the preservation of symmetries when objects are moved and resized. This method is equivalent to the menu selection **Tools** → **Smart**.

Syntax

```
Pd.ToolsSmart ()
```

4.5.1.1.2.114 ToolsSquare Method

Turns on or off the drawing of rectangles as square. This method is equivalent to the menu selection **Tools** → **Square**.

Syntax

```
ToolsSquare ()
```

4.5.1.1.2.115 ToolsWriteAProgram Method

Converts the path diagram in the Amos Graphics window into an equivalent Visual Basic program. This method is equivalent to the menu selection **Tools** → **Write a Program**.

Syntax

```
Pd.ToolsWriteAProgram ()
```

4.5.1.1.2.116 UndoResume Method

Enables undo-ing of modifications to the path diagram. Typically, a program that modifies a path diagram begins with UndoToHere 106 and ends with `UndoResume`. Then all modifications made by the program can be undone with one press of the **Undo** button  .

Syntax

```
Pd.UndoResume ()
```

See Use the Amos Graphics classes to change the appearance of latent variables 492

4.5.1.1.2.117 UndoToHere Method

Saves the state of the path diagram. A subsequent Undo restores this state.

After **UndoToHere**, modifications to a path diagram that are normally undo-able, such as drawing or erasing an object, cannot be undone. To restore the ability to undo such modifications use the UndoResume [105] method.

Typically, a program that modifies a path diagram begins with **UndoToHere** and ends with UndoResume [105]. Then all modifications made by the program can be undone with one click of the **Undo** button  .

Syntax

Pd.UndoToHere ()

See Use the Amos Graphics classes to change the appearance of latent variables [492]

4.5.1.1.2.118 Unobserved Method

Draws an ellipse to represent an unobserved variable.

Syntax

result = **Pd.Unobserved** (*name*)
result = **Pd.Unobserved** (*name, meanOrInterceptValue, varianceValue*)
result = **Pd.Unobserved** (*name, meanOrInterceptName, varianceValue*)
result = **Pd.Unobserved** (*name, meanOrInterceptValue, varianceName*)
result = **Pd.Unobserved** (*name, meanOrInterceptName, varianceName*)
result = **Pd.Unobserved** (*name, x, y, width, height*)
result = **Pd.Unobserved** (*name, meanOrInterceptValue, varianceValue, x, y, width, height*)
result = **Pd.Unobserved** (*name, meanOrInterceptName, varianceValue, x, y, width, height*)
result = **Pd.Unobserved** (*name, meanOrInterceptValue, varianceName, x, y, width, height*)
result = **Pd.Unobserved** (*name, meanOrInterceptName, varianceName, x, y, width, height*)

The **Unobserved** method syntax has the following parts:

Part	Description
result	(*Object of type* **PDElement**) An ellipse in the path diagram that represents an unobserved variable in the model.
name	(*String*) The name of the unobserved variable.
meanOrInterceptValue	(*Double*) The value of the variable's mean (if the variable is exogenous) or intercept (if the variable is endogenous).

	This argument should be supplied only when means and intercepts are explicit model parameters.
meanOrInterceptName	(*String*) The name of the variable's mean (if the variable is exogenous) or intercept (if the variable is endogenous). If means and intercepts are not explicit model parameters, *meanOrInterceptName* should be the empty string.
varianceValue	(*Double*) The value of the variable's variance. *varianceValue* should be supplied as an argument to the **Unobserved** method only when creating an exogenous variable.
varianceName	(*String*) A name for the variable's variance. *varianceName* should be supplied as an argument to the **Unobserved** method only when creating an exogenous variable.
x	(*Double*) The distance from the left edge of the path diagram to the center of the ellipse, measured in units of 1/96 inch.
y	(*Double*) The distance from the top edge of the path diagram to the center of the ellipse, measured in units of 1/96 inch.
width	(*Double*) The ellipse's width, measured in units of 1/96 inch.
height	(*Double*) The ellipse's height, measured in units of 1/96 inch.

Remarks

If you do not specify a name or value for a parameter, the parameter will be unconstrained. If you do not specify a height, width, x-coordinate and y-coordinate, the ellipse will be assigned an arbitrary height and width and a random x-coordinate and y-coordinate.

4.5.1.1.2.119 View AnalysisProperties Method

Opens the **Analysis Properties** dialog. This method is equivalent to the menu selection **View** → **AnalysisProperties**.

Syntax

```
Pd.ViewAnalysisProperties ()
```

4.5.1.1.2.120 View FullScreen Method

Displays the Amos Graphics window maximized without a title bar, or displays it normally.

Syntax

Pd.ViewFullScreen (*tf*)

The **ViewFullScreen** method syntax has the following parts:

Part	Description
tf	If *tf* is True, display the Amos Graphics window maximized without a title bar. Otherwise, display it normally.

4.5.1.1.2.121 View InterfaceProperties Method

Opens the **Interface Properties** dialog.

Syntax

Pd.ViewInterfaceProperties ()

4.5.1.1.2.122 View MatrixRepresentation Method

The **ViewMatrixRepresentation** method has no effect. It is provided for compatibility with earlier versions of Amos.

Syntax

Pd.ViewMatrixRepresentation ()

4.5.1.1.2.123 View ObjectProperties Method

Opens the **Object Properties** dialog.

Syntax

Pd.ViewObjectProperties ()

Pd.ViewObjectProperties (*theElement*)

Pd.ViewObjectProperties (*elementNumber*)

Pd.ViewObjectProperties (*variableName*)

The **ViewObjectProperties** method syntax has the following parts:

Part	Description

theElement	When the **Object Properties** dialog opens it displays the properties of *theElement*.
elementNumber	When the **Object Properties** dialog opens it displays the properties of object number *elementNumber*. (Objects in a path diagram are numbered starting with 1.)
variableName	When the **Object Properties** dialog opens it displays the properties of the ellipse or rectangle that represents the variable called *variableName*.

4.5.1.1.2.124 View Parameters Method

Displays a list of model parameters.

Syntax

```
Pd.ViewParameters ()
```

4.5.1.1.2.125 View SwitchToOtherView

Toggles between the **Path diagram** view and the **Tables** view. This method is equivalent to the menu selection **View** → **Switch to Other View**.

Syntax

```
Pd.ViewSwitchToOtherView ()
```

4.5.1.1.2.126 View TextOutput Method

Displays the text output from an analysis. This method is equivalent to the menu selection **View** → **Text Output**.

Syntax

```
Pd.ViewTextOutput ()
```

4.5.1.1.2.127 View VariablesInDataset Method

Displays a list of the variables in the data file.

Syntax

```
Pd.ViewVariablesInDataset ()
```

4.5.1.1.2.128 ViewVariablesInModel Method

Displays a list of all variables in the model.

Syntax

Pd.ViewVariablesInModel ()

4.5.1.1.2.129 GetWindow Method

Gets the Amos Graphics window.

Syntax

result = **Pd.GetWindow** ()

The **Getindow** method syntax has the following parts:

Part	Description
result	The Amos Graphics window, of type System.Windows.Window.

4.5.1.1.2.130 XYObject Method

Gets the object with coordinates (*x*, *y*).

Syntax

result = **Pd.XYObject** (*x*, *y*)

The **XYObject** method syntax has the following parts:

Part	Description
result	The object (of type **PDElement**) with coordinates (x, y). If there is no object at (x, y), *result* = **nothing**.
x, *y*	The coordinates of a point. x is expressed in inches from the left margin. y is expressed in inches from the top margin.

See MouseDown and MouseUp Events Example 121

4.5.1.1.3 Events

This section documents the events of the Pd class.

4.5.1.1.3.1 AboutToShow MsgBox Event

Event that occurs just before Amos Graphics displays certain message boxes.

Syntax

Pd.AboutToShowMsgBox (*messageID*, *prompt*, *theMessageBoxResult*)

The **AboutToShowMsgBox** method syntax has the following parts:

Part	Description
messageID	(Integer) A number that identifies the message that is about to be displayed. At present, messageID=1 is the only value that occurs.
prompt	The text that is about to be displayed in a message box.
theMessageBoxResult	(Of type Windows.MessageBoxResult) If you set *theMessageBoxResult*=None, Amos Graphics will display the message box. If you set *theMessageBoxResult* to a value other than None, Amos Graphics will not display the message box. Instead, Amos Graphics will execute the code that would have been executed if the message box had been displayed and the user had clicked the response button corresponding to *theMessageBoxResult*.

Settings

The settings for *messageID* are:

Value	Description
1	"Do you want to save your work?..."

See example 111.

The following plugin suppresses the **Do you want to save your work** message box. Because the AboutToShowMsgBox event handler sets **Handled=vbYes**, Amos Graphics does not display the message box, but instead continues as though the user had clicked the **Yes** button.

```
Imports Amos
Imports System.Windows.Forms
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      AddHandler Pd.AboutToShowMsgBox, AddressOf AboutToShowMsgBox
      MessageBox.Show("AboutToShowMsgBox Event Example is now installed.")
   End Function

   Private Sub AboutToShowMsgBox(ByVal MessageID As Integer, ByVal Prompt As String, ByRef Handled As DialogResult)
      If MessageID = 1 Then
         'Do you want to save your work?
         Handled = DialogResult.Yes
      End If
   End Sub

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.3.2 AmwFileRead Event

Event that occurs after Amos Graphics has finished reading a path diagram (*.amw) file.

Syntax

`Pd.AmwFileRead` (*fileName*)

The **`AmwFileRead`** method syntax has the following parts:

Part	Description
fileName	(String) Name of the file that was read.

See example 112.

Running the following plugin creates an AmwFileRead event handler. Each time Amos Graphics reads an AMW file, the event handler displays the file's name in a message box.

```
Imports Microsoft.VisualBasic
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
  Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    AddHandler Pd.AmwFileRead, AddressOf Amw FileRead
    MsgBox("Amw FileRead Event Example is now  installed.")
  End Function

  Private Sub Amw FileRead(ByVal FileName As String)
    MsgBox(FileName & " has just been read.")
  End Sub

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.5.1.1.3.3 Amw FileWritten Event

Event that occurs after Amos Graphics has finished writing a path diagram (*.amw) file.

Syntax

`Pd.AmwFileWritten` (*fileName*, *isTemplate*)

The **`AmwFileWritten`** method syntax has the following parts:

Part	Description
fileName	(String) Name of the file that was written.
isTemplate	(boolean) True if the file was written as a template (*.amt) file. False if it was written as an ordinary path diagram (*.amw) file.

See example 113.

Running the following plugin creates an AmwFileWritten event handler. Each time Amos Graphics writes an AMW file, the event handler displays the file's name in a message box.

```
Imports Microsoft.VisualBasic
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
  Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    AddHandler Pd.AmwFileWritten, AddressOf Amw FileWritten
    MsgBox("Amw FileWritten Event Example is now  installed.")
  End Function

  Private Sub Amw FileWritten(ByVal FileName As String, ByVal IsTemplate As Boolean)
    If IsTemplate Then
      MsgBox(FileName & " has just been w ritten as a template file.")
    Else
      MsgBox(FileName & " has just been w ritten.")
    End If
  End Sub

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.5.1.1.3.4 ButtonPressed Event

Event that occurs when a toolbox button, or the equivalent menu item, is clicked.

Syntax

`Pd.ButtonPressed` (*buttonNumber*, *handled*)

The **`ButtonPressed`** event syntax has the following parts:

Part	Description
buttonNumber	An integer that identifies the button that caused the event.
handled	Set *handled*=1 to cause Amos Graphics to perform normal processing of the button click after the return from the **`ButtonPressed`** event. Set *handled*=0 to suppress normal processing of the button click. In other words, set *handled*=1 if you want to substitute your handling of a button click in place of Amos Graphics's normal handling. Set *handled*=0 to perform your handling of a button click before Amos Graphics's normal handling.

Settings

The settings for *buttonNumber* are:

Value	Menu item
38	File->New
95	File->New with Template...
39	File->Open...
70	File->Retrieve Backup...
40	File->Save
41	File->Save As...
96	File->Save As Template...
43	File->Data Files...
44	File->Print...
104	File->Manager...
46	Exit

Value	Menu item
51	Edit->Undo
52	Edit->Redo
53	Edit->Copy (to clipboard)
14	Edit->Select
67	Edit->Select All
68	Edit->Deselect All
80	Edit->Link
9	Edit->Move
10	Edit->Duplicate

5	Edit->Erase
23	Edit->Move Parameter
76	Edit->Reflect
75	Edit->Rotate
24	Edit->Shape of Object
20	Edit->Space Horizontally
21	Edit->Space Vertically
86	Edit->Drag Properties...
37	Edit->Fit to Page
66	Edit->Touch Up

Value	Menu item
87	View->Interface Properties...
88	View->Analysis Properties...
85	View->Object Properties...
89	View->Variables in Model...
90	View->Variables in Dataset...
91	View->Parameters...
92	View->Switch to Other View
42	View->Text Output
60	View->Full Screen

Value	Menu item
1	Diagram->Draw Observed

2	Diagram->Draw Unobserved
3	Diagram->Draw Path
4	Diagram->Draw Covariance
6	Diagram->Figure Caption
77	Diagram->Draw Indicator Variable
78	Diagram->Draw Unique Variable
8	Diagram->Zoom
31	Diagram->Zoom In
32	Diagram->Zoom Out
33	Diagram->Zoom Page
30	Diagram->Scroll
25	Diagram->Loupe
83	Diagram->Redraw diagram

Value	Menu item
7	Analyze->Calculate Estimates
98	Analyze->Stop Calculating Estimates
94	Analyze->Manage Groups...
93	Analyze->Manage Models...
71	Analyze->Modeling Lab...
22	Analyze->Toggle Observed/Unobserved
79	Analyze->Degrees of freedom...
107	Analyze->Specification Search...

108	Analyze->Multiple-Group Analysis...
110	Analyze->Bayesian Estimation...
111	Analyze->Data Imputation...

Value	Menu item
102	Tools->List Font...
74	Tools->Smart
72	Tools->Outline
54	Tools->Square
55	Tools->Golden
109	Tools->Seed Manager...
56	Tools->Customize

Value	Menu item
102	Plugins->Plugins...

Value	Menu item
57	Help->Contents
103	Help->Amos on the Web
58	Help->About Amos

See example 118.

The following plugin prevents the user from drawing observed variables. Running the plugin creates a **ButtonPressed** event handler that is executed each time the user presses a button in the toolbox (or makes a selection from the Amos Graphics menu). If the user presses the **Draw Observed** button, the event handler sets **Handled = True**, preventing Amos Graphics from responding to the button press.

```
Imports Microsoft.VisualBasic
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
  Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    AddHandler Pd.ButtonPressed, AddressOf ButtonPressed
    MsgBox("ButtonPressed Event Example is now installed",, "ButtonPressed Event")
  End Function

  Private Sub ButtonPressed(ByVal ButtonNumber As Integer, ByRef Handled As Boolean)
    If ButtonNumber = 101 Then
      MsgBox("Sorry, you are not allowed to draw boxes.",, "ButtonPressed Event")
      'Suppress Amos Graphics's normal response
      'to the Draw Observed button.
      Handled = True
    Else
      'Allow Amos Graphics to respond in the normal way
      'to the button click.
      Handled = False
    End If
  End Sub

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.5.1.1.3.5 MouseDown Event

Event that occurs when a mouse button is pressed.

Syntax

Pd.MouseDown (*button*, *shift*, *x*, *y*)

The **MouseDown** event syntax has the following parts:

Part	Description
button	An enum of type System.Windows.Forms.MouseButtons that identifies the button that was pressed to cause the event.
shift	Returns an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed. A bit is set if the key is down. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The shift argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none

	of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of shift would be 6.
x, *y*	(Single) The coordinates of the mouse pointer. x is expressed in inches from the left margin. y is expressed in inches from the top margin.

See MouseDown and MouseUp Events Example 121

The following plugin creates a MouseDown event handler that displays a message when the user presses a mouse button. If the user clicks an object such as a rectangle, the event handler displays a description of the object. If the user clicks a region of the path diagram that is not occupied by an object, the event handler displays the message "No object".

```
Imports Amos
Imports Amos.Pd
Imports System.Environment
Imports Microsoft.VisualBasic
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
  Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    AddHandler Pd.MouseDown, AddressOf MouseDown
    MsgBox("MouseDown Event Example is now installed.")
  End Function

  Private Sub MouseDown(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByVal Y As Single)
    Dim message As String
    message &= "x = " & X
    message &= New Line & "y = " & Y
    message &= New Line & New Line

    Dim pde As PDElement = XYObject(X, Y)
    If pde Is Nothing Then
      message &= "No object"
    ElseIf pde.IsObservedVariable Then
      message &= "Rectangle: " & pde.NameOrCaption
    ElseIf pde.IsUnobservedVariable Then
      message &= "Ellipse: " & pde.NameOrCaption
    ElseIf pde.IsPath Then
      message &= "Single-headed arrow"
    ElseIf pde.IsCovariance Then
      message &= "Double-headed arrow"
    ElseIf pde.IsCaption Then
      message &= "Caption: "
      message &= vbCrLf & pde.NameOrCaption
    End If
    MsgBox(message, MsgBoxStyle.MsgBoxSetForeground)
  End Sub

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.5.1.1.3.6 MouseUp Event

Event that occurs when a mouse button is released.

Syntax

Pd.MouseUp (*button*, *shift*, *x*, *y*)

The **MouseUp** event syntax has the following parts:

Part	Description
button	An enum of type System.Windows.Forms.MouseButtons that identifies the button that was pressed to cause the event.
shift	Returns an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the *button* argument is released. A bit is set if the key is down. The *shift* argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The *shift* argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of *shift* would be 6.
x, y	(Single) The coordinates of the mouse pointer. x is expressed in inches from the left margin. y is expressed in inches from the top margin.

See example 121.

The following plugin creates a MouseUp event handler that displays a message when the user releases a mouse button. If the user clicks an object such as a rectangle, the event handler displays a description of the object. If the user clicks a region of the path diagram that is not occupied by an object, the event handler displays the message "No object".

```
Imports Amos
Imports Amos.Pd
Imports System.Environment
Imports Microsoft.VisualBasic
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      AddHandler Pd.MouseUp, AddressOf MouseUp
      MsgBox("MouseUp Event Example is now  installed.")
   End Function

   Private Sub MouseUp(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByVal Y As Single)
      Dim message As String
      message &= "x = " & X
      message &= New Line & "y = " & Y
      message &= New Line & New Line
      Dim pde As PDElement = XYObject(X, Y)

      If pde Is Nothing Then
         message &= "No object"
      ElseIf pde.IsObservedVariable Then
         message &= "Rectangle: " & pde.NameOrCaption
      ElseIf pde.IsUnobservedVariable Then
         message &= "Ellipse: " & pde.NameOrCaption
      ElseIf pde.IsPath Then
         message &= "Single-headed arrow"
      ElseIf pde.IsCovariance Then
         message &= "Double-headed arrow"
      ElseIf pde.IsCaption Then
         message &= "Caption: "
         message &= vbCrLf & pde.NameOrCaption
      End If
      MsgBox(message, MsgBoxStyle.MsgBoxSetForeground)
   End Sub

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.1.3.7 New ObjectCreated Event

Event that occurs after a new object is added to the path diagram.

Syntax

`Pd.NewObjectCreated` (*nObjects*)

The **`NewObjectCreated`** event syntax has the following parts:

Part	Description

nObjects	The number of objects in the path diagram, including the one that was just added.

4.5.1.1.3.8 ObjectEntered Event

Event that occurs when the mouse pointer comes close to an object and the object changes color.

Syntax

Pd.ObjectEntered (*button*, *shift*, *objectNumber*)

The **ObjectEntered** event syntax has the following parts:

Part	Description
button	An enum of type System.Windows.Forms.MouseButtons that gives the state of the mouse buttons when the event occurred.
shift	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys. A bit is set if the key is down. The *shift* argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The *shift* argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of *shift* would be 6.
objectNumber	A number identifying the object that changed color when the mouse pointer came near.

4.5.1.1.3.9 OpenWindowsUpdated Event

Event that occurs when a change in the model or the data requires an update of the following windows.

- Variables in Model
- Variables in Dataset
- Parameters
- Object Properties

These windows are updated automatically. The event is raised so that you can update your own windows at the same time if necessary.

Syntax

Pd.OpenWindowsUpdated ()

4.5.1.1.3.10 OutputIsInvalid Event

Event that occurs when Amos Graphics disables the toolbox buttons and the menu items that are used for displaying results.

Syntax

Pd.OutputIsInvalid ()

4.5.1.1.3.11 PDChanged Event

Event that occurs when Amos Graphics starts a new path diagram or opens an existing path diagram.

Syntax

Pd.PDChanged ()

4.5.1.1.3.12 PostFitResults Event

Event that occurs after model fitting is complete.

Syntax

Pd.PostFitResults (*sem*, *modelName*, *status*)

The **PostFitResults** event syntax has the following parts:

Part	Description
sem	An object of type **AmosEngine**. *sem* is the **AmosEngine** instance used by Amos Graphics. The "Group 3" methods of the *sem* object can be used in the **PostFitResults** event to obtain the results of the analysis (see Timing is Everything 156 and Group 3: Methods for retrieving results 160).
modelName	A string containing the name of the model that has just been fitted.
status	Zero if the model was successfully fitted. Nonzero otherwise.

See Use the Amos Graphics classes to calculate a new fit measure 490

4.5.1.1.3.13 PreFitOptions Event

Event that occurs before model fitting.

Syntax

Pd.PreFitOptions (*sem*)

The **PreFitOptions** event syntax has the following parts:

Part	Description
sem	An object of type **AmosEngine**. *sem* is the **AmosEngine** instance used by Amos Graphics. The "Group 1" methods of the *sem* object can be used in the **PreFitOptions** event to specify options that affect model fitting (see Timing is Everything 156 and Group 1: Declarative methods 157).

See Use the Amos Graphics classes to calculate a new fit measure 490

4.5.1.1.3.14 QueryUnload Event Method

Event that occurs before the Amos Graphics window is closed. If you need to be sure that the Amos Graphics window stays open during some operation, you can trap this event in order to keep the window from closing.

Syntax

Pd.QueryUnload (*cancel*)

The **QueryUnload** event syntax has the following parts:

Part	Description
cancel	(Boolean) Set cancel to True in order to prevent the Amos Graphics window from closing. Otherwise, the window will close.

4.5.1.1.3.15 Quitting Event

Event that occurs when the Amos Graphics window is closed.

Syntax

Pd.Quitting ()

See MouseDown and MouseUp Events Example 121

4.5.1.2 PDElement Class Members

A **PDElement** object is an element of a path diagram — a rectangle, ellipse, arrow or figure caption.

4.5.1.2.1 Properties

This section documents the properties of the PDElement class.

4.5.1.2.1.1 BorderColor Property

Gets or sets the color of lines used to draw arrows and to draw the outlines of rectangles and ellipses.

Syntax

object.**borderColor** [= *color*]

The **borderColor** property syntax has the following parts:

Part	Description
object	A rectangle, ellipse or arrow (an object of type **PDElement).**
color	A 32-bit ARGB value corresponding to a **System.Drawing.Color** structure.

See Use the Amos Graphics classes to change the appearance of latent variables 492

4.5.1.2.1.2 Estimate1 Property

Gets a parameter estimate — either a regression weight, a covariance or a variance.

Syntax

result = *object*.**estimate1**

The **estimate1** property syntax has the following parts:

Part	Description
result	When parameter estimates are displayed on the path diagram in the Amos window, *result* is a parameter estimate associated with object. *result* is a regression weight if object is a single-headed arrow. *result* is a covariance if object is a double-headed arrow. *result* is a variance if object is an exogenous variable.
object	A rectangle, ellipse or arrow (an object of type **PDElement**.)

4.5.1.2.1.3 Estimate2 Property

Gets a parameter estimate — either a mean or an intercept.

Syntax

result = *object*.**estimate2**

The **estimate2** property syntax has the following parts:

Part	Description
result	When parameter estimates are displayed on the path diagram in the Amos window, and when means and intercepts are explicit model parameters (see To estimate means and intercepts), *result* is a parameter estimate associated with object. *result* is an intercept if object is an endogenous variable. *result* is a mean if object is an exogenous variable.
object	A rectangle or ellipse (an object of type **PDElement**.)

4.5.1.2.1.4 FillColor Property

Gets or sets the color of the interior of rectangles and ellipses. The **FillColor** property is ignored if FillStyle [127]=0.

Syntax

object.**FillColor** [= *color*]

The **FillColor** property syntax has the following parts:

Part	Description
object	An object of type **PDElement**.
color	A 32-bit ARGB value corresponding to a System.Drawing.Color structure.

See Use the Amos Graphics classes to change the appearance of latent variables [492]

4.5.1.2.1.5 FillStyle Property

The **FillStyle** property is ignored. It is retained for syntactic compatibility with previous versions of Amos.

Syntax

object.**fillStyle** [= *value*]

The **fillStyle** property syntax has the following parts:

Part	Description
object	A rectangle or ellipse (an object of type **PDElement**.)

value	0 if object is to be filled with the solid color specified by the fillColor 127 property. 1 if object is to be transparent.

4.5.1.2.1.6 Height Property

Gets or sets the height of a rectangle, ellipse or title.

Syntax

object.**Height** *[= value]*

The **Height** property syntax has the following parts:

Part	Description
object	An object of type **PDElement**.
value	The height of *object* in logical pixel units if *object* is a rectangle, ellipse or title. Zero if *object* is an arrow.

See Use the Amos Graphics classes to resize all rectangles 499

4.5.1.2.1.7 InvisibleName Property

Gets or sets a value that is True for a variable whose name is invisible. The value of this property has no effect on the display of parameters, rectangles, ellipses or arrows.

Setting an object's **InvisibleName** property to True has the effect of unchecking **Show name** on the **Visibility** tab of the **Object Properties dialog.**

The **InvisibleName** property is ignored if **Use visibility settings** is unchecked on the **Visibility** tab of the **Object Properties** dialog.

Syntax

object.**InvisibleName** *[= value]*

The **InvisibleName** property syntax has the following parts:

Part	Description
object	A rectangle or ellipse (an object of type **PDElement**.)
value	True if *object*'s name is invisible.

See InvisiblePicture Property Example 130

4.5.1.2.1.8 InvisibleParameters Property

Gets or sets a value that is True for a rectangle, ellipse or arrow whose parameters are invisible. The value of this property has no effect on the display of rectangles, ellipses, arrows or variable names.

Setting an object's **InvisibleParameters** property to True has the effect of unchecking **Show parameters** on the **Visibility** tab of the **Object Properties** dialog.

The **InvisibleParameters** property is ignored if **Use visibility settings** is unchecked on the **Visibility** tab of the **Object Properties** dialog.

Syntax

object.**InvisibleParameters** *[= value]*

The **InvisibleParameters** property syntax has the following parts:

Part	Description
object	An object of type **PDElement**.
value	True if *object*'s parameters are invisible.

See InvisiblePicture Property Example 130

4.5.1.2.1.9 InvisiblePicture Property

Gets or sets a value that is True for a rectangle, ellipse or arrow that is invisible. The value of this property has no effect on the display of parameters or variable names.

Setting an object's **InvisiblePicture** property to True has the effect of unchecking **Show picture** on the **Visibility** tab of the **Object Properties dialog.**

The **InvisiblePicture** property is ignored if **Use visibility settings** is unchecked on the **Visibility** tab of the **Object Properties** dialog.

Syntax

object.**InvisiblePicture** *[= value]*

The **InvisiblePicture** property syntax has the following parts:

Part	Description
object	An object of type **PDElement**.
value	True if *object*'s rectangle, ellipse or arrow is invisible.

See example 130.

The following plugin makes residual variables invisible. Their variances and their variable names are also made invisible. Furthermore, regression weights for paths leading from residual variables are made invisible. The UndoToHere 106 / UndoResume 105 method pair allows you to undo the changes by pressing  .

```
Imports Microsoft.VisualBasic
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim e As PDElement
      Pd.UndoToHere()
      For Each e In Pd.PDElements
         If e.IsUniqueVariable Then
            e.InvisibleName = True
            e.InvisibleParameters = True

            e.InvisiblePicture = True

         ElseIf e.IsPath Then
            If e.Variable1.IsUniqueVariable Then
               e.InvisibleParameters = True
            End If
         End If
      Next
      Pd.DiagramRedraw Diagram()
      Pd.UndoResume()
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.2.1.10 IsHighlighted Property

Gets or sets a value that is True for the object that is highlighted and False for other objects. (By default, the "highlighted" object is colored red.)

Syntax

object.**IsHighlighted** [= *value*]

The **IsHighlighted** property syntax has the following parts:

Part	Description
object	An object of type **PDElement**.
value	True if object is highlighted.

4.5.1.2.1.11 IsSelected Property

Gets or sets a value that determines whether a path diagram element has been "selected". Typically, path diagram elements are selected by using the EditSelect 75 method or the EditSelectAll 76 method.

Syntax

object.**IsSelected** [= *value*]

The **IsSelected** property syntax has the following parts:

Part	Description
object	An object of type **PDElement**.
value	True if object is a selected path diagram object.

See Use the Amos Graphics classes to draw double-headed arrows 496

4.5.1.2.1.12 LongLabel Property

Gets or sets a character string used (instead of a variable's name) to label a rectangle or ellipse.

Syntax

object.**LongLabel** [= *value*]

The **LongLabel** property syntax has the following parts:

Part	Description
object	A rectangle or ellipse (an object of type **PDElement**.)
value	A character string used as a label for *object*.

4.5.1.2.1.13 NameColor Property

Gets or sets the color of variable names and figure captions.

Syntax

object.**NameColor** [= *Color*]

The **NameColor** property syntax has the following parts:

Part	Description
object	An object of type **PDElement**.

color	A 32-bit ARGB value corresponding to a System.Drawing.Color structure.

See Use the Amos Graphics classes to change the appearance of latent variables 492

4.5.1.2.1.14 NameFontBold Property

Gets or sets a value that determines whether a variable name or figure caption is displayed in a bold font.

Syntax

object.**NameFontBold** [= *value*]

The **NameFontBold** property syntax has the following parts:

Part	Description
object	A rectangle, ellipse or figure caption (an object of type **PDElement**.)
value	(Integer) -1 if object's name is displayed in a bold font. 0 otherwise.

See example 132.

The following plugin changes the font used for latent variable names to bold italic. The Undraw 155 and Draw 144 methods make the changes immediately visible. The UndoToHere 106 / UndoResume 105 method pair allows you to undo the changes by pressing  .

```
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim x As PDElement
      UndoToHere()
      For Each x In PDElements
         If x.IsLatentVariable Then
            x.Undraw ()

            x.NameFontBold = True

            x.NameFontItalic = True
            x.Draw ()
         End If
      Next
      UndoResume()
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.2.1.15 NameFontItalic Property

Gets or sets a value that determines whether a variable name or figure caption is displayed in an italic font.

Syntax

object.**NameFontItalic** *[= value]*

The **NameFontItalic** property syntax has the following parts:

Part	Description
object	A rectangle, ellipse or figure caption (an object of type **PDElement**.)
value	(Integer) -1 if object's name is displayed in an italic font. 0 otherwise.

See NameFontBold Property Example 132

4.5.1.2.1.16 NameFontSize Property

Gets or sets the font size used to display a variable name or figure caption.

Syntax

object.**NameFontSize** [= *value*]

The **NameFontSize** property syntax has the following parts:

Part	Description
object	A rectangle, ellipse or figure caption (an object of type **PDElement**.)
value	(Single) The font size used to display object's name.

See Use the Amos Graphics classes to draw a path diagram 494

4.5.1.2.1.17 NameHeight Property

Gets the height of a variable name or figure caption.

Syntax

result = *object*.**NameHeight**

The **NameHeight** property syntax has the following parts:

Part	Description
result	The height, in inches, of object's variable name or figure caption
object	A rectangle, ellipse or figure caption (an object of type **PDElement**.)

See Use the Amos Graphics classes to resize all rectangles 499

4.5.1.2.1.18 NameOrCaption Property

Gets or sets a figure caption or a variable name.

Syntax

object.**NameOrCaption** [= *value*]

The **NameOrCaption** property syntax has the following parts:

Part	Description
object	A rectangle, ellipse or figure caption (an object of type **PDElement**.)
value	A figure caption or variable name.

See Use the Amos Graphics classes to draw a path diagram 494

4.5.1.2.1.19 NameWidth Property

Gets the width of a variable name or figure caption.

Syntax

result = *object*.**NameWidth**

The **NameWidth** property syntax has the following parts:

Part	Description
result	The width, in inches, of *object*'s variable name or figure caption
object	A rectangle, ellipse or figure caption (an object of type **PDElement**.)

See Use the Amos Graphics classes to resize all rectangles 499

4.5.1.2.1.20 OriginX Property

Gets or sets the *X* coordinate of the center of a rectangle or ellipse.

Syntax

object.**OriginX** [= *value*]

The **OriginX** property syntax has the following parts:

Part	Description
object	A rectangle or ellipse (an object of type **PDElement**.)
value	(Single) The distance, in inches, of object's center from the left margin.

See DiagramZoom Method Example 64

4.5.1.2.1.21 OriginY Property

Gets or sets the *Y* coordinate of the center of a rectangle or ellipse.

Syntax

object.**OriginY** [= *value*]

The **OriginY** property syntax has the following parts:

Part	Description
object	A rectangle or ellipse (an object of type **PDElement**.)

value	(Single) The distance, in inches, of object's center from the top margin.

See DiagramZoom Method Example [64]

4.5.1.2.1.22 ParameterColor Property

Gets or sets the color of parameter constraints and parameter estimates.

Syntax

object.**ParameterColor** [= *color*]

The **ParameterColor** property syntax has the following parts:

Part	Description
object	An object of type **PDElement**.
color	A 32-bit ARGB value corresponding to a System.Drawing.Color structure.

See Use the Amos Graphics classes to change the appearance of latent variables [492]

4.5.1.2.1.23 ParameterFontBold Property

Gets or sets a value that determines whether a variable's parameter constraints and parameter estimates are displayed in a bold font.

Syntax

object.**ParameterFontBold** *[= value]*

The **ParameterFontBold** property syntax has the following parts:

Part	Description
object	A rectangle, ellipse or arrow (an object of type **PDElement**.)
value	(Integer) -1 if parameter constraints and parameter estimates associated with object are displayed in a bold font. 0 otherwise.

See example [136].

The following plugin changes the font used for regression weights to bold italic. The Undraw [155] and Draw [144] methods make the changes immediately visible. The UndoToHere [106] / UndoResume [105] method pair allows you to undo the changes by pressing  .

```
Imports Microsoft.VisualBasic
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim e As PDElement
      UndoToHere()
      For Each e In PDElements
         If e.IsPath Then
            e.ParameterFontBold = True
            e.ParameterFontItalic = True
         End If
      Next
      Pd.Refresh()
      UndoResume()
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.2.1.24 ParameterFontItalic Property

Gets or sets a value that determines whether a variable's parameter constraints and parameter estimates are displayed in an italic font.

Syntax

object.**ParameterFontItalic** *[= value]*

The **ParameterFontItalic** property syntax has the following parts:

Part	Description
object	A rectangle, ellipse or arrow (an object of type **PDElement**.)
value	(Integer) -1 if parameter constraints and parameter estimates associated with object are displayed in an italic font. 0 otherwise.

See ParameterFontBold Property Example 136

4.5.1.2.1.25 ParameterFontSize Property

Gets or sets the font size used to display a variable's parameter constraints and parameter estimates.

Syntax

object.**ParameterFontSize** *[= value]*

The **ParameterFontSize** property syntax has the following parts:

Part	Description
object	A rectangle, ellipse or arrow (an object of type **PDElement**.)
value	(Single) The font size used to display parameter constraints and parameter estimates associated with object.

See Use the Amos Graphics classes to draw a path diagram 494

4.5.1.2.1.26 ParameterFormat Property

Gets or sets the parameter formats for this object. These formats override those specified with in the **View Interface Properties** dialog. The **ParameterFormat** property can be modified interactively using the **Format** tab of the **Object Properties** dialog.

Syntax

object.**ParameterFormat** (*formatName*, *formatType*)[= *value*]

The **ParameterFormat** property syntax has the following parts:

Part	Description
object	A rectangle, ellipse or arrow (an object of type **PDElement.**).
formatName	A string containing the name of a parameter format — either "Unstandardized estimates", "Standardized estimates" or the name of a user-defined format as defined on the Amos Graphics **View** → **Interface Properties** → **Formats** tab. See To create a new format.
formatType	A constant, as described in Settings, specifying a parameter type.
value	A parameter format containing the format descriptors x.xx, y.yy and z.zz, in addition to any labels. Use x.xx for unstandardized regression weights, variances and covariances, y.yy for mean and intercept parameters, and z.zz for standardized regression weights, correlations and squared multiple correlations. To increase the number of decimal digits on the display, use format descriptors with additional trailing characters, e.g., x.xxx for 3-digit precision. A detailed description of parameter formats can be found in the Amos Graphics online help, under To create a new format.

Settings

The settings for *formatType* are:

Constant	Value	Description
pdExogenousNoMeans	0	When means are not estimated, *value* is the format for the variance of an exogenous variable.
pdExogenousMeans	1	When means are estimated, *value* is the format for the mean and variance of an exogenous variable.
pdEndogenousNoMeans	2	When means are not estimated, *value* is the format for an endogenous variable. (An endogenous variable has no parameters associated with it when means are not estimated. However, *value* is displayed on the path diagram as a constant string.)
pdEndogenousMeans	3	When means and intercepts are estimated, *value* is the format for the intercept associated with an endogenous variable.
pdPaths	4	*value* is the format for a regression weight.
pdCovariances	5	*value* is the format for a covariance.

See example 139.

The following plugin modifies the format for displaying estimates of means and variances of unique variables. When means are not estimated, variances are displayed with three decimal places of precision. When means are estimated, both means and variances are displayed with three decimal places of precision.

```
Imports Microsoft.VisualBasic
Imports Amos
Imports Amos.PDElement.PDFormatType
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
  Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    Dim x As PDElement
    For Each x In Pd.PDElements
      If x.IsUniqueVariable Then

        x.ParameterFormat("Unstandardized Estimates", pdExogenousMeans) = _
        "y.yyy, x.xxx"
        x.ParameterFormat("Unstandardized Estimates", pdExogenousNoMeans) = "x.xxx"

      End If
    Next
  End Function

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.5.1.2.1.27 ParameterOrientation Property

Gets or sets a value that determines the orientation of parameter constraints and estimates.

Syntax

object.**ParameterOrientation** *[= value]*

The **ParameterOrientation** property syntax has the following parts:

Part	Description
object	A rectangle, ellipse or arrow (an object of type **PDElement**.)
value	0=Horizontal; 1=Oblique; 2=Oblique, Inverted.

See example 140.

The following plugin changes the orientation of regression weights to oblique. The UndoToHere 106 / UndoResume 105 method pair allows you to undo the changes by pressing  .

```
Imports Microsoft.VisualBasic
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim E As PDElement
      UndoToHere()
      For Each E In PDElements
         If E.IsPath Then
            E.ParameterOrientation = 1
         End If
      Next
      DiagramRedraw Diagram()
      UndoResume()
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.2.1.28 Penwidth Property

Gets or sets the width of lines used to draw rectangles, ellipses and arrows.

Syntax

object.**Penwidth** [= *value*]

The **Penwidth** property syntax has the following parts:

Part	Description
object	A rectangle, ellipse or arrow (an object of type **PDElement**.)
value	Of type integer, the width (in units of 1/96 inch) of lines used to draw rectangles, ellipses and arrows.

See Use the Amos Graphics classes to change the appearance of latent variables 492

4.5.1.2.1.29 TermX Property

For rectangles, gets or sets the *X* coordinate of the lower right corner. For ellipses, gets or sets the *X* coordinate of the lower right corner of the bounding rectangle.

Syntax

object.**TermX** [= *value*]

The **TermX** property syntax has the following parts:

Part	Description
object	A rectangle or ellipse (an object of type **PDElement**.)
value	If object is a rectangle, *value* is the X coordinate of its lower right corner. If object is an ellipse, *value* is the X coordinate of the lower right corner of its bounding rectangle.

4.5.1.2.1.30 TermY Property

For rectangles, gets or sets the *Y* coordinate of the lower right corner. For ellipses, gets or sets the *Y* coordinate of the lower right corner of the bounding rectangle.

Syntax

object.**TermY** [= *value*]

The **TermY** property syntax has the following parts:

Part	Description
object	A rectangle or ellipse (an object of type **PDElement**.)
value	If object is a rectangle, *value* is the Y coordinate of its lower right corner. If object is an ellipse, *value* is the Y coordinate of the lower right corner of its bounding rectangle.

4.5.1.2.1.31 Value1 Property

Gets or sets a parameter name or fixed parameter value for a regression weight, covariance or variance.

Syntax

object.**Value1** [= *value*]

The **Value1** property syntax has the following parts:

Part	Description
object	A rectangle, ellipse or arrow (an object of type **PDElement**.)
value	(String) A parameter name or a string representation of a numeric constant.

See Use the Amos Graphics classes to draw a path diagram 494

4.5.1.2.1.32 Value2 Property

Gets or sets a parameter name or fixed parameter value for a mean or intercept.

Syntax

object.**Value2** [= *value*]

The **Value2** property syntax has the following parts:

Part	Description
object	A rectangle or ellipse (an object of type **PDElement**.)
value	(String) A parameter name or a string representation of a numeric constant.

4.5.1.2.1.33 Variable1 Property

Gets a rectangle or an ellipse (an object of type **PDElement**.) For a single-headed arrow, **Variable1** is the object that the arrow points away from. For a double-headed arrow, **Variable1** is one of the objects that the arrow points to.

Syntax

result = *object*.**Variable1**

The **Variable1** property syntax has the following parts:

Part	Description
result	A rectangle or an ellipse (an object of type **PDElement**.) If object is a single-headed arrow, *result* is the variable that it points away from. If object is a double-headed arrow, *result* is one of the variables that it points to.
object	A single-headed arrow or a double-headed arrow (an object of type **PDElement**.)

4.5.1.2.1.34 Variable2 Property

Gets a rectangle or an ellipse (an object of type **PDElement**.) For a single-headed arrow, **Variable2** is the object that the arrow points to. For a double-headed arrow, **Variable2** is one of the objects that the arrow points to.

Syntax

result = *object*.**Variable2**

The **Variable2** property syntax has the following parts:

Part	Description
result	A rectangle or an ellipse (an object of type **PDElement**.) If object is a single-headed arrow, *result* is the variable that it points to. If object is a double-headed arrow, *result* is one of the variables that it points to.
object	A single-headed arrow or a double-headed arrow (an object of type **PDElement**.)

4.5.1.2.1.35 Width Property

Gets or sets the width of rectangle, ellipse or title.

Syntax

object.**Width** [= *value*]

The **Width** property syntax has the following parts:

Part	Description
object	An object of type **PDElement**.
value	The width of *object* in logical pixel units if *object* is a rectangle, ellipse or title. Zero if *object* is an arrow.

See Use the Amos Graphics classes to resize all rectangles 499

4.5.1.2.2 Methods

This section documents the methods of the PDElement class.

4.5.1.2.2.1 Draw Method

Draws an object.

Syntax

object.**Draw** ()

The **Draw** method syntax has the following parts:

Part	Description
object	An object of type **PDElement**.

See Use the Amos Graphics classes to change the appearance of latent variables 492

4.5.1.2.2.2 IsCaption Method

Returns True for figure captions.

Syntax

result = *object*.**IsCaption** ()

The **IsCaption** method syntax has the following parts:

Part	Description
result	True if *object* is a figure caption.
object	An object of type **PDElement**.

See MouseDown and MouseUp Events Example 121

4.5.1.2.2.3 IsCovariance Method

Returns True for double-headed arrows (covariances).

Syntax

result = *object*.**IsCovariance** ()

The **IsCovariance** method syntax has the following parts:

Part	Description
result	True if *object* is a double-headed arrow.
object	An object of type **PDElement**.

See example 145.

When you run this plugin, double-headed arrows in the path diagram are redrawn using a pen that is 3/96 inch wide. All other objects are redrawn using a pen that is 1/96 inch wide. The UndoToHere 106 / UndoResume 105 method pair allows you to undo the changes by pressing  .

```
Imports Microsoft.VisualBasic
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
  Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    Dim E As PDElement
    UndoToHere()
    For Each E In PDElements

      If E.IsCovariance Then
        E.Penwidth = 3
      Else
        E.Penwidth = 1
      End If

    Next
    DiagramRedraw Diagram()
    UndoResume()
  End Function

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.5.1.2.2.4 IsEndogenousVariable Method

Returns True for endogenous variables.

Syntax

result = *object*.**IsEndogenousVariable** ()

The **IsEndogenousVariable** method syntax has the following parts:

Part	Description
result	True if *object* is an endogenous variable.
object	An object of type **PDElement**.

See example 146.

When you run this plugin, endogenous variables in the path diagram are redrawn using a pen that is 3/96 inch wide. All other objects are redrawn using a pen that is 1/96 inch wide. The UndoToHere 106 / UndoResume 105 method pair allows you to undo the changes by pressing  .

```
Imports Microsoft.VisualBasic
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
  Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    Dim E As PDElement
    UndoToHere()
    For Each E In PDElements

      If E.IsEndogenousVariable Then
        E.Penwidth = 3
      Else
        E.Penwidth = 1
      End If

    Next
    DiagramRedraw Diagram()
    UndoResume()
  End Function

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.5.1.2.2.5 IsExogenousVariable Method

Returns True for exogenous variables.

Syntax

result = *object*.**IsExogenousVariable** ()

The **IsExogenousVariable** method syntax has the following parts:

Part	Description
result	True if *object* is an exogenous variable.
object	An object of type **PDElement**.

See example 147.

When you run this plugin, exogenous variables are redrawn using a pen that is 3/96 inch wide. All other objects are redrawn using a pen that is 1/96 inch wide. The UndoToHere 106 / UndoResume 105 method pair allows you to undo the changes by pressing  .

```
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim E As PDElement
      UndoToHere()
      For Each E In PDElements

         If E.IsExogenousVariable Then
            E.Penwidth = 3
         Else
            E.Penwidth = 1
         End If

      Next
      DiagramRedraw Diagram()
      UndoResume()
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.2.2.6 IsLatentVariable Method

Returns True for latent variables.

Syntax

result = *object*.**IsLatentVariable** ()

The **IsLatentVariable** method syntax has the following parts:

Part	Description
result	True if *object* is a latent variable.
object	An object of type **PDElement**.

See Use the Amos Graphics classes to change the appearance of latent variables 492

4.5.1.2.2.7 IsObservedVariable Method

Returns True for observed variables.

Syntax

result = *object*.**IsObservedVariable** ()

The **IsObservedVariable** method syntax has the following parts:

Part	Description
result	True if *object* is an observed variable.
object	An object of type **PDElement**.

See example 149.

When you run this plugin, rectangles are redrawn using a pen that is 3/96 inch wide. All other objects are redrawn using a pen that is 1/96 inch wide. The UndoToHere 106 / UndoResume 105 method pair allows you to undo the changes by pressing  .

```
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim E As PDElement
      UndoToHere()
      For Each E In PDElements

         If E.IsObservedVariable Then
            E.Penwidth = 3
         Else
            E.Penwidth = 1
         End If

      Next
      DiagramRedraw Diagram()
      UndoResume()
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.2.2.8 IsPath Method

Returns True for single-headed arrows (regression weights).

Syntax

result = *object*.**IsPath** ()

The **IsPath** method syntax has the following parts:

Part	Description
result	True if *object* is a single-headed arrow.
object	An object of type **PDElement**.

See example 150.

When you run this plugin, single-headed arrows are redrawn using a pen that is 3/96 inch wide. All other objects are redrawn using a pen that is 1/96 inch wide. The UndoToHere 106 / UndoResume 105 method pair allows you to undo the changes by pressing  .

```
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim E As PDElement
      UndoToHere()
      For Each E In PDElements

         If E.IsPath Then
            E.Penwidth = 3
         Else
            E.Penwidth = 1
         End If

      Next
      DiagramRedraw Diagram()
      UndoResume()
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.2.2.9 IsUniqueVariable Method

Returns True for unique variables — *i.e.*, for variables that are unobserved and exogenous, and that affect only one other variable.

Syntax

result = *object*.**IsUniqueVariable** ()

The **IsUniqueVariable** method syntax has the following parts:

Part	Description

result	True if *object* is a unique variable.
object	An object of type **PDElement**.

See example 151.

When you run this plugin, unique variables are redrawn using a pen that is 3/96 inch wide. All other objects are redrawn using a pen that is 1/96 inch wide. The UndoToHere 106 / UndoResume 105 method pair allows you to undo the changes by pressing  .

```
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim E As PDElement
      UndoToHere()
      For Each E In PDElements

         If E.IsUniqueVariable Then
            E.Penwidth = 3
         Else
            E.Penwidth = 1
         End If

      Next
      DiagramRedraw Diagram()
      UndoResume()
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.2.2.10 IsUnobservedVariable Method

Returns True for ellipses (unobserved variables).

Syntax

result = *object*.**IsUnobservedVariable** ()

The **IsUnobservedVariable** method syntax has the following parts:

Part	Description
result	True if *object* is an unobserved variable.

object	An object of type **PDElement**.

See example 152.

When you run this plugin, ellipses are redrawn using a pen that is 3/96 inch wide. All other objects are redrawn using a pen that is 1/96 inch wide. The UndoToHere 106 / UndoResume 105 method pair allows you to undo the changes by pressing  .

```
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
  Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    Dim E As PDElement
    UndoToHere()
    For Each E In PDElements

      If E.IsUnobservedVariable Then
        E.Penwidth = 3
      Else
        E.Penwidth = 1
      End If

    Next
    DiagramRedraw Diagram()
    UndoResume()
  End Function

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.5.1.2.2.11 IsVariable Method

Returns True for rectangles and ellipses (*i.e.*, for variables).

Syntax

result = *object*.**IsVariable** ()

The **IsVariable** method syntax has the following parts:

Part	Description
result	True if *object* is a rectangle or ellipse.
object	An object of type **PDElement**.

See example [153].

When you run this plugin, rectangles and ellipses are redrawn using a pen that is 3/96 inch wide. All other objects are redrawn using a pen that is 1/96 inch wide. The UndoToHere [106] / UndoResume [105] method pair allows you to undo the changes by pressing  .

```
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
  Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    Dim E As PDElement
    UndoToHere()
    For Each E In PDElements

      If E.IsVariable Then
        E.Penwidth = 3
      Else
        E.Penwidth = 1
      End If
    Next

    DiagramRedraw Diagram()
    UndoResume()
  End Function

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.5.1.2.2.12 PropertyGet Method

Retrieves the value of a property that was previously saved using the PropertySave [155] method.

Syntax

result = *object*.**PropertyGet** (*propertyName*, *defaultValue*)

The **PropertyGet** method syntax has the following parts:

Part	Description
result	(String) The value of the property called *propertyName*.
object	An object of type **PDElement**.
propertyName	(String) The name of a property.

defaultValue	A string that is returned as the value of **PropertyGet** when no property with the name *propertyName* exists.

See Use the Amos Graphics classes to create user-defined properties 492

4.5.1.2.2.13 PropertyRemove Method

Removes a property associated with a path diagram or with an element of a path diagram.

Syntax

object.**PropertyRemove** (*propertyName*)

The **PropertyRemove** method syntax has the following parts:

Part	Description
object	An object of type **PDElement**.
propertyName	A string containing the name of a property.

See example 154.

When the path diagram of Example 4 (which includes a variable called **Performance**) is in the Amos Graphics window, the following plugin creates a property called **Reliability** associated with the **Performance** variable. **Reliability** is assigned the value ".8230". The PropertyGet 99 method is then used to retrieve and display the value. After the property is removed with **PropertyRemove**, an attempt to retrieve the property with PropertyGet 99 returns the value "Undefined".

```
Imports Microsoft.VisualBasic
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim E As PDElement
      E = Pd.PDE("Performance")
      E.PropertySave("Reliability", ".8230")
      'The following statement displays ".8230"
      MsgBox(E.PropertyGet("Reliability", "Undefined"))

      E.PropertyRemove("Reliability")

      'The following statement displays "Undefined"
      MsgBox(E.PropertyGet("Reliability", "Undefined"))
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.5.1.2.2.14 PropertySave Method

Saves a string that can later be retrieved by name using the PropertyGet 153 method.

Syntax

object.**PropertySave** (*propertyName*, *value*)

The **PropertySave** method syntax has the following parts:

Part	Description
object	An object of type **PDElement**.
propertyName	A string.
value	A string. A subsequent use of the PropertyGet 99 method that specifies a property name of *propertyName* will return *value*.

See Use the Amos Graphics classes to create user-defined properties 492

4.5.1.2.2.15 Undraw Method

Temporarily erases an object.

Syntax

object.**Undraw** ()

The **Undraw** method syntax has the following parts:

Part	Description
object	An object of type **PDElement**.

See Use the Amos Graphics classes to change the appearance of latent variables 492

4.5.2 AmosEngine Class Reference

An Amos Engine object is a structural equation model complete with references to data files, computational options, structural declarations and output features. An Amos Engine object also performs the calculations for structural equation modeling, but has no graphical user interface.

If you are not using Amos's built-in program editor, you need to provide a reference to **AmosEngineLib.dll** in order to use the **AmosEngine** class. In Visual Studio 2003:

- Click **Project -> Add Reference**.
- In the **Add Reference** dialog, click **Browse**.
- When the **Select Component** dialog opens, select AmosEngineLib.dll from the Amos program directory and click **Open**.
- In the **Add Reference** dialog, click **OK**.

4.5.2.1 Timing is Everything

When writing an Amos program, you have to pay attention to the order in which the Amos engine's methods are called. Amos methods are divided into three general groups.

Group 1 — Declarative Methods

These are computational and output options that apply to the entire analysis. These methods tell the Amos Engine which statistics to compute and how to compute them.

Group 2 — Data and Model Specification Methods

This group consists of data description and model specification commands for a sample of data with multigroup or multisample analyses. These commands may vary among samples.

Group 3 — Methods for Retrieving Results

These are commands to…well, retrieve results.

The rule is that groups must appear in order: Group 1, then Group 2, and finally Group 3.

There is also a special group that consists only of the Initialize method 288. If the optional Initialize method 288 is used, it must come even before the Group 1 methods. Some methods can be placed in more than one group.

4.5.2.1.1 Group 1: Declarative methods

4.5.2.1.2 Group 2: Data and model specification methods

4.5.2.1.3 Group 3: Methods for retrieving results

4.5.2.1.4 Special Case

If the following optional method is used, it must be used before any other method.

4.5.2.2 AmosEngine Class Members

This section documents the members of the **AmosEngine** class.

4.5.2.2.1 Properties

This section documents the properties of the **AmosEngine** class.

4.5.2.2.1.1 AmosDir Property

The Amos program directory.

Syntax

result = **AmosEngine.AmosDir**

The **AmosDir** property syntax has the following parts:

Part	Description
result	The Amos program directory. *result* is a character string ending with a backslash character, for example, "C:\Program Files\IBM\SPSS\Amos\32\".

Placement[156]: [1], [2] or [3]

The following program fits the model of Example 8.

```
Module MainModule
    ' AmosDir Property Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")

        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = (a) spatial + (1) err_c")
        Sem.AStructure("lozenges = (b) spatial + (1) err_l")

        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = (c) verbal + (1) err_s")
        Sem.AStructure("w ordmean = (d) verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2 Methods

This section documents the methods of the **AmosEngine** class.

4.5.2.2.2.1 Adf Method

Specifies estimation by Browne's (1982)[509] asymptotically distribution-free criterion, minimizing (D1) together with (D4) in Appendix B.

Syntax

object.**Adf** ()

The **Adf** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement[156]: [1].

Default

When you do not specify an estimation criterion, the maximum likelihood criterion (see MI Method 305) is used.

See Also

BootAdf Method 177

Gls Method 284

Ml Method 305

Sls Method 392

Uls Method 403

The following program uses the ADF estimation criterion to fit the model of Example 8.

```
Module MainModule
    ' Adf Method Example
    Public Sub Main()
        Dim Sem As AmosEngineLib.AmosEngine = New  AmosEngineLib.AmosEngine

        Sem.Adf()

        Sem.TextOutput()
        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")
        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.2 Admissible Method

Returns True if parameter estimates are admissible in all groups.

Syntax

object.**Admissible** ()

The **Admissible** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement 156: [3].

See Also

Stable Method [395]

The following example demonstrates the Admissible method

```
Module MainModule
   ' Admissible Method Example
   Public Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Sem.TextOutput()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
      Sem.GroupName("girls")
      Sem.AStructure("academic = GPA + attract + e1 (1)")
      Sem.AStructure("attract = height + w eight + rating + academic + e2 (1)")
      Sem.AStructure("e2 <--> e1")

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
      Sem.GroupName("boys")
      Sem.AStructure("academic = GPA + attract + e1 (1)")
      Sem.AStructure("attract = height + w eight + rating + academic + e2 (1)")
      Sem.AStructure("e2 <--> e1")

      If Sem.Admissible Then
         System.Diagnostics.Debug.WriteLine("Admissible")
         Else
      System.Diagnostics.Debug.WriteLine("Inadmissible")
      End If

      If Sem.Stable Then
         System.Diagnostics.Debug.WriteLine("Stable")
      Else
         System.Diagnostics.Debug.WriteLine("Unstable")
      End If
      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.3 AllImpliedMoments Method

Controls whether the implied covariance matrix for all variables is reported. When means and intercepts are explicitly modeled, **AllImpliedMoments** also controls the reporting of implied means.

Syntax

object.**AllImpliedMoments** ()
object.**AllImpliedMoments** (*tf*)

The **AllImpliedMoments** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

tf	Optional. A boolean value that controls the reporting of implied moments. True (default) requests the output. False suppresses it.

Placement[156]: [1].

Default

The implied moments for all variables are not reported.

Remarks

The 'implied' variances, covariances and means are estimates of the corresponding population values under the assumption that the specified model is correct.

If you use both the Standardized[396] and the **AllImpliedMoments** methods, the implied correlation matrix will be reported, in addition to the implied covariance matrix.

AllImpliedMoments is identical to ImpliedMoments, except that AllImpliedMoments displays implied variances, covariances and means for all variables in the model, not just for the observed variables.

See Also

ImpliedMoments Method[286]

ResidualMoments Method[377]

SampleMoments Method[389]

TextOutput Method[398]

The following program fits the model of Example 8 and displays implied covariances for 8 variables (6 measured variables and 2 latent variables).

```
Module MainModule
    ' AllImpliedMoments Method Example
    Public Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.AllImpliedMoments()

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")

        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")

        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.4 Allow Unidentified Method

Controls whether Amos tries to fit a model that appears to be unidentified.

Syntax

object.**AllowUnidentified** ()
object.**AllowUnidentified** (*tf*)

The **AllowUnidentified** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	(Boolean) Optional. If *tf* is True (the default), Amos tries to fit a model even if it appears to be unidentified.

Placement 156: [1].

Default

When a model appears to be unidentified, Amos displays an error message, and quits.

Remarks

Attempting to fit an unidentified model is generally a bad idea for the following reasons: It uses more memory, and usually takes more time. It prevents Amos from using one of its most important tests for a correct solution, namely, that the matrix of second derivatives be positive definite. For hypothesis testing purposes, it requires Amos to make a correction to degrees of freedom based on the number of

additional parameter constraints needed to achieve identifiability. The difficulty of numerically recognizing and diagnosing nonidentifiability is discussed in Appendix D.

See Also

NonPositive Method 339

The following program fits a model that is the same as Example 8, except that the model is not identified.

```
Module MainModule
    ' Allow Unidentified Method Example
    Public Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine

        Sem.AllowUnidentified()

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")

        Sem.AStructure("visperc = spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")

        Sem.AStructure("paragraph = verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.5 AnyMissingValues Method

Returns True if any group has data with missing values.

Syntax

object.**AnyMissingValues**

The **AnyMissingValues** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement 156: [3].

Remarks

The **AnyMissingValues** method should be used after one of the following methods: FitModel 237, FitAllModels 234.

See Also

InputVariableHasMissingValues Method 292

The following program displays the message "There are missing values." because the **Attg_yng** data contains a missing value for **age**.

```
Module MainModule
    ' AnyMissingValues Method Example
    Public Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.ModelMeansAndIntercepts()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Attg_yng")
        Sem.AStructure("age <--> vocabulary (0)")

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Attg_old")
        Sem.AStructure("age <--> vocabulary (0)")

        Sem.FitAllModels()

        If Sem.AnyMissingValues Then
            System.Diagnostics.Debug.WriteLine("There are missing values.")
        Else
            System.Diagnostics.Debug.WriteLine("There are no missing values.")
        End If

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.6 AStructure Method

The **AStructure** method is used to specify a model. It can be used to:

- Name the variables and parameters in the model.
- Specify linear dependencies among the variables.
- Place equality constraints on parameters.
- Specify start values for parameters.

Syntax

object.**AStructure** (*s*)

The **AStructure** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
s	A string that refers to one of the following: • a regression weight • a regression equation

	• the covariance between two exogenous variables • the variance of a single exogenous variable *s* is discussed more fully under Extended explanation of the AStructure method 170.

Placement 156: [2].

Remarks

Use the MStructure 315 method to constrain the means of exogenous variables.

The **AStructure** and MStructure 315 methods are provided for compatibility with the obsolete **$structure** and **$mstructure** commands. Consider using the newer Path 364, Cov 212, Var 405, Mean 303 and Intercept 299 methods for model specification.

See Also

MStructure Method 315

The following program fits the model of Example 8.

```
Module MainModule
   ' AStructure Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Sem.TextOutput()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")

      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.AStructure("w ordmean = verbal + (1) err_w ")

      Sem.Dispose()
   End Sub
End Module
```

There are two ways to use the **AStructure** method to specify linear relationships among variables. The first way uses the symbol '**<**' or the symbol '**>**' to represent a linear dependency. For example, the following lines specify that **variable1** depends directly on **variable2**.

```
   Dim Sem as New  AmosEngine
   . . .
   Sem.AStructure("variable1 < variable2")
```

The following lines have the same effect.

```
   Dim Sem as New  AmosEngine
   . . .
   Sem.AStructure("variable2 > variable1")
```

That is, in the path diagram there is an arrow pointing from **variable2** to **variable1**. Amos estimates the corresponding regression weight.

The **AStructure** method ignores dashes, so that the following four lines are equivalent.

```
Sem.AStructure("variable1<variable2")
Sem.AStructure("variable1 <--- variable2")
Sem.AStructure("variable1 <---------- variable2")
Sem.AStructure("variable2 ----------> variable1")
```

By default, Amos assumes that the regression weights are unconstrained. However, you can set the value of any regression weight to a constant, and you can require any number of regression weights to be equal to each other. The following example shows how to impose such constraints:

```
Dim Sem as New AmosEngine
. . .
Sem.AStructure("variable1 <--- variable2 (red)")
Sem.AStructure("variable1 <--- variable3 (red)")
Sem.AStructure("variable1 <--- variable4 (1)")
Sem.AStructure("variable1 <--- variable5")
Sem.AStructure("variable1 <--- variable6 (blue)")
Sem.AStructure("variable1 <--- variable7 (blue)")
Sem.AStructure("variable1 <--- variable8 (blue)")
. . .
```

In this example, the first two regression weights are required to be equal because they are both labeled red. Similarly, the last three regression weights are required to be equal because they are both labeled blue. The regression weight for the regression of **variable1** on **variable4** is fixed at 1. (Amos will not attempt to estimate this regression weight.) The regression weight for predicting **variable1** from **variable5** is not constrained.

A linear relationship can also be described by an equation, as in the following lines.

```
Dim Sem as New AmosEngine
. . .
Sem.AStructure("variable1 = () variable2 + () variable3 + () variable4")
```

Here, **variable1** is specified to be a linear function of **variable2**, **variable3** and **variable4**. The empty parentheses represent unknown regression weights that Amos will estimate. Actually, the empty parentheses can be left out, as in the following lines, which are equivalent to the ones just previous.

```
Dim Sem as New AmosEngine
. . .
Sem.AStructure("variable1 = variable2 + variable3 + variable4")
```

Amos takes it for granted that it is supposed to estimate a regression weight for each variable on the right hand side of this equation. The parentheses sometimes contribute to readability, and, as will be shown below, provide the means for placing constraints on regression weights.

In using this method for specifying linear relationships, it is possible to specify one or more intercepts in a regression equation. For example:

```
Dim Sem as New AmosEngine
. . .
Sem.AStructure("var1 = () var2 + ()")
Sem.AStructure("var3 = () var4 + () + ()")
. . .
```

Here, **var1** is specified to be a multiple of **var2**, except for an intercept term. **var3** is specified to be a multiple of **var4**, except for two intercept terms. The five sets of empty parentheses represent five

parameters that Amos is supposed to estimate - two regression weights and three intercepts. Again, the empty parentheses can be left out if desired.

Constraints can be placed on regression weights and intercepts as in the following example:

```
Dim Sem as New AmosEngine
. . .
Sem.AStructure("var1 = (alpha) var2 + (charlie)")
Sem.AStructure("var3 = (alpha) var4 + (charlie) + (50)")
. . .
```

Here, Amos is required to estimate two parameters. The two regression weights labeled alpha are required to be equal. Their common value constitutes one parameter. The two intercepts labeled charlie are required to be equal. Their common value constitutes the second parameter. The remaining intercept is fixed at 50, so it doesn't have to be estimated.

With one exception, Amos assumes that the *exogenous variables in a model are correlated*, and it estimates the covariance between every pair of exogenous variables. The exception to this default assumption concerns unique variables — exogenous variables that are unobserved and have a direct effect on only one variable. Amos assumes that *unique variables are uncorrelated* with each other, and with every other exogenous variable in the model.

You may explicitly permit two variables to be correlated by using the string, **<>**, as illustrated in the following example:

```
Dim Sem as New AmosEngine
. . .
Sem.AStructure("variable2 <> variable3")
```

where Amos is told that **variable2** and **variable3** may be correlated, and is asked to estimate their covariance.

Since the **AStructure** method ignores dashes, the following three lines are equivalent:

```
Sem.AStructure("variable2<>variable3")
Sem.AStructure("variable2 <---> variable3")
Sem.AStructure("variable2 <----------> variable3")
```

You can place constraints on the covariances of exogenous variables, as in the following example:

```
Dim Sem as New AmosEngine
. . .
Sem.AStructure("variable1 <---> variable2 (alpha)")
Sem.AStructure("variable2 <---> variable3 (alpha)")
Sem.AStructure("variable3 <---> variable4 (alpha)")
Sem.AStructure("variable1 <---> variable3 (beta)")
Sem.AStructure("variable2 <---> variable4 (beta)")
Sem.AStructure("variable1 <---> variable2 (0)")
. . .
```

In this example, the first three covariances listed are required to be equal because they are all labeled alpha. Similarly, the two covariances labeled beta are required to be equal to each other. **variable1** and **variable2** are declared to be uncorrelated, so that Amos will not attempt to estimate their covariance. (It is also possible to fix a covariance to a nonzero value, although reasons for doing so are rare.)

By default, Amos assumes that there are no constraints on the variances of the exogenous variables in the model. However, you can constrain the variances, as in the following example:

```
Dim Sem as New AmosEngine
. . .
Sem.AStructure("variable1 (gamma)")
Sem.AStructure("variable2 (gamma)")
Sem.AStructure("variable3 (gamma)")
Sem.AStructure("variable4 (1)")
Sem.AStructure("variable5 (2)")
Sem.AStructure("variable6 (delta)")
Sem.AStructure("variable7 (delta)")
. . .
```

In this example, the variance of **variable4** is fixed at 1, and the variance of **variable5** is fixed at 2. Amos does not attempt to estimate these fixed parameters. **variable1**, **variable2** and **variable3** are required to have the same variance because they are all labeled gamma. Similarly, **variable6** and **variable7** are required to have the same variance because they are both labeled delta.

To provide an initial value for a parameter, type the initial value followed by question mark. In the following example the variances of **variable4** and **variable5** are given initial values of 15 and 16:

```
Dim Sem as New AmosEngine
. . .
Sem.AStructure("variable1 (gamma)")
Sem.AStructure("variable2 (gamma)")
Sem.AStructure("variable3 (gamma)")
Sem.AStructure("variable4 (15?)")
Sem.AStructure("variable5 (16?)")
Sem.AStructure("variable6 (delta)")
Sem.AStructure("variable7 (delta)")
. . .
```

To give a parameter a non-numeric label as well as an initial value, type the non-numeric label, followed by a colon, followed by the initial value. In the following example the variances of **variable1**, **variable2** and **variable3** are constrained to be equal and given an initial value of 8, while the variances of **variable6** and **variable7** are constrained to be equal and given an initial value of 9:

```
Dim Sem as New AmosEngine
. . .
Sem.AStructure("variable1 (gamma : 8)")
Sem.AStructure("variable2 (gamma : 8)")
Sem.AStructure("variable3 (gamma : 8)")
Sem.AStructure("variable4 (1)")
Sem.AStructure("variable5 (2)")
Sem.AStructure("variable6 (delta : 9)")
Sem.AStructure("variable7 (delta : 9)")
. . .
```

4.5.2.2.2.7 BeginGroup Method

Specifies the data file, and begins the model specification for a single group. **BeginGroup** is a simplified form of the BeginGroupEx 175 method. The database format is inferred from the data file name.

Syntax

object.**BeginGroup** (*fileName*)

object.**BeginGroup** (*fileName*, *tableName*)

object.**BeginGroup** (*fileName*, *groupingVariable*, *groupingValue*)

object.**BeginGroup** (*fileName, tableName, groupingVariable, groupingValue*)

The **BeginGroup** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
fileName	Name of a data file.
tableName	Name of a data table within the data file. Required for database formats, such as Excel 5, Excel 97 and Access, in which a file can contain multiple data tables. Must be omitted for database formats, such as Excel 4, dbase and SPSS Statistics, for which a file can contain only a single data table.
groupingVariable	groupingVariable and groupingValue are optional. Together they select a subset of cases from the data file for analysis. The analysis includes those cases for which the variable named groupingVariable takes on the value groupingValue.
groupingValue	See GroupingVariable above.

Placement 156: [2].

See Also

GroupName Method 285

The following program fits the model of Example 11-a.

```
Module MainModule
    ' BeginGroup Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
        Sem.GroupName("girls")
        Sem.AStructure("academic = GPA + attract + e1 (1)")
        Sem.AStructure("attract = height + w eight + rating + academic + e2 (1)")
        Sem.AStructure("e2 <--> e1")

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
        Sem.GroupName("boys")
        Sem.AStructure("academic = GPA + attract + e1 (1)")
        Sem.AStructure("attract = height + w eight + rating + academic + e2 (1)")
        Sem.AStructure("e2 <--> e1")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.8 BeginGroupEx Method

Specifies the data file, and begins the model specification for a single group.

Syntax

object.**BeginGroupEx** (*dbFormat*, *fileName*, *tableName*)
object.**BeginGroupEx** (*dbFormat*, *fileName*, *tableName, groupingVariable, groupingValue*)

The **BeginGroupEx** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
dbFormat	A database format specifier.
fileName	Name of a data file.
tableName	Name of a data table within the data file. Required for database formats, such as Excel 5, Excel 97 and Access, in which a file can contain multiple data tables. Must be the empty string ("") for database formats, such as Excel 4, dbase and SPSS Statistics, for which a file can contain only a single data table.
groupingVariable	*GroupingVariable* and GroupingValue are optional. Together they select a subset of cases from the data file for analysis. The analysis includes those cases for which the variable named VariableName takes on the value GroupingValue.
groupingValue	See GroupingVariable above.

Placement 156: [2].

Settings

The settings for *DbFormat* are:

Constant	Value	Description
mmDBASE3	0	Dbase III
mmDBASE4	1	Dbase IV
mmDBASE5	2	Dbase V

mmEXCEL3	3	Excel 3
mmEXCEL4	4	Excel 4
mmEXCEL5	5	Excel 5, Excel 7
mmEXCEL97	6	Excel 97, Excel 8
mmFOXPRO20	7	Foxpro 2.0
mmFOXPRO25	8	Foxpro 2.5
mmFOXPRO26	9	Foxpro 2.6
mmLOTUSWK1	11	Lotus *.wk1
mmLOTUSWK3	12	Lotus *.wk3
mmLOTUSWK4	13	Lotus *.wk4
mmAccess	14	Microsoft Access
mmSPSS	18	SPSS Statistics
mmText	19	Text

See Also

BeginGroup Method 173

GroupName Method 285

The following program fits the model of Example 11-a.

```
Imports PXMLPersist.CDataTable.cDatabaseFormat
Module MainModule
   ' BeginGroupEx Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Sem.TextOutput()

      Sem.BeginGroupEx(mmEXCEL97, AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
      Sem.GroupName("girls")
      Sem.AStructure("academic = GPA + attract + e1 (1)")
      Sem.AStructure("attract = height + w eight + rating + academic + e2 (1)")
      Sem.AStructure("e2 <--> e1")

      Sem.BeginGroupEx(mmEXCEL97, AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
      Sem.GroupName("boys")
      Sem.AStructure("academic = GPA + attract + e1 (1)")
      Sem.AStructure("attract = height + w eight + rating + academic + e2 (1)")
      Sem.AStructure("e2 <--> e1")

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.9 BootAdf Method

Controls display of the histogram of discrepancies,

$$C_{ADF}(\hat{\alpha}_1,\mathbf{a}),C_{ADF}(\hat{\alpha}_2,\mathbf{a}),\ldots,C_{ADF}(\hat{\alpha}_B,\mathbf{a})$$

In the above formula, $\mathbf{a}$ is the vector of sample moments, *B* is the number of bootstrap samples and $\hat{\alpha}_b$ is the vector of implied moments obtained by fitting the model to the *b*-th bootstrap sample. The mean and standard deviation of the distribution are also reported.

Syntax

object.**BootAdf** ()
object.**BootAdf** (*tf*)

The **BootAdf** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. If *tf* is True (the default), the histogram is displayed. Otherwise, not

Placement[156]: [1].

Default

The distribution of

$$C_{ADF}(\hat{\alpha}_1,\mathbf{a}), C_{ADF}(\hat{\alpha}_2,\mathbf{a}), \ldots, C_{ADF}(\hat{\alpha}_B,\mathbf{a})$$

is reported only when a bootstrap is performed using the **Adf** method.

Remarks

When a bootstrap is not performed, **BootAdf** is ignored. See Example 21 in the *User's Guide* for a demonstration of the **BootAdf** method.

See Also

Adf Method 163

BootGls Method 183

BootMl Method 185

BootSls Method 186

Bootstrap Method 188

BootUls Method 192

BootVerify Method 193

Seed Method 390

The following example demonstrates the BootAdf method.

```
Module MainModule
    ' BootAdf Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.Bootstrap(200)

        Sem.BootAdf()

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")

        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.10 BootBS Method

Carries out the bootstrap procedure of Bollen and Stine (1992) [508] for testing the hypothesis that the specified model is correct.

Syntax

object.**BootBS** ()
object.**BootBS** (*tf*)

The **BootBS** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. True (default) specifies that a Bollen-Stine test is to be carried out.

Placement [156]: [1].

Default

The Bollen-Stine test is not carried out.

Remarks

If you use **BootBS**, you must also use Bootstrap [188] to specify the number of bootstrap samples. However, when **BootBS** is used, bootstrapped standard errors are not reported. To obtain bootstrapped standard errors, run the problem without the **BootBS** Method. The **BootBS** Method is only for testing *model fit* under non-normality.

See Also

Bootstrap Method [188]

BootVerify Method [193]

Seed Method [390]

The following example demonstrates the BootBS method.

```
Module MainModule
    ' BootBS Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.BootBS()
        Sem.Bootstrap(2000)

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

In this example, **BootBS** requests a Bollen-Stine bootstrap analysis, and **Bootstrap 2000** requests 2000 bootstrap samples.

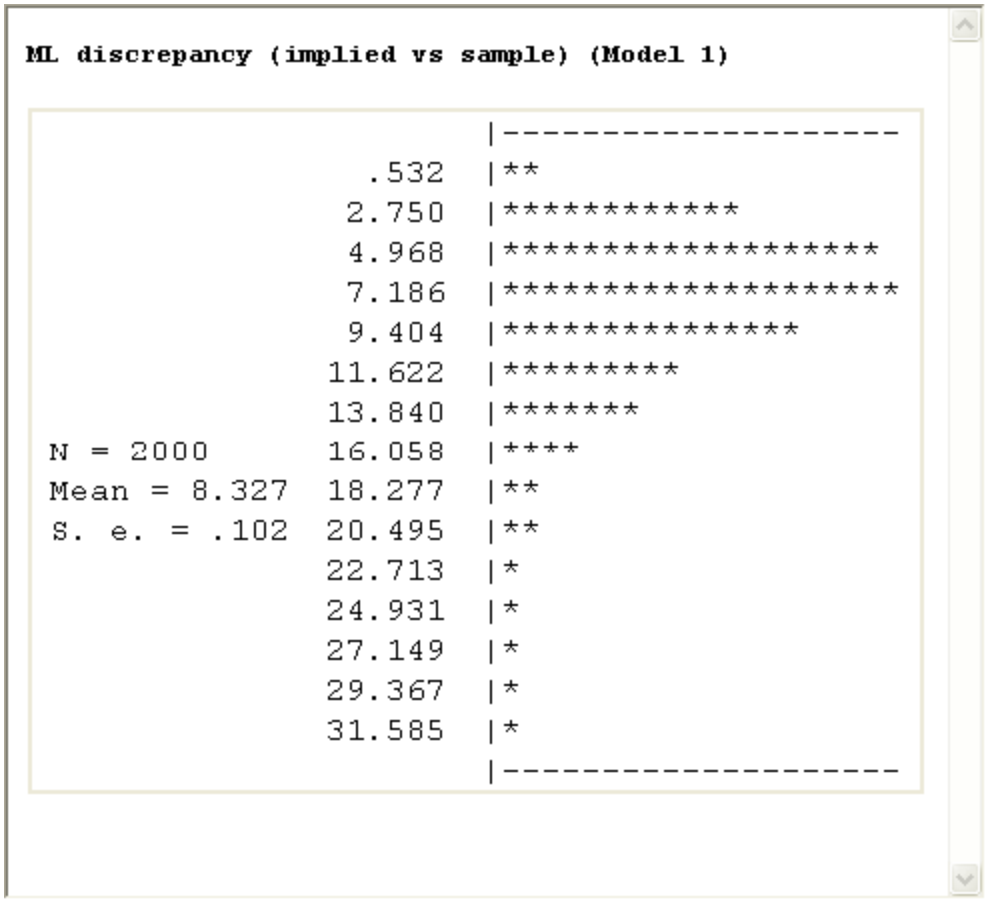
The Bollen-Stine procedure provides a test of the hypothesis that the model is correct. This is the same null hypothesis that is tested by the conventional chi-square test of fit in maximum likelihood, generalized least squares and asymptotically distribution-free estimation. The objective of the procedure is to ascertain the probability that the discrepancy function would be as large as it actually turned out to be in the current sample, under the hypothesis that your model is correct.

In the Bollen-Stine approach, a transformation of the sample data is carried out so as to make your model fit the transformed data exactly. Bootstrap samples are drawn from the transformed sample data. The distribution of the discrepancy function across bootstrap samples is then taken as an estimate of its distribution under the hypothesis that the model is correct.

Using maximum likelihood estimation (Amos's default), the likelihood ratio chi-square statistic is 7.853 with 8 degrees of freedom (p = .448). The following output indicates that 46.2% of the 2000 bootstrap samples had a likelihood ratio chi-square statistic greater than 7.853.

```
Testing the null hypothesis that the specified model is correct:
        Bollen-Stine bootstrapped p = 0.462
```

Thus, the departure of the data from the model is significant at the .462 level. In other words, the data do not depart significantly from the model at any conventional significance level. The distribution of 2000 likelihood ratio chi-square statistics obtained from the 2000 bootstrap samples is as follows.



This distribution resembles the chi-square distribution with eight degrees of freedom insofar as it is positively skewed and has a mean of about eight (actually 8.327). Unfortunately, Amos does not provide the information needed to do a more detailed comparison with the chi-square distribution.

4.5.2.2.2.11 BootFactor Method

Speeds up the bootstrap 188 algorithm and makes it more reliable under the assumption that standard errors are inversely proportional to the square root of sample size.

Syntax

object.**BootFactor** (*M*)

The **BootFactor** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
M	A positive integer. In the BootFactor example 183, *M*=5. Suppose that your data consist of *G* independent samples (groups) with sample sizes

	$N_1, N_2, \ldots, N_G$. Each bootstrap sample will be obtained from the original sample by drawing MN_1 observations at random (with replacement) from the first original sample, MN_2 observations from the second original sample, and so on. As an example, suppose you have two independent groups with 200 cases in the first group and 250 cases in the second group, and that *M* = 5. Then each bootstrap sample will consist of 1000 cases drawn from the first group and 1250 cases drawn from the second group. The bootstrapped standard error displayed by Amos for any parameter estimate will be the standard deviation of that estimate across bootstrap replications, multiplied by $\sqrt{M}$.

Placement[156]: [1].

Default

M = 1.

Remarks

Using a value for *M* other than 1 requires the assumption that the standard error of each estimate is inversely proportional to the square root of sample size.

The use of *M* > 1 can substantially reduce computation time, and reduces the probability of encountering a bootstrap sample for which parameter estimation is impossible. The larger *M* is, the larger the bootstrap samples will be, the more closely their sample moments will resemble the moments of the original sample, and the more closely the parameter estimates for the bootstrap samples will resemble the parameter estimates from the original sample. Since the parameter estimates from the original sample are used as initial values in the analysis of each bootstrap sample, a large value for *M* reduces the amount of computation required to estimate parameters for a bootstrap sample. Of course, if *M* is set to a very large value, generating the bootstrap samples will become the dominant cost factor. A very large *M* may also create numerical problems.

The use of *M* > *1* solves a problem described in a special case by Dolker, Halperin and Divgi (1982)[512]. With small samples and *M=1*, the sample covariance matrix in a bootstrap sample may be singular even though the covariance matrix in the original sample is nonsingular. The occurrence of a singular covariance matrix in a bootstrap sample prohibits estimation by the Gls[284] or Adf[163] methods. The larger *M* is, the smaller are the chances of finding a singular sample covariance matrix in a bootstrap sample.

It is not possible to perform a Bollen-Stine bootstrap test of fit or to obtain bootstrap confidence intervals (ConfidenceBC[207] or ConfidencePC[209]) if *M* >1.

See Also

Bootstrap Method[188]

BootVerify Method 193

Seed Method 390

This example demonstrates the BootFactor method.

```
Module MainModule
    ' BootFactor Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.Bootstrap(200)
        Sem.BootFactor(5)

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.12 BootGls Method

Controls display of the histogram of discrepancies,

$$C_{GLS}(\hat{\alpha}_1,\mathbf{a}), C_{GLS}(\hat{\alpha}_2,\mathbf{a}),\ldots, C_{GLS}(\hat{\alpha}_B,\mathbf{a})$$

In the above formula, $\mathbf{a}$ is the vector of sample moments, *B* is the number of bootstrap samples and $\hat{\alpha}_b$ is the vector of implied moments obtained by fitting the model to the *b*-th bootstrap sample. The mean and standard deviation of the distribution are also reported.

Syntax

object.**BootGls** ()
object.**BootGls** (*tf*)

The **BootGls** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. If *tf* is True (the default), the histogram is displayed. Otherwise, not.

Placement[156]: [1].

Default

The distribution of

$C_{GLS}(\hat{\alpha}_1,\mathbf{a}), C_{GLS}(\hat{\alpha}_2,\mathbf{a}),\ldots, C_{GLS}(\hat{\alpha}_B,\mathbf{a})$ is reported only when a bootstrap is performed using the **Gls** method.

Remarks

When a bootstrap is not performed, **BootGls** is ignored. For a demonstration of the **BootGls** method, see the files **Ex21-adf.vb**, **Ex21-gls.vb**, **Ex21-ml.vb** and **Ex21- uls.vb** in the Examples subdirectory.

See Also

BootAdf Method[177]

BootMl Method[185]

Bootstrap Method[188]

BootSls Method[186]

BootUls Method[192]

BootVerify Method[193]

Gls Method[284]

Seed Method[390]

This example demonstrates the BootGls method.

```
Module MainModule
   ' BootGls Method Example
   Sub Main()
      Dim Sem As New AmosEngineLib.AmosEngine

      Sem.Bootstrap(200)
      Sem.BootGls()

      Sem.TextOutput()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.AStructure("w ordmean = verbal + (1) err_w ")

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.13 BootMl Method

Controls display of the histogram of discrepancies,

$$C_{KL}(\hat{\alpha}_b, \mathbf{a}_b) - C_{KL}(\mathbf{a}, \mathbf{a}), \quad b = 1, \ldots, B$$

.

In the above formula, $\mathbf{a}$ is the vector of sample moments, *B* is the number of bootstrap samples and $\hat{\alpha}_b$ is the vector of implied moments obtained by fitting the model to the *b*-th bootstrap sample. The mean and standard deviation of the distribution are also reported.

Syntax

object.**BootMl** ()
object.**BootMl** (*tf*)

The **BootMl** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. If *tf* is True (the default), the histogram is displayed. Otherwise, not.

Placement[156]: [1].

Default

The distribution of

$$C_{KL}(\hat{\alpha}_b, \mathbf{a}_b) - C_{KL}(\mathbf{a}, \mathbf{a}), \quad b = 1, \ldots, B$$

, is reported only when a bootstrap is performed using the **Ml** method.

Remarks

When no bootstrap is performed, **BootMl** is ignored. For a demonstration of the **BootMl** method, see the files **Ex21-adf.vb**, **Ex21-gls.vb**, **Ex21-ml.vb** and **Ex21- uls.vb** in the Examples subdirectory.

See Also

BootAdf Method[177]

BootGls Method[183]

BootSls Method[186]

Bootstrap Method[188]

BootUls Method[192]

BootVerify Method 193

MI Method 305

Seed Method 390

This example demonstrates the BootMI method.

```
Module MainModule
    ' BootMI Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.Bootstrap(200)
        Sem.Gls()

        Sem.BootMI()

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.14 BootSls Method

Controls display of the histogram of discrepancies,

$$C_{SLS}(\hat{\alpha}_1, \mathbf{a}), C_{SLS}(\hat{\alpha}_2, \mathbf{a}), \ldots, C_{SLS}(\hat{\alpha}_B, \mathbf{a})$$

In the above formula, $\mathbf{a}$ is the vector of sample moments, *B* is the number of bootstrap samples and $\hat{\alpha}_b$ is the vector of implied moments obtained by fitting the model to the *b*-th bootstrap sample. The mean and standard deviation of the distribution are also reported.

Syntax

object.**BootSls** ()
object.**BootSls** (*tf*)

The **BootSls** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

tf	Optional. If *tf* is True (the default), the histogram is displayed. Otherwise, not.

Placement[156]: [1].

Default

The distribution of

$C_{SLS}(\hat{\alpha}_1, \mathbf{a}), C_{SLS}(\hat{\alpha}_2, \mathbf{a}), \ldots, C_{SLS}(\hat{\alpha}_B, \mathbf{a})$ is reported only when a bootstrap is performed using the Sls[392] method.

Remarks

When a bootstrap is not performed, **BootSls** is ignored. For a demonstration of the **BootSls** method, see the files **Ex21-adf.vb, Ex21-gls.vb**, **Ex21-ml.vb** and **Ex21- uls.vb** in the Examples subdirectory.

See Also

BootAdf Method[177]

BootGls Method[183]

BootMl Method[185]

Bootstrap Method[188]

BootUls Method[192]

BootVerify Method[193]

Seed Method[390]

BootFactor Method Example[183]

This example demonstrates the BootSIs method.

```
Module MainModule
    ' BootSIs Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.Bootstrap(200)
        Sem.BootSIs()

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.15 Bootstrap Method

Requests bootstrap *standard errors* for parameter estimates using the bootstrap algorithm of Efron (1982) [513] and specifies the number of bootstrap samples.

Syntax

object.**Bootstrap** (*nSamples*)

The **Bootstrap** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
nSamples	The number of bootstrap samples. If the value of *nSamples* is zero, no bootstrap is performed.

Placement [156]: [1].

Default

No bootstrap is performed.

Remarks

Amos displays bootstrap standard errors for the estimates displayed by the methods: Standardized [396], Smc [393], FactorScoreWeights [231], TotalEffects [402], SampleMoments [389], ImpliedMoments [286] and AllImpliedMoments [165].

Bootstrap standard errors are reported only for those quantities that are estimated. For example, to obtain bootstrapped standard errors for squared multiple correlations, you need to use the Smc 393 method. Similarly, to obtain bootstrapped standard errors for sample correlations, you must use both SampleMoments 389 and Standardized 396. Note that standard errors for estimated indirect effects are provided along with the bootstrapped standard errors of the TotalEffects 402 method.

Bootstrap requires raw data unless you use MonteCarlo 314.

See Also

BootAdf Method 177

BootBS Method 179

BootFactor Method 181

BootGls Method 183

BootMl Method 185

BootUls Method 192

BootVerify Method 193

ConfidenceBC Method 207

ConfidencePC Method 209

GetBCLowerBounds, GetBCUpperBounds Methods 243

GetBootSampleEstimates Method 251

GetPCLowerBounds, GetPCUpperBounds Methods 269

GetStandardErrors Method 278

MonteCarlo Method 314

Seed Method 390

This example demonstrates the Bootstrap method.

```
Module MainModule
    ' Bootstrap Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.Bootstrap(2000)

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

The accuracy of bootstrap estimates of standard error increases with sample size and with the number of bootstrap replications. Unfortunately, there are no guidelines regarding adequate sample size for the broad range of models allowed by Amos. As for the required number of bootstrap replications, Efron (1982) [513] gives some suggestions.

Subject to the limitations in the topic Accuracy of the bootstrap [190], bootstrapping offers the following advantages within Amos: Bootstrapping does not require distributional assumptions (although it does require independent observations). Bootstrapped standard errors are available for most of the statistics produced by Amos (not just for model parameters). Bootstrapping works for any estimation criterion, including Uls [403] and Sls [392]. Bootstrapping works even if the specified model is wrong.

Amos uses the parameter estimates from the original sample as initial estimates in the iterative estimation procedure for each bootstrap sample. An alternative procedure, not implemented in Amos, would be to repeat for each bootstrap sample the same procedure for choosing initial values that was used in the analysis of the original sample. In principle, this approach would provide the most faithful replication of the analysis of the original sample.

The correctness of Amos's strategy for choosing initial values depends on whether the initial values affect the final values, and there are two issues here. One issue concerns the possible existence of multiple local minima of the discrepancy function. If there are multiple local minima, the choice of initial values will determine which local minimum appears as the final solution. For this reason, it may be that using the same initial values for every bootstrap replication would tend to produce unusually small estimated standard errors. Amos's choice of initial values in bootstrap replications is thus problematical in the presence of multiple local minima. On the other hand, it is not clear that computing fresh initial estimates for each bootstrap replication would be worth the trouble. If multiple local minima are suspected, the

dependability of the entire estimation procedure is open to question, so that it would be cold comfort in any case to have estimates of standard errors even if they could be had.

A second issue in the choice of initial values for bootstrap replications concerns the numerical accuracy of Amos estimates. Neglecting the possibility of multiple local minima, it remains true that the choice of initial values will have at least a marginal effect on the final parameter estimates in each bootstrap replication. This is partly due to round-off error and partly due to the fact that Amos uses an iterative procedure that terminates at a more or less arbitrary point (see the documentation of the Crit1 217 and Crit2 218 methods).

There is thus the possibility that using the same initial estimates for each bootstrap replication will systematically influence the parameter estimates in each replication in such a way as to affect the bootstrapped standard errors. Numerical experiments have shown, however, that variability in parameter estimates resulting from the manipulation of initial values is negligible compared to variability from one bootstrap sample to another. Of course, for a statistic with a very small standard error, numerical inaccuracies may be the primary source of variability from one bootstrap sample to another. The behavior of Amos in such extreme cases has not been investigated.

In fitting a structural equation model, you have to impose constraints on the model so as to fix the unit of measurement of each unobserved variable. If you are planning to use the Bootstrap 188 method, you should fix the scales of the unobserved variables by placing appropriate constraints on the *regression weights*, and not by constraining the variances of the unobserved variables. This method for fixing units of measurement is necessary for the following reason: If the scales of measurement of the unobserved variables are fixed by constraining their variances, the criterion of minimizing the discrepancy function will determine some of the regression weights only up to a sign change. That is, given one set of parameter estimates, it will be possible to change the signs of some of the regression weights without affecting the fit of the model. This is actually an example of nonidentifiability and also an example of multiple local minima, but it is a benign example unless you are bootstrapping. In bootstrapping, if the signs of some regression weights are arbitrary, their estimates will tend to 'jump around' from one bootstrap replication to another, and the reported bootstrap standard errors will be artificially inflated as a result.

Amos discards a bootstrap sample if it cannot estimate parameters for that sample. (The chances of this happening increase with the number of bootstrap replications and decrease with sample size.) The number of discarded bootstrap samples is reported. Inadmissible solutions and unstable systems encountered during bootstrap replications are not reported.

The amount of computation required for bootstrapping is highly variable. In general, computational cost increases with the number of bootstrap replications, the number of variables and the number of parameters, and decreases (up to a point) with sample size. For a large problem, you can request two or three bootstrap replications (**`Bootstrap 2`** or **`Bootstrap 3`**) to get a time estimate.

4.5.2.2.2.16 BootUls Method

Controls the display of the histogram of discrepancies,

$$C_{ULS}(\hat{\alpha}_1, \mathbf{a}), C_{ULS}(\hat{\alpha}_2, \mathbf{a}), \ldots, C_{ULS}(\hat{\alpha}_B, \mathbf{a})$$

In the above formula, $\mathbf{a}$ is the vector of sample moments, *B* is the number of bootstrap samples and $\hat{\alpha}_b$ is the vector of implied moments obtained by fitting the model to the *b*-th bootstrap sample. The mean and standard deviation of the distribution are also reported.

Syntax

object.**BootUls** ()
object.**BootUls** (*tf*)

The **BootUls** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. If *tf* is True (the default), the histogram is displayed. Otherwise, not.

Placement[156]: [1].

Default

The distribution of

$C_{ULS}(\hat{\alpha}_1, \mathbf{a}), C_{ULS}(\hat{\alpha}_2, \mathbf{a}), \ldots, C_{ULS}(\hat{\alpha}_B, \mathbf{a})$ is reported only when a bootstrap is performed using the Uls[403] method.

Remarks

When a bootstrap is not performed, **BootUls** is ignored. For a demonstration of the **BootUls** method, see the files **Ex21-adf.vb**, **Ex21-gls.vb**, **Ex21-ml.vb** and **Ex21-uls.vb** in the Examples subdirectory.

See Also

BootAdf Method[177]

BootGls Method[183]

BootMl Method[185]

BootSls Method[186]

Bootstrap Method[188]

BootVerify Method 193

UIs Method 403

Seed Method 390

This example demonstrates the BootUIs method.

```
Module MainModule
   ' BootUIs Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine

      Sem.Bootstrap(200)
      Sem.BootUIs()

      Sem.TextOutput()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.astructure("w ordmean = verbal + (1) err_w ")

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.17 BootVerify Method

Controls the reporting of detailed information about individual bootstrap samples.

Syntax

object.**BootVerify** ()
object.**BootVerify** (*tf*)

The **BootVerify** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. True (default) requests the reporting of detailed information. False suppresses it.

Placement 156: [1].

Default

Detailed information is not reported.

Remarks

Detailed information consists of the frequency with which each observation from the original sample appears in each bootstrap sample.

See Also

Bootstrap Method 188

GetBootSampleEstimates Method 251

NeedBootSampleEstimates Method 325

Seed Method 390

This example demonstrates the BootVerify method.

```
Module MainModule
   ' BootVerify Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine

      Sem.Bootstrap(2000)
      Sem.BootVerify()

      Sem.TextOutput()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.AStructure("w ordmean = verbal + (1) err_w ")

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.18 ChiCorrect Method

Specifies the value of the constant, *r*, in Appendix A of the *User's Guide*.

Syntax

object.**ChiCorrect** (*r*)

The **ChiCorrect** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
r	The constant r in Appendix A of the User's Guide.

Placement[156]: [1].

Default

r is the number of groups.

Remarks

When the EmuLisrel6[224] method is used, the default value for *r* cannot be changed.

The choice of *r* affects only the discrepancy function and standard errors. It does not affect parameter estimates.

There are few guidelines for departing from Amos's default value for *r*. Browne (1982, p. 98)[509], mentions favorably a formula due to Swain (1975)[526]:

$$r = \frac{p(2p^2+3p-1)-y(2y^2+3y-1)}{12d}$$

where

$$y = \tfrac{1}{2}[(1+8q)^{1/2}-1],$$

using the notation in Appendix A of the *User's Guide*.

The Swain formula is intended for the case of a single group with unconstrained means and intercepts, where the model is invariant under a constant scaling factor. A one-group model in which means and intercepts are unconstrained was called "invariant under a constant scaling factor" by Browne (1982, p. 77)[509] if, given any parameter vector, , and a positive number, *c*, there exists * such that

$$\Sigma^{(1)}(\boldsymbol{\gamma}^*) = d\,\Sigma^{(1)}(\boldsymbol{\gamma}).$$

When means and intercepts are highly constrained, some consideration should be given to using the **ChiCorrect** method to specify *r* = *0*.

See Also

Cmin Method[197]

This example demonstrates the ChiCorrect method.

```
Module MainModule
    ' ChiCorrect Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.ChiCorrect(0)

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.19 ChiSquareProbability Method

Gets the "p value" for a chi square statistic.

Syntax

result = *object*.**ChiSquareProbability** (*chiSquare*, *df*)

The **ChiSquareProbability** method syntax has the following parts:

Part	Description
result	The probability that a chi square statistic with *df* degrees of freedom will exceed *chiSquare*.
object	An object of type **AmosEngine**.
chiSquare	a chi square value
df	degrees of freedom

Placement[156]: [1], [2] or [3].

See Also

P Method[353]

The following program displays the probability that a chi square variable with one degree of freedom will exceed 3.841.

```
Imports System.Diagnostics
Imports AmosEngineLib
Module MainModule
    ' ChiSquareProbability Method Example
    Sub Main()
        Debug.WriteLine(AmosEngine.ChiSquareProbability(3.841, 1))
    End Sub
End Module
```

4.5.2.2.2.20 Cmin Method

Gets the minimized value of the discrepancy function.

Syntax

result = *object*.**Cmin** ()

The **Cmin** method syntax has the following parts:

Part	Description
result	The minimized value of the discrepancy function.
object	An object of type **AmosEngine**.

Placement[156]: [3].

Remarks

If you have used the Model[306] method to define more than one model, the **Cmin** method gets the minimized discrepancy function for the most recently fitted model. The second example shows how to obtain the minimized discrepancy for multiple models.

The following program shows how to display various fit measures when only one model is defined (i.e., when the Model[306] method has been used only once or not at all).

```
Imports System.Diagnostics
Module MainModule
    ' Cmin Method Example 1
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Debug.WriteLine("Chi Square = " & Sem.Cmin)
        Debug.WriteLine("Degrees of Freedom = " & Sem.df)
        Debug.WriteLine("p = " & Sem.p)
        Debug.WriteLine("Number of parameters = " & Sem.npar)
        Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & Sem.NcpLo & _
        ", " & Sem.NcpHi & ")")
        Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
        Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)

        Sem.Dispose()
    End Sub
End Module
```

The following program shows how to display various fit measures for multiple models (i.e., when the Model (306) method has been used more than once).

```
Imports System.Diagnostics
Module MainModule
   ' Cmin Method Example 2
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Dim i As Integer

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (a) spatial + (1) err_v")
      Sem.AStructure("cubes = (b) spatial + (1) err_c")
      Sem.AStructure("lozenges = (c) spatial + (1) err_l")
      Sem.AStructure("paragraph = (d) verbal + (1) err_p")
      Sem.AStructure("sentence = (e) verbal + (1) err_s")
      Sem.AStructure("w ordmean = (f) verbal + (1) err_w ")
      Sem.Var("spatial", 1)
      Sem.Var("verbal", 1)
      Sem.Model("Congeneric")
      Sem.Model("tau-equivalent", "a = b = c", "d = e = f")

      For i = 1 To 2
         Debug.WriteLine("")
         Debug.WriteLine("Model number " & i)
         Sem.FitModel(i)
         Debug.WriteLine("Chi Square = " & Sem.Cmin)
         Debug.WriteLine("Degrees of Freedom = " & Sem.df)
         Debug.WriteLine("p = " & Sem.p)
         Debug.WriteLine("Number of parameters = " & Sem.npar)
         Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & _
         Sem.NcpLo & ", " & Sem.NcpHi & ")")
         Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
         Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)
      Next

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.21 ColumnNames Method

Obtains the variable names associated with the columns of a matrix of estimates.

Syntax

object.**ColumnNames** (*matrixID, theColumnNames*)
object.**ColumnNames** (*matrixID, theColumnNames, groupNumber*)

The **ColumnNames** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.

theColumnNames	A string array declared as Dim theColumnNames() as String in the calling program. The Amos Engine will redimension the array so that theColumnNames(1) is the first column name in the array.
groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement[156]: [3].

Settings

The settings for *MatrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.

DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

The NeedEstimates 329 method must be used to warn that a particular matrix of estimates will be needed before the **ColumnNames** method can be used to obtain the column names for that matrix. For example, you have to use

object.**NeedEstimates** (ImpliedMeans)

before using

object.**ColumnNames** (ImpliedMeans, ...)

See Also

ColumnNumbers Method 203

RowNames Method 382

RowNumbers Method 386

The following program fits Models A and B of Example 11. The matrix of total effects is displayed for each group and model.

```
Imports AmosEngineLib
Imports AmosEngineLib.AmosEngine.TMatrixID
Imports Microsoft.VisualBasic
Imports System.Diagnostics
Module MainModule
    ' ColumnNames Method Example
    Sub Main()
        Dim CNames() As String, RNames() As String, X(,) As Double
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.NeedEstimates(TotalEffects)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
        Sem.GroupName("girls")
        Sem.AStructure("academic = (g1) GPA + (g2) attract + (1) e1")
        Sem.AStructure( _
        "attract = (g3) height + (g4) w eight + (g5) rating + (g6) academic + (1) e2 ")
        Sem.AStructure("e2 <--> e1")

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
        Sem.GroupName("boys")
        Sem.AStructure("academic = (b1) GPA + (b2) attract + (1) e1")
        Sem.AStructure( _
        "attract = (b3) height + (b4) w eight + (b5) rating + (b6) academic + (1) e2 ")
        Sem.AStructure("e2 <--> e1")

        Sem.Model("Model_A")
        Sem.Model("Model_B", "g1=b1", "g2=b2", "g3=b3", "g4=b4", "g5=b5", "g6=b6")

        'Print total effects for each model and each group
        Dim ModelNumber As Integer
        Dim GroupNumber As Integer
        For ModelNumber = 1 To 2
            Sem.FitModel(ModelNumber)
            For GroupNumber = 1 To 2
                Sem.GetEstimates(TotalEffects, X, GroupNumber)

                Sem.ColumnNames(TotalEffects, CNames, GroupNumber)

                Sem.Row Names(TotalEffects, RNames, GroupNumber)
                Debug.WriteLine(vbCrLf & "Group " & GroupNumber & ", Model " & ModelNumber)
                PrintMatrix(X, CNames, RNames)
            Next
        Next
        Sem.Dispose()
    End Sub

    'Print a matrix in the debug w indow
    Sub PrintMatrix(ByVal TheMatrix(,) As Double, ByVal CNames$(), ByVal RNames$())
        Dim NRow s1 As Integer, NColumns1 As Integer
        Dim i As Integer, j As Integer
        NRow s1 = UBound(RNames)
        NColumns1 = UBound(CNames)

        Debug.Write(" ")
        For j = 0 To NColumns1
            Debug.Write(CNames(j).PadLeft(10))
        Next
        Debug.WriteLine("")

        For i = 0 To NRow s1
```

```
        Debug.Write(RNames(i).PadRight(8))
        For j = 0 To NColumns1
           Debug.Write(TheMatrix(i, j).ToString(".00000").PadLeft(10))
        Next
        Debug.WriteLine("")
     Next
  End Sub
End Module
```

4.5.2.2.2.22 ColumnNumbers Method

Obtains the variable numbers associated with the columns of a matrix.

Syntax

object.**ColumnNumbers** (*matrixID, theVariableNumbers*)
object.**ColumnNumbers** (*matrixID, theVariableNumbers, groupNumber*)

The **ColumnNumbers** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings
theVariableNumbers	An integer array declared in the calling program as **Dim theVariableNumbers() as Integer** The Amos Engine redimensions the array.
groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement 156: [3].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.

ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

The NeedEstimates 329 method must be used to warn that a particular matrix of estimates will be needed before the **ColumnNumbers** method can be used to obtain the column numbers for that matrix. For example, you have to use

object.**NeedEstimates** (ImpliedMeans)

before using

object.**ColumnNumbers** (ImpliedMeans, ...)

See Also

ColumnNames Method 199

RowNames Method 382

RowNumbers Method 386

The following program fits Models A and B of Example 11. The matrix of total effects is displayed for each group and model. Rows and columns of the matrix are labeled with Amos's internal variable numbers.

```
Imports AmosEngineLib
Imports AmosEngineLib.AmosEngine.TMatrixID
Imports System.Diagnostics
Imports Microsoft.VisualBasic
Module MainModule
    ' ColumnNumbers Method Example
    Sub Main()
        Dim CNumbers() As Integer, RNumbers() As Integer, X(,) As Double
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.NeedEstimates(TotalEffects)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
        Sem.GroupName("girls")
        Sem.AStructure("academic = (g1) GPA + (g2) attract + (1) e1")
        Sem.AStructure( _
        "attract = (g3) height + (g4) w eight + (g5) rating + (g6) academic + (1) e2")
        Sem.AStructure("e2 <--> e1")

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
        Sem.GroupName("boys")
        Sem.AStructure("academic = (b1) GPA + (b2) attract + (1) e1")
        Sem.AStructure( _
        "attract = (b3) height + (b4) w eight + (b5) rating + (b6) academic + (1) e2")
        Sem.AStructure("e2 <--> e1")

        Sem.Model("Model_A")
        Sem.Model("Model_B", "g1=b1", "g2=b2", "g3=b3", "g4=b4", "g5=b5", "g6=b6")

        'Print total effects for each model and each group
        Dim ModelNumber As Integer
        Dim GroupNumber As Integer
        For ModelNumber = 1 To 2
            Sem.FitModel(ModelNumber)
            For GroupNumber = 1 To 2
                Sem.GetEstimates(TotalEffects, X, GroupNumber)
                Sem.ColumnNumbers(TotalEffects, CNumbers, GroupNumber)
                Sem.Row Numbers(TotalEffects, RNumbers, GroupNumber)
                Debug.WriteLine(vbCrLf & "Group " & GroupNumber & ", Model " & ModelNumber)
                PrintMatrix1(X, CNumbers, RNumbers)
            Next
        Next
        Sem.Dispose()
    End Sub

    'Print a matrix in the debug w indow
    Sub PrintMatrix1(ByVal TheMatrix(,) As Double, ByVal CNumbers() As Integer, ByVal RNumbers() As Integer)
        Dim NRow s1 As Integer, NColumns1 As Integer
        Dim i As Integer, j As Integer
        NRow s1 = UBound(RNumbers)
        NColumns1 = UBound(CNumbers)

        Debug.Write(" ")
        For j = 0 To NColumns1
            Debug.Write(CNumbers(j).ToString.PadLeft(10))
        Next
        Debug.WriteLine("")
        For i = 0 To NRow s1
            Debug.Write(RNumbers(i).ToString.PadLeft(8))
            For j = 0 To NColumns1
                Debug.Write(TheMatrix(i, j).ToString("#.00000").PadLeft(10))
```

```
        Next
        Debug.WriteLine("")
    Next
  End Sub
End Module
```

4.5.2.2.2.23 ConfidenceBC Method

Controls whether bias-corrected bootstrap confidence intervals are reported.

Syntax

object.**ConfidenceBC** (*confidenceLevel*)

The **ConfidenceBC** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
confidenceLevel	Desired confidence level. If the value of *confidenceLevel* is zero, bias-corrected confidence intervals are not reported.

Placement 156: [1].

Default

Bias-corrected confidence intervals are not reported.

Remarks

Amos can produce bootstrapped confidence intervals for all parameter estimates, as well as for estimates displayed by the methods: Standardized 396, Smc 393, FactorScoreWeights 231, TotalEffects 402, SampleMoments 389, ImpliedMoments 286 and AllImpliedMoments 165.

Bootstrap confidence intervals are reported only for those quantities that are estimated. For example, to obtain bootstrapped standard errors for squared multiple correlations, you need to use the Smc method. Similarly, to obtain bootstrapped standard errors for sample correlations, you must use both SampleMoments 389 and Standardized 396 methods.

When you use **ConfidenceBC**, you must also use Bootstrap 188 to specify the number of bootstrap samples. Note that bias-corrected confidence intervals for estimated indirect effects are provided along with the bias-corrected confidence intervals of the TotalEffects 402 method.

ConfidenceBC requires raw data unless you use MonteCarlo 314.

See Also

Bootstrap Method 188

ConfidencePC Method 209

GetBCLowerBounds, GetBCUpperBounds Methods 243

MonteCarlo Method 314

NeedBCLowerBounds, NeedBCUpperBounds Methods 320

This example demonstrates the ConfidenceBC method.

```
Module MainModule
    ' ConfidenceBC Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.Bootstrap(2000)
        Sem.ConfidenceBC(90) '90% confidence intervals

        Sem.Standardized()

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

In the output from the example, the 90% confidence intervals for the standardized estimates appear as follows.

Standardized Regression Weights: (Group number 1 - Model 1)

Parameter			Estimate	Lower	Upper	P
visperc	<---	spatial	.703	.503	.901	.001
cubes	<---	spatial	.654	.467	.793	.001
lozenges	<---	spatial	.736	.559	.951	.001
paragraph	<---	verbal	.880	.794	.943	.001
sentence	<---	verbal	.827	.750	.890	.001
wordmean	<---	verbal	.841	.731	.912	.002

Correlations: (Group number 1 - Model 1)

Parameter			Estimate	Lower	Upper	P
spatial	<-->	verbal	.487	.250	.681	.003

The confidence interval for the correlation between **spatial** and **verbal**, for example, is [.250, .681]. Since the confidence interval does not include zero, you would reject the hypothesis that the correlation is zero in the population, using a two-sided test with a significance level of .10. To carry out a similar two-sided test with a significance level of .05, you would need to request a 95% confidence interval (**ConfidenceBC 95**). You can also refer to the value in the "p" column. Each "p" value reveals indirectly how small the confidence level would have to be to yield a confidence interval that includes the value zero. A value of *p* in the "p" column indicates that a 100(1-*p*)% confidence interval would have one of its end points at zero. In this sense, a *p* value can be used to test the hypothesis that an estimate has a population value of zero. For example, the correlation between **spatial** and **verbal** has a *p* value of .003, which means that a 99.7% confidence interval would have its lower boundary at zero. In other words, a confidence interval at any conventional confidence level, such as .95 or .99, would not include zero, and you would reject at any conventional significance level the hypothesis that the correlation is zero in the population.

4.5.2.2.2.24 ConfidencePC Method

Controls whether bootstrap confidence intervals obtained by using the percentile method (Efron, 1987 [513]) are reported.

Syntax

object.**ConfidencePC** (*confidenceLevel*)

The **ConfidencePC** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
confidenceLevel	Desired confidence level. If the value of *confidenceLevel* is zero, percentile confidence intervals are not reported.

Placement [156]: [1].

Default

Percentile confidence intervals are not reported.

Remarks

Amos can produce bootstrapped confidence intervals for all parameter estimates, as well as for estimates displayed by the methods: Standardized [396], Smc [393], FactorScoreWeights [231], TotalEffects [402], SampleMoments [389], ImpliedMoments [286] and AllImpliedMoments [165].

Bootstrap confidence intervals are reported only for those quantities that are estimated. For example, to obtain bootstrapped standard errors for squared multiple correlations, you need to use the Smc [393]

method. Similarly, to obtain bootstrapped standard errors for sample correlations, you must use both SampleMoments 389 and Standardized 396.

When you use **ConfidencePC**, you must also use Bootstrap 188 to specify the number of bootstrap samples. Note that percentile confidence intervals for estimated indirect effects are provided along with the percentile confidence intervals of the TotalEffects 402 method.

ConfidencePC requires raw data unless you use MonteCarlo 314.

See Also

Bootstrap Method 188

ConfidenceBC Method 207

GetBCLowerBounds, GetBCUpperBounds Methods 243

MonteCarlo Method 314

NeedBCLowerBounds, NeedBCUpperBounds Methods 320

More:

ConfidencePC Method Example 210

Discussion of the example 210

This example demonstrates the ConfidencePC method.

```
Module MainModule
    ' ConfidencePC Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.Bootstrap(2000)
        Sem.ConfidencePC(90) '90% confidence intervals

        Sem.Standardized()

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()

    End Sub
End Module
```

In the output from the example, the 90% confidence intervals for the standardized estimates appear as follows.

Standardized Regression Weights: (Group number 1 - Model 1)

Parameter			Estimate	Lower	Upper	P
visperc	<---	spatial	.703	.506	.906	.001
cubes	<---	spatial	.654	.475	.799	.001
lozenges	<---	spatial	.736	.535	.926	.001
paragraph	<---	verbal	.880	.794	.943	.001
sentence	<---	verbal	.827	.749	.890	.001
wordmean	<---	verbal	.841	.741	.919	.001

Correlations: (Group number 1 - Model 1)

Parameter			Estimate	Lower	Upper	P
spatial	<-->	verbal	.487	.277	.705	.002

The confidence interval for the correlation between **spatial** and **verbal**, for example, is [.277, .705]. Since the confidence interval does not include zero, you would reject the hypothesis that the correlation is zero in the population, using a two-sided test with a significance level of .10. To carry out a similar two-sided test with a significance level of .05, you would need to request a 95% confidence interval (**`Confidencebc 95`**). You can also refer to the value in the "p" column. Each "p" value reveals indirectly how small the confidence level would have to be to yield a confidence interval that includes the value zero. A value of *p* in the "p" column indicates that a 100(1-*p*)% confidence interval would have one of its end points at zero. In this sense, a *p* value can be used to test the hypothesis that an estimate has a population value of zero. For example, the correlation between **spatial** and **verbal** has a *p* value of .002, which implies that a 99.8% confidence interval would have its lower boundary at zero. In other words, a confidence interval at any conventional confidence level, such as .95 or .99, would not include zero, and you would reject at any conventional significance level the hypothesis that the correlation is zero in the population.

4.5.2.2.2.25 Corest Method

Controls whether an estimate of the correlation matrix of the parameter estimates is reported.

Syntax

object.**`Corest`** ()
object.**`Corest`** (*tf*)

The **`Corest`** method syntax has the following parts:

Part	Description
object	An object of type **`AmosEngine`**.

tf	Optional. If *tf* is True (default), an estimate of the correlation matrix of the parameter estimates is calculated. Otherwise, not.

Placement[156]: [1].

Default

The correlation matrix of parameter estimates is not estimated.

Remarks

An estimate of the correlation matrix of the parameter estimates is available only for the case of maximum likelihood (Ml[305]), generalized least squares (Gls[284]), and asymptotically distribution-free (Adf[163]) estimation. When other estimation criteria are used, the **Corest** method is ignored.

See Also

Covest Method[215]

Crdiff Method[216]

This example demonstrates the Corest method.

```
Module MainModule
    ' Corest Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.Corest()
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.26 Cov Method

Specifies a covariance as a model parameter.

Syntax

object.**Cov** (*leftVariableName*, *rightVariableName*)
object.**Cov** (*leftVariableName*, *rightVariableName*, *parameterValue*)
object.**Cov** (*leftVariableName*, *rightVariableName*, *parameterName*)

The **Cov** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
leftVariableName *rightVariableName*	The character strings, leftVariableName and rightVariableName, are the names of two exogenous variables. The Cov method makes their covariance a model parameter.
parameterValue	(Optional) Parameter value. If *parameterValue* = 3, say, then the covariance is fixed at 3.
parameterName	(Optional) Parameter name. If parameterName = "abc", say, then the covariance is named "abc". It is constrained to be equal to any other parameters named "abc". If parameterName = "3?", say then the covariance is given an initial value of 3, and is unconstrained. If parameterName = "abc:3", say, then the covariance is named "abc" and is given an initial value of 3. It is constrained to be equal to any other parameters named "abc". If parameterName is an empty string (""), the covariance is an unconstrained parameter.

Placement[156]: [2].

Default

With one exception, Amos assumes that the *exogenous variables in a model are correlated*, and it estimates the covariance between every pair of exogenous variables. The exception to this default assumption concerns unique variables - exogenous variables that are unobserved and have a direct effect on only one variable. Amos assumes that *unique variables are uncorrelated* with each other, and with every other exogenous variable in the model.

You can modify these defaults with the GenerateDefaultCovariances[241] method.

Remarks

If *parameterValue* and *parameterName* are omitted, the covariance is an unconstrained parameter.

See Also

AStructure Method 169

Intercept Method 299

Mean Method 303

MStructure Method 315

Path Method 364

Var Method 405

The following program uses the **Path** 364, **Cov** 212 and **Var** 405 methods to specify Model C of Example 6.

```
Module MainModule
   ' Cov Method Example
   Sub Main()
      Dim Sem As New AmosEngineLib.AmosEngine

      Sem.TextOutput()
      Sem.Standardized()
      Sem.Smc()
      Sem.AllImpliedMoments()
      Sem.FactorScoreWeights()
      Sem.TotalEffects()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Wheaton")
      Sem.Path("anomia67", "67_alienation", 1)
      Sem.Path("anomia67", "eps1", 1)
      Sem.Path("pow les67", "67_alienation", "path_p")
      Sem.Path("pow les67", "eps2", 1)
      Sem.Path("anomia71", "71_alienation", 1)
      Sem.Path("anomia71", "eps3", 1)
      Sem.Path("pow les71", "71_alienation", "path_p")
      Sem.Path("pow les71", "eps4", 1)
      Sem.Path("67_alienation", "ses")
      Sem.Path("67_alienation", "zeta1", 1)
      Sem.Path("71_alienation", "67_alienation")
      Sem.Path("71_alienation", "ses")
      Sem.Path("71_alienation", "zeta2", 1)
      Sem.Path("education", "ses", 1)
      Sem.Path("education", "delta1", 1)
      Sem.Path("SEI", "ses")
      Sem.Path("SEI", "delta2", 1)
      Sem.Cov("eps3", "eps1")
      Sem.Var("eps1", "var_a")
      Sem.Var("eps2", "var_p")
      Sem.Var("eps3", "var_a")
      Sem.Var("eps4", "var_p")

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.27 Covest Method

Controls whether an estimate of the covariance matrix of the parameter estimates is reported.

Syntax

object.**Covest** ()
object.**Covest** (*tf*)

The **Covest** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. If *tf* is True (default), an estimate of the covariance matrix of the parameter estimates is calculated. Otherwise, not.

Placement[156]: [1].

Default

The covariance matrix of parameter estimates is not estimated.

Remarks

An estimate of the covariance matrix of the parameter estimates is available only for the case of maximum likelihood (Ml[305]), generalized least squares (Gls[284]), and asymptotically distribution-free (Adf[163]) estimation. When other estimation criteria are used, the **Covest** method is ignored.

See Also

Corest Method[211]

Crdiff Method[216]

This example demonstrates the Covest method.

```
Module MainModule
    ' Covest Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.Covest()
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.28 Crdiff Method

Controls whether Amos calculates a 'critical ratio' for each pair of parameter estimates. The critical ratio for a pair of parameter estimates provides a test of the hypothesis that the two parameters are equal.

Syntax

object.**Crdiff** ()
object.**Crdiff** (*tf*)

The **Crdiff** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. If *tf* is True (default), Amos calculates critical ratios for differences between parameters. Otherwise, not.

Placement[156]: [1].

Default

Critical ratios for differences between parameters are not calculated.

Remarks

Critical ratios for differences between parameters are available only for the case of maximum likelihood (Ml[305]), generalized least squares (Gls[284]), and asymptotically distribution-free (Adf[163]) estimation. When other estimation criteria are used, the **Crdiff** method is ignored.

See Also

Corest Method 211

Covest Method 215

This example demonstrates the Crdiff method.

```
Module MainModule
   ' Crdiff Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine

      Sem.Crdiff()
      Sem.TextOutput()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.AStructure("w ordmean = verbal + (1) err_w ")

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.29 Crit1 Method

Affects one of the convergence criteria used in deciding whether a local minimum has been reached. Amos requires the absolute value of each first order derivative to be less than the value specified by **Crit1** at the end of the final iteration.

Syntax

object.**Crit1** (*threshold*)

The **Crit1** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
threshold	Threshold for the absolute value of each first order derivative.

Placement 156: [1].

Default

The absolute value of each first order derivative is required to be less than .00001 at the end of the final iteration.

See Also

Crit2 Method 218

Fisher Method 233

Iterations Method 301

Technical Method 397

Time Method 400

This example demonstrates the Crit1 method.

```
Module MainModule
    ' Crit1 Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.Crit1(0.0000001)

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.30 Crit2 Method

Affects one of the convergence criteria used in deciding whether a local minimum has been reached. Amos requires that, on the final iteration, the distance traveled in the parameter space (the square root of the sum of squared changes in the parameter values) be less than the threshold specified by **Crit2**.

Syntax

object.**Crit2** (*threshold*)

The **Crit2** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
threshold	Threshold for the distance traveled in the parameter space on the final iteration.

Placement 156: [1].

Default

The distance traveled in the parameter space is required to be less than .001 on the final iteration.

See Also

Crit1 Method 217

Fisher Method 233

Iterations Method 301

Technical Method 397

Time Method 400

This example demonstrates the Crit2 method.

```
Module MainModule
    ' Crit2 Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine

        Sem.Crit2(0.0001)

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.31 DataFileNCases Method

Gets the number of cases in a dataset.

Syntax

object.**DataFileNCases** ()
object.**DataFileNCases** (*groupNumber*)

The **DataFileNCases** method syntax has the following parts:

Part	Description

object	An object of type **AmosEngine**.
groupNumber	Optional number. If *groupNumber* is present, it is a group number, where the first group is group number 1. If *groupNumber* is omitted, the method returns the number of cases in the group referred to by the most recent use of the BeginGroupEx 175 method.

Placement 156: [2].

See Also

DataFileNVariables Method 220

This example demonstrates the DataFileNCases method.

```
Imports System.Diagnostics
Module MainModule
    ' DataFileNCases Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
        Sem.GroupName("girls")
        Debug.WriteLine(Sem.DataFileNCases & " girls")
        Sem.AStructure("academic = GPA + attract + e1 (1)")
        Sem.AStructure("attract = height + w eight + rating + academic + e2 (1)")
        Sem.AStructure("e2 <--> e1")

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
        Sem.GroupName("boys")
        Debug.WriteLine(Sem.DataFileNCases & " boys")
        Sem.AStructure("academic = GPA + attract + e1 (1)")
        Sem.AStructure("attract = height + w eight + rating + academic + e2 (1)")
        Sem.AStructure("e2 <--> e1")

        'Once again:
        Debug.WriteLine(Sem.DataFileNCases(1) & " girls")
        Debug.WriteLine(Sem.DataFileNCases(2) & " boys")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.32 DataFileNVariables Method

Gets the number of variables in a dataset.

Syntax

object.**DataFileNVariables** ()
object.**DataFileNVariables** (*groupNumber*)

The **DataFileNVariables** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
groupNumber	Optional number. If *groupNumber* is present, it is a group number, where the first group is group number 1. If *groupNumber* is omitted, the method returns the number of variables in the dataset referred to by the most recent use of the BeginGroupEx 175 method.

Placement 156: [2].

See Also

DataFileNCases Method 219

This example demonstrates the DataFileNVariables method.

```
Imports System.Diagnostics
Module MainModule
   ' DataFileNVariables Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
      Sem.GroupName("girls")
      Debug.WriteLine(Sem.DataFileNVariables & " variables in the girls' group")
      Sem.AStructure("academic = GPA + attract + e1 (1)")
      Sem.AStructure("attract = height + w eight + rating + academic + e2 (1)")
      Sem.AStructure("e2 <--> e1")

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
      Sem.GroupName("boys")
      Debug.WriteLine(Sem.DataFileNVariables & " variables in the boys' group")
      Sem.AStructure("academic = GPA + attract + e1 (1)")
      Sem.AStructure("attract = height + w eight + rating + academic + e2 (1)")
      Sem.AStructure("e2 <--> e1")

      'Once again:
      Debug.WriteLine(Sem.DataFileNVariables(1) & " variables in the girls' group")
      Debug.WriteLine(Sem.DataFileNVariables(2) & " variables in the boys' group")

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.33 Df Method

Gets the degrees of freedom.

Syntax

result = *object*.**Df** ()

The **Df** method syntax has the following parts:

Part	Description
result	The degrees of freedom.
object	An object of type **AmosEngine**.

Placement[156]: [3].

Remarks

If you have used the Model[306] method to define more than one model, the **Df** method returns the degrees of freedom for the most recently fitted model. The second example shows how to obtain the degrees of freedom for multiple models.

See Also

Npar Method[343]

The following program shows how to obtain various fit measures when only one model is defined (*i.e.*, when the Model[306] method has been used only once or not at all).

```
Imports System.Diagnostics
Module MainModule
    ' Df Method Example 1
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Debug.WriteLine("Chi Square = " & Sem.Cmin)
        Debug.WriteLine("Degrees of Freedom = " & Sem.df)
        Debug.WriteLine("p = " & Sem.p)
        Debug.WriteLine("Number of parameters = " & Sem.npar)
        Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & _
        Sem.NcpLo & ", " & Sem.NcpHi & ")")
        Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
        Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)

        Sem.Dispose()
    End Sub
End Module
```

The following program shows how to obtain various fit measures for multiple models (i.e., when the Model[306] method has been used more than once).

```
Imports System.Diagnostics
Module MainModule
   ' Df Method Example 2
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Dim i As Integer

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (a) spatial + (1) err_v")
      Sem.AStructure("cubes = (b) spatial + (1) err_c")
      Sem.AStructure("lozenges = (c) spatial + (1) err_l")
      Sem.AStructure("paragraph = (d) verbal + (1) err_p")
      Sem.AStructure("sentence = (e) verbal + (1) err_s")
      Sem.AStructure("w ordmean = (f) verbal + (1) err_w ")
      Sem.Var("spatial", 1)
      Sem.Var("verbal", 1)

      Sem.Model("Congeneric")
      Sem.Model("tau-equivalent", "a = b = c", "d = e = f")

      For i = 1 To 2
         Debug.WriteLine("")
         Debug.WriteLine("Model number " & i)
         Sem.FitModel(i)
         Debug.WriteLine("Chi Square = " & Sem.Cmin)
         Debug.WriteLine("Degrees of Freedom = " & Sem.df)
         Debug.WriteLine("p = " & Sem.p)
         Debug.WriteLine("Number of parameters = " & Sem.npar)
         Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & _
               Sem.NcpLo & ", " & Sem.NcpHi & ")")
         Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
         Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)
      Next

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.34 Dispose Method

Releases resources used by an **AmosEngine** object. Only one instance of the **AmosEngine** class can exist at a time, so it is essential that the resources used by one instance be released before another instance is created.

Syntax

object.**Dispose** ()

The **Dispose** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement [156]: [3].

4.5.2.2.2.35 Emulisrel6 Method

Substitutes (D1a) for (D1) in Appendix B.

Syntax

object.**Emulisrel6** ()
object.**Emulisrel6** (*tf*)

The Emulisrel6 method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	If *tf* is True (default) then (D1a) is minimized. Otherwise, (D1) is minimized.

Placement[156]: [1].

Default

(D1) is minimized.

Remarks

The effect of using Emulisrel6 is usually too small to matter. It has no effect at all in single group analyses. (D1a) appears to be the function minimized by the Lisrel program (Jöreskog & Sörbom, 1989[517]).

See Also

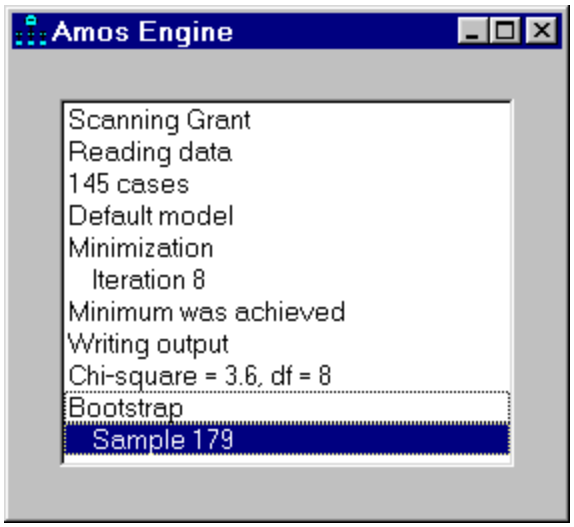
ChiCorrect Method[194]

This example demonstrates the Emulisrel6 method.

```
Module MainModule
   ' Emulisrel6 Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine

      Sem.Emulisrel6()

      Sem.TextOutput()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.AStructure("w ordmean = verbal + (1) err_w ")

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.36 EnableDisplay Method

Controls whether the Amos Engine displays a progress window.



Syntax

object.**EnableDisplay** (*tf*)

The **EnableDisplay** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

tf	If *tf* is True (default), the progress window is displayed. Otherwise, not.

Placement[156]: [1].

Default

The progress window is displayed.

4.5.2.2.2.37 Evaluate0 and EvaluateEx0 Methods

Both **Evaluate0** and **EvaluateEx0** calculate the discrepancy function. **EvaluateEx0** has the additional side effect of calculating derived estimates (implied moments, indirect effects, factor score weights, and so forth), which can subsequently be retrieved using GetEstimates[260] or GetEstimatesEx[265].

Syntax

object.**Evaluate0** (*ind, f*)
object.**EvaluateEx0** (*ind, f*)

The **Evaluate0** and **EvaluateEx0** method syntaxes have the following parts:

Part	Description
object	An object of type **AmosEngine**.
ind	A variable of type Integer. On return, ind=0 if the discrepancy function is defined at the current parameter values. ind<>0 if the discrepancy function is undefined.
f	A variable of type Double. On return, if ind=0, f is the value of the discrepancy function evaluated at the current parameter values.

Placement[156]: [3].

Remarks

Evaluate0 and **EvaluateEx0** evaluate the discrepancy function at the current parameter values, which can be retrieved using ParameterValue[362] or ParameterVector[364], and which can be set using PutParameterValue[372] or PutParameterVector[373].

See Use the AmosEngine class to evaluate derivatives numerically and display the results with the Amos Debug class[499]

4.5.2.2.2.38 Evaluate1 and EvaluateEx1 Methods

Both **Evaluate1** and **EvaluateEx1** calculate the discrepancy function and its first derivatives. **EvaluateEx1** has the additional side effect of calculating derived estimates (implied moments, indirect

effects, factor score weights, and so forth), which can subsequently be retrieved using GetEstimates 260 or GetEstimatesEx 265.

Syntax

object.**Evaluate1** (*ind, f, g*)
object.**EvaluateEx1** (*ind, f, g*)

The **Evaluate1** and **EvaluateEx1** method syntaxes have the following parts:

Part	Description
object	An object of type **AmosEngine**.
ind	A variable of type Integer. On return, ind=0 if the discrepancy function is defined at the current parameter values. ind<>0 if the discrepancy function is undefined.
f	A variable of type Double. On return, if ind=0, f is the value of the discrepancy function evaluated at the current parameter values.
g	An array of type double. On return, if ind=0, g is a one-dimensional array containing the first derivatives of the discrepancy function evaluated at the current parameter values.

Placement 156: [3].

Remarks

Evaluate1 and **EvaluateEx1** evaluate the discrepancy function and derivatives at the current parameter values, which can be retrieved using ParameterValue 362 or ParameterVector 364, and which can be set using PutParameterValue 372 or PutParameterVector 373.

See Use the AmosEngine class to evaluate derivatives numerically and display the results with the Amos Debug class 499

4.5.2.2.2.39 Evaluate2a and EvaluateEx2a Methods

Evaluate2a and **EvaluateEx2a** calculate the discrepancy function and its first derivatives.

Evaluate2a calculates approximate second derivatives (twice the Fisher information matrix in the case of maximum likelihood estimation) while **EvaluateEx2a** calculates twice the inverse of the matrix of approximate second derivatives.

EvaluateEx2a has the additional side effect of calculating derived estimates (implied moments, indirect effects, factor score weights, and so forth), which can subsequently be retrieved using GetEstimates 260 or GetEstimatesEx 265.

Syntax

object.**Evaluate2a** *ind, f, g, h*
object.**EvaluateEx2a** *ind, f, g, v*

The **Evaluate2a** and **EvaluateEx2a** method syntaxes have the following parts:

Part	Description
object	An object of type **AmosEngine**.
ind	A variable of type Integer. On return, ind=0 if the discrepancy function is defined at the current parameter values. ind<>0 if the discrepancy function is undefined.
f	A variable of type Double. On return, if ind=0, f is the value of the discrepancy function evaluated at the current parameter values.
g	An array of type double. On return, if ind=0, g is a one-dimensional array containing the first derivatives of the discrepancy function evaluated at the current parameter values.
h	An array of type double. On return, if ind=0, h is a one-dimensional array containing approximate second derivatives of the discrepancy function evaluated at the current parameter values.
v	An array of type double. On return, if ind=0, v is a one-dimensional array containing twice the inverse of the approximate second derivatives, provided that the inverse was successfully calculated. Use WasInverted after EvaluateEx2a to learn whether the inverse was successfully calculated.

Placement 156: [3].

Remarks

Evaluate2a and **EvaluateEx2a** evaluate the discrepancy function and derivatives at the current parameter values, which can be retrieved using ParameterValue 362 or ParameterVector 364, and which can be set using PutParameterValue 372 or PutParameterVector 373.

The following program fits the model of Example 8. It then displays two matrices: 1) the matrix of approximate second derivatives, and 2) two times the inverse of the matrix of approximate second derivatives.

```
Module MainModule
    ' Evaluate2a and EvaluateEx2a methods example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Dim ad As New AmosDebug.AmosDebug
        Dim Originalparameters() As Double
        Sem.Covest()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = (a) spatial + (1) err_c")
        Sem.AStructure("lozenges = (b) spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = (c) verbal + (1) err_s")
        Sem.AStructure("w ordmean = (d) verbal + (1) err_w ")

        If (Sem.FitModel() = 0) Then
            Dim Ind As Integer
            Dim F As Double
            Dim G() As Double
            Dim H() As Double
            Dim Hl() As Double

            Sem.Evaluate2a(Ind, F, G, H)

            If Ind = 0 Then
                ad.PrintTriangle(H, "Approximate 2nd derivatives")
            Else
                ad.PrintX("Approximate 2nd derivatives could not be calculated.")
            End If

            Sem.EvaluateEx2a(Ind, F, G, Hl)

            If Ind = 0 Then
                If Sem.WasInverted Then
                    ad.PrintTriangle(Hl, "Approximate 2nd derivatives inverted (times 2)")
                Else
                    ad.PrintX("Matrix of 2nd derivatives could not be inverted.")
                End If
            End If
        End If

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.40 Evaluate2e and EvaluateEx2e Methods

Evaluate2e and **EvaluateEx2e** calculate the discrepancy function and its first derivatives.

Evaluate2e calculates second derivatives (twice the observed Fisher information matrix, in the case of maximum likelihood estimation) while **EvaluateEx2e** calculates twice the inverse of the matrix of second derivatives.

EvaluateEx2e has the additional side effect of calculating derived estimates (implied moments, indirect effects, factor score weights, and so forth), which can subsequently be retrieved using GetEstimates 260 or GetEstimatesEx 265.

Syntax

object.**Evaluate2e** (*ind, f, g, h*)
object.**EvaluateEx2e** (*ind, f, g, v*)

The **Evaluate2e** and **EvaluateEx2e** method syntaxes have the following parts:

Part	Description
object	An object of type **AmosEngine**.
ind	A variable of type Integer. On return, ind=0 if the discrepancy function is defined at the current parameter values. ind<>0 if the discrepancy function is undefined.
f	A variable of type Double. On return, if ind=0, f is the value of the discrepancy function evaluated at the current parameter values.
g	An array of type double. On return, if ind=0, g is a one-dimensional array containing the first derivatives of the discrepancy function evaluated at the current parameter values.
h	An array of type double. On return, if ind=0, h is a one-dimensional array containing second derivatives of the discrepancy function evaluated at the current parameter values.
v	An array of type double. On return, if ind=0, v is a one-dimensional array containing twice the inverse of the second derivatives, provided that the inverse was successfully calculated. Use WasInverted 410 after EvaluateEx2e to learn whether the inverse was successfully calculated.

Placement 156: [3].

Remarks

Evaluate2e and **EvaluateEx2e** evaluate the discrepancy function and derivatives at the current parameter values, which can be retrieved using ParameterValue 362 or ParameterVector 364, and which can be set using PutParameterValue 372 or PutParameterVector 373.

See Use the AmosEngine class to evaluate derivatives numerically and display the results with the Amos Debug class 499

The following program fits the model of Example 8. It then displays two matrices: 1) the matrix of second derivatives, and 2) two times the inverse of the matrix of second derivatives.

```
Module MainModule
    ' Evaluate2e and EvaluateEx2e methods example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Dim ad As New AmosDebug.AmosDebug
        Dim Originalparameters() As Double
        Sem.Covest()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\userguide.xls", "grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = (a) spatial + (1) err_c")
        Sem.AStructure("lozenges = (b) spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = (c) verbal + (1) err_s")
        Sem.AStructure("w ordmean = (d) verbal + (1) err_w ")

        If (Sem.FitModel() = 0) Then
            Dim Ind As Integer
            Dim F As Double
            Dim G() As Double
            Dim H() As Double
            Dim HI() As Double

            Sem.Evaluate2e(Ind, F, G, H)

            If Ind = 0 Then

                ad.PrintTriangle(H, "2nd derivatives")
            Else
                ad.PrintX("2nd derivatives could not be calculated.")
            End If

            Sem.EvaluateEx2e(Ind, F, G, HI)

            If Ind = 0 Then

                If Sem.WasInverted Then
                    ad.PrintTriangle(HI, "2nd derivatives inverted (times 2)")
                Else
                    ad.PrintX("Matrix of 2nd derivatives could not be inverted.")
                End If
            End If
        End If

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.41 FactorScoreWeights Method

Controls whether regression weights for predicting the unobserved variables from the observed variables are displayed. The regression weights are computed by the formula $\mathbf{W} = \mathbf{B}\mathbf{S}^{-1}$ where

W is the matrix of regression weights

S is the matrix of covariances among the observed variables

B is the matrix of covariances between the unobserved and observed variables.

Syntax

object.**FactorScoreWeights** ()
object.**FactorScoreWeights** (*tf*)

The **FactorScoreWeights** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. If *tf* is True (the default), regression weights are reported. Otherwise, not.

Placement[156]: [1].

Default

Factor score weights are not reported.

Remarks

This method is called "**FactorScoreWeights**" in conformance with usage in common factor analysis, where scores on the unobserved variables are called 'factor scores'. The use of **FactorScoreWeights** is not limited to common factor analysis models, however.

See Also

GetEstimates Method[260]

NeedEstimates Method[329]

This example demonstrates the **FactorScoreWeights** method.

```
Module MainModule
    ' FactorScoreWeights Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine

        Sem.FactorScoreWeights()
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.42 Fisher Method

Employs Fisher's scoring method in the case of maximum likelihood estimation (Ml [305]), or the Gauss-Newton method in the case of least squares estimation (Uls [403], Sls [392], Gls [284] or Adf [163]).

Syntax

object.**Fisher** (*nIterations*)

The **Fisher** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
nIterations	Upper limit on the number of iterations that will be performed by Fisher's scoring method or the Gauss-Newton method. If Amos has not obtained a solution by the time this limit is reached, the program will revert to the standard Amos algorithm.

Placement [156]: [1].

Default

Only exact derivatives are used.

Remarks

For some combinations of a model with an estimation method, the **Fisher** method is highly effective, and may even converge in a single iteration (Kendall & Stuart, 1973 [518], Section 18.21). However, **Fisher** usually makes Amos slower and less reliable.

See Also

Crit1 Method 217

Crit2 Method 218

Iterations Method 301

Technical Method 397

Time Method 400

This example demonstrates the **Fisher** method.

```
Module MainModule
    ' Fisher Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.Fisher(5)
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.43 FitAllModels Method

Fits all models.

Syntax

status = *object*.**FitAllModels** ()

The **FitAllModels** method syntax has the following parts:

Part	Description
status	Zero if the most recently analyzed model was successfully fitted. Nonzero otherwise.
object	An object of type **AmosEngine**.

Placement 156: [3].

Remarks

The **FitAllModels** method causes all models to be fitted.

If you have used the Model 306 method to specify multiple models, and want to have access to the results from models other than the last model fitted, use the FitModel 237 method once for each model.

See Also

FitModel Method 237

This example demonstrates the **FitAllModels** method.

```
Imports System.Diagnostics
Module MainModule
    ' FitAllModels Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.TextOutput()
        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.FitAllModels()

        Debug.WriteLine("Chi Square = " & Sem.Cmin)
        Debug.WriteLine("Degrees of Freedom = " & Sem.df)
        Debug.WriteLine("p = " & Sem.p)
        Debug.WriteLine("Number of parameters = " & Sem.npar)
        Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & _
                Sem.NcpLo & ", " & Sem.NcpHi & ")")
        Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
        Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.44 FitMLMoments Method

Fits the model to the sample covariance matrix that is the maximum likelihood estimate (rather than the unbiased estimate). In other words, the sample covariance matrix is composed of sums of squares and cross products divided by **N** (rather than by **N – 1)**.

Syntax

object.**FitMLMoments** ()

The **FitMLMoments** method syntax has the following parts:

Part	Description

object	An object of type **AmosEngine**.

Placement[156]: [1].

Default

Amos fits the model to the biased sample covariance matrix, which is the maximum likelihood estimate, unless you use the FitUnbiasedMoments[239] method.

Remarks

FitMLMoments and InputMLMoments[289] have different effects. InputMLMoments[289] specifies that any sample covariance matrix that is read from a data file is a maximum likelihood estimate. FitMLMoments, on the other hand, tells Amos to *fit the model* to the sample covariance matrix ($\mathbf{S}^{(g)}$ in Appendices A and B) that is the maximum likelihood estimate.

See Also

FitUnbiasedMoments Method[239]

InputMLMoments Method[289]

InputUnbiasedMoments Method[291]

This example demonstrates the **FitMLMoments** method.

```
Module MainModule
    ' FitMLMoments Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.TextOutput()

        Sem.InputMLMoments()
        Sem.FitMLMoments()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Wheaton")
        Sem.AStructure("anomia67 <--- 67_alienation (1)")
        Sem.AStructure("anomia67 <--- eps1 (1)")
        Sem.AStructure("powles67 <--- 67_alienation (path_p)")
        Sem.AStructure("powles67 <--- eps2 (1)")
        Sem.AStructure("anomia71 <--- 71_alienation (1)")
        Sem.AStructure("anomia71 <--- eps3 (1)")
        Sem.AStructure("powles71 <--- 71_alienation (path_p)")
        Sem.AStructure("powles71 <--- eps4 (1)")
        Sem.AStructure("67_alienation <--- ses")
        Sem.AStructure("67_alienation <--- zeta1 (1)")
        Sem.AStructure("71_alienation <--- 67_alienation")
        Sem.AStructure("71_alienation <--- ses")
        Sem.AStructure("71_alienation <--- zeta2 (1)")
        Sem.AStructure("education <--- ses (1)")
        Sem.AStructure("education <--- delta1 (1)")
        Sem.AStructure("SEI <--- ses")
        Sem.AStructure("SEI <--- delta2 (1)")
        Sem.AStructure("eps3 <--> eps1")
        Sem.AStructure("eps1 (var_a)")
        Sem.AStructure("eps2 (var_p)")
        Sem.AStructure("eps3 (var_a)")
        Sem.AStructure("eps4 (var_p)")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.45 FitModel Method

Fits a single model.

Syntax

status = *object*.**FitModel** ()
status = *object*.**FitModel** (*modelName*)
status = *object*.**FitModel** (*modelNumber*)

The **FitModel** method syntax has the following parts:

Part	Description
status	Zero if the model was successfully fitted. Nonzero otherwise.

object	An object of type **AmosEngine**.
modelName	The name of a model.
modelNumber	(Integer) A model number. The first model is model number 1.

Placement[156]: [3].

Remarks

If neither modelName nor modelNumber is specified, model number 1 is fitted.

All other methods that can be used to obtain the results of an analysis (such as Rmsea[380], Cmin[197], GetEstimates[260], and so forth) will provide results only for the most recently fitted model. Say that you want to print the RMSEA for each of several models. This can be done as follows.

```
Dim Sem As AmosEngine
. . .
Sem.FitModel 1
Debug.Print Sem.Rmsea
Sem.FitModel 2
Debug.Print Sem.Rmsea
Sem.FitModel 3
Debug.Print Sem.Rmsea
. . .
```

See Also

FitAllModels Method[234]

The following program shows how to display various fit measures for multiple models (i.e., when the Model[306] method has been used more than once).

```
Imports System.Diagnostics
Module MainModule
   ' FitModel Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Dim i As Integer

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (a) spatial + (1) err_v")
      Sem.AStructure("cubes = (b) spatial + (1) err_c")
      Sem.AStructure("lozenges = (c) spatial + (1) err_l")
      Sem.AStructure("paragraph = (d) verbal + (1) err_p")
      Sem.AStructure("sentence = (e) verbal + (1) err_s")
      Sem.AStructure("w ordmean = (f) verbal + (1) err_w ")
      Sem.Var("spatial", 1)
      Sem.Var("verbal", 1)

      Sem.Model("Congeneric")
      Sem.Model("tau-equivalent", "a = b = c", "d = e = f")

      For i = 1 To 2
         Debug.WriteLine("")
         Debug.WriteLine("Model number " & i)

         Sem.FitModel(i)

         Debug.WriteLine("Chi Square = " & Sem.Cmin)
         Debug.WriteLine("Degrees of Freedom = " & Sem.df)
         Debug.WriteLine("p = " & Sem.p)
         Debug.WriteLine("Number of parameters = " & Sem.npar)
         Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & _
                  Sem.NcpLo & ", " & Sem.NcpHi & ")")
         Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
         Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)
      Next

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.46 FitUnbiasedMoments Method

Fits the model to the unbiased sample covariance matrix (rather than the maximum likelihood estimate). In other words, the sample covariance matrix is composed of sums of squares and cross products divided by **N - 1** (rather than by **N**).

Syntax

object.**FitUnbiasedMoments** ()

The **FitUnbiasedMoments** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement[156]: [1].

Default

Amos fits the model to the biased sample covariance matrix, which is the maximum likelihood estimate, unless you use the **FitUnbiasedMoments** method.

Remarks

FitUnbiasedMoments and **InputUnbiasedMoments**[291] have different effects. **InputUnbiasedMoments**[291] specifies that any sample covariance matrix that is read from a data file is an unbiased estimate. FitUnbiasedMoments, on the other hand, tells Amos to *fit the model* to the sample covariance matrix ($\mathbf{S}^{(g)}$ in the *User's Guide*, Appendices A and B) that is an unbiased estimate.

See Also

FitMLMoments Method[235]

InputMLMoments Method[289]

InputUnbiasedMoments Method[291]

This example demonstrates the **FitUnbiasedMoments** method.

```
Module MainModule
    ' FitUnbiasedMoments Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.TextOutput()

        Sem.InputUnbiasedMoments()
        Sem.FitUnbiasedMoments()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Wheaton")
        Sem.AStructure("anomia67 <--- 67_alienation (1)")
        Sem.AStructure("anomia67 <--- eps1 (1)")
        Sem.AStructure("powles67 <--- 67_alienation (path_p)")
        Sem.AStructure("powles67 <--- eps2 (1)")
        Sem.AStructure("anomia71 <--- 71_alienation (1)")
        Sem.AStructure("anomia71 <--- eps3 (1)")
        Sem.AStructure("powles71 <--- 71_alienation (path_p)")
        Sem.AStructure("powles71 <--- eps4 (1)")
        Sem.AStructure("67_alienation <--- ses")
        Sem.AStructure("67_alienation <--- zeta1 (1)")
        Sem.AStructure("71_alienation <--- 67_alienation")
        Sem.AStructure("71_alienation <--- ses")
        Sem.AStructure("71_alienation <--- zeta2 (1)")
        Sem.AStructure("education <--- ses (1)")
        Sem.AStructure("education <--- delta1 (1)")
        Sem.AStructure("SEI <--- ses")
        Sem.AStructure("SEI <--- delta2 (1)")
        Sem.AStructure("eps3 <--> eps1")
        Sem.AStructure("eps1 (var_a)")
        Sem.AStructure("eps2 (var_p)")
        Sem.AStructure("eps3 (var_a)")
        Sem.AStructure("eps4 (var_p)")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.47 GenerateDefaultCovariances Method

Controls whether exogenous variables (except for unique variables) are allowed to covary by default.

Syntax

object.**GenerateDefaultCovariances** ()
object.**GenerateDefaultCovariances** (*tf*)

The **GenerateDefaultCovariances** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

tf	Optional. If *tf* is True (the default), exogenous variables (with the exception of residual variables) are allowed to covary by default. If *tf* is False, all exogenous variables are assumed to be uncorrelated.

Placement[156]: [1].

Default

Residual variables are assumed to be uncorrelated among themselves and with every other exogenous variable. Exogenous variables that are not residual variables are assumed to be correlated.

GenerateDefaultCovariances is not used in the following program, taken from Example 8 . Therefore, by default, **spatial** and **verbal** are assumed to be correlated.

```
Module MainModule
   ' GenerateDefaultCovariances Method Example 1
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Sem.TextOutput()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.AStructure("w ordmean = verbal + (1) err_w ")

      Sem.Dispose()
   End Sub
End Module
```

In the following program, the **GenerateDefaultCovariances** method prevents the automatic generation of a covariance parameter for the two variables, **spatial** and **verbal**. **Spatial** and **verbal** are therefore required to be uncorrelated.

```
Module MainModule
   ' GenerateDefaultCovariances Method Example 2
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine

      Sem.GenerateDefaultCovariances(False)

      Sem.TextOutput()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.AStructure("w ordmean = verbal + (1) err_w ")

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.48 GetBCLowerBounds, GetBCUpperBounds Methods

Gets the lower and upper bounds of the bias-corrected bootstrap confidence intervals for the elements of a matrix of estimates.

Syntax

object.**GetBCLowerBounds** (*matrixID*, *theMatrixBase0*)
object.**GetBCLowerBounds** (*matrixID*, *theMatrixBase0*, *groupNumber*)

object.**GetBCUpperBounds** (*matrixID*, *theMatrixBase0*)
object.**GetBCUpperBounds** (*matrixID*, *theMatrixBase0*, *groupNumber*)

The **GetBCLowerBounds** and **GetBCLowerBounds** method syntaxes have the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.
theMatrixBase0	A double precision array, dimensioned in the calling routine as **Dim theMatrixBase0() As Double** or as **Dim theMatrixBase0(,) As Double** On return from **GetBCLowerBounds** each element of theMatrixBase0 contains the lower bound on the bias-corrected confidence interval for the corresponding element of the matrix of estimates specified by matrixID. On return from **GetBCUpperbounds**, theMatrixBase0 contains upper bounds.
groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement[156]: [3].

Settings

The settings for *matrixID* are:

Constant	Value	Description

SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

In order to use **GetBCLowerBounds** (*matrixID*), you must first use NeedBCLowerBounds 320 (*matrixID*). For example, you have to use

object.NeedBCLowerBounds 320 (FactorScoreWeights)

before using

object.**GetBCLowerBounds** (FactorScoreWeights, ...)

Similarly, in order to use GetBCUpperBounds (*matrixID*), you must first use NeedBCUpperBounds 320 (*matrixID*). For example, you have to use

object.NeedBCUpperBounds 320 (FactorScoreWeights)

before using

object.**GetBCUpperBounds** (FactorScoreWeights, ...)

Bootstrap 188 can be used to specify the number of bootstrap samples. By default, 1000 bootstrap samples will be generated.

ConfidenceBC 207 can be used to specify the confidence level. By default, 90% confidence intervals will be estimated.

See Also

Bootstrap Method 188

ConfidenceBC Method 207

GetBootSampleEstimates Method 251

GetEstimates Method 260

GetPCLowerBounds, GetPCUpperBounds Methods 269

GetStandardErrors Method 278

NeedBCLowerBounds, NeedBCUpperBounds Methods 320

This example demonstrates the **GetBCLowerBounds** and **GetBCUpperBounds** methods.

```
Imports System.Diagnostics
Imports AmosEngineLib.AmosEngine.TMatrixID
Imports Microsoft.VisualBasic
Module MainModule
    ' GetBCLowerBounds, GetBCUpperBounds Methods Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.Bootstrap(2000)
        Sem.ConfidenceBC(90) '90% confidence intervals
        Sem.NeedBCLowerBounds(FactorScoreWeights)
        Sem.NeedBCUpperBounds(FactorScoreWeights)
        Sem.Standardized()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("wordmean = verbal + (1) err_w")

        Dim X(,) As Double
        Dim RNames() As String
        Dim CNames() As String

        'Get the row and column variable names
        Sem.RowNames(FactorScoreWeights, RNames)
        Sem.ColumnNames(FactorScoreWeights, CNames)

        'Print the lower bounds
        Sem.GetBCLowerBounds(FactorScoreWeights, X)
        Debug.WriteLine(vbCrLf & "Confidence intervals on factor score weights -- lower bound")
        PrintMatrix(X, CNames, RNames)

        'Print the upper bounds
        Sem.GetBCUpperBounds(FactorScoreWeights, X)
        Debug.WriteLine(vbCrLf & "Confidence intervals on factor score weights -- upper bound")
        PrintMatrix(X, CNames, RNames)

        Sem.Dispose()
    End Sub

    'Print a matrix in the debug window
    Sub PrintMatrix(ByVal TheMatrix(,) As Double, ByVal CNames$(), ByVal RNames$())
        Dim NRows1 As Integer, NColumns1 As Integer
        Dim i As Integer, j As Integer
        NRows1 = UBound(RNames)
        NColumns1 = UBound(CNames)

        Debug.Write(" ")
        For j = 0 To NColumns1
            Debug.Write(CNames(j).PadLeft(10))
        Next

        Debug.WriteLine("")
        For i = 0 To NRows1
            Debug.Write(RNames(i).PadRight(8))
            For j = 0 To NColumns1
                Debug.Write(TheMatrix(i, j).ToString(".00000").PadLeft(10))
            Next
```

```
        Debug.WriteLine("")
    Next
  End Sub
End Module
```

4.5.2.2.2.49 GetBCLowerBoundsEx, GetBCUpperBoundsEx Methods

Gets the lower and upper bounds of bias-corrected bootstrap confidence intervals for the elements of a matrix of estimates.

Syntax

object.**GetBCLowerBoundsEx** (*matrixID*, *am*)
object.**GetBCUpperBoundsEx** (*matrixID*, *am*, *groupNumber*)

The **GetBCLowerBoundsEx** and **GetBCLowerBoundsEx** method syntaxes have the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.
am	An object of type AmosMatrix 411. On return from **GetBCLowerBoundsEx** *am* contains lower bounds on confidence intervals for the estimates specified by matrixID. On return from **GetBCUpperBoundsEx**, *am* contains upper bounds.
groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement 156: [3].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.

ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

In order to use **GetBCLowerBoundsEx** (*matrixID*), you must first use NeedBCLowerBounds 320 (*matrixID*). For example, you have to use

```
object.NeedBCLowerBounds 320 (FactorScoreWeights)
```

before using

object.**GetBCLowerBoundsEx** (maFactorScoreWeights, …)

Similarly, in order to use **GetBCUpperBoundsEx** (*matrixID*), you must first use NeedBCUpperBounds[320] (*matrixID*). For example, you have to use

object.NeedBCUpperBounds[320] (FactorScoreWeights)

before using

object.**GetBCUpperBoundsEx** (FactorScoreWeights, …)

Bootstrap[188] can be used to specify the number of bootstrap samples. By default, 1000 bootstrap samples are generated.

ConfidenceBC[207] can be used to specify the confidence level. By default, 90% confidence intervals are estimated.

GetBCLowerBoundsEx and GetBCUpperBoundsEx differ from GetBCLowerBounds[243] and GetBCUpperBounds[243] in the following way. GetBCLowerBoundsEx and GetBCUpperBoundsEx assign values to the members of an AmosMatrix[411] object, which contains the matrix of lower bounds (or upper bounds) as well as the variable names and variable numbers associated with the matrix's rows and columns. GetBCLowerBounds[243] and GetBCUpperBounds[243], by contrast, merely set a double array equal to the matrix of lower bounds (or upper bounds). Additional calls to RowNames[382], RowNumbers[386], ColumnNames[199] and ColumnNumbers[203] are necessary if there is a need for the variable names and variable numbers associated with the matrix's rows and columns.

GetBCLowerBoundsEx and **GetBCUpperBoundsEx** are often more convenient, but GetBCLowerBounds[243] and GetBCUpperBounds[243] are faster.

This example demonstrates the **GetBCLowerBoundsEx** and **GetBCUpperBoundsEx** methods.

```
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    ' GetBCLowerBoundsEx, GetBCUpperBoundsEx Methods Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.Bootstrap(2000)
        Sem.ConfidenceBC(90) '90% confidence intervals
        Sem.NeedBCLowerBounds(FactorScoreWeights)
        Sem.NeedBCUpperBounds(FactorScoreWeights)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("wordmean = verbal + (1) err_w")

        Dim am As New AmosEngineLib.AmosMatrix
        Dim ad As New AmosDebug.AmosDebug

        Sem.GetBCLowerBoundsEx(FactorScoreWeights, am)
        ad.PrintX(am, "Confidence intervals on factor score weights -- lower bounds")

        Sem.GetBCUpperBoundsEx(FactorScoreWeights, am)
        ad.PrintX(am, "Confidence intervals on factor score weights -- upper bounds")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.50 GetBootSampleEstimates Method

Gets a matrix of estimates from an individual bootstrap sample.

Syntax

object.**GetBootSampleEstimates** (*matrixID*, *theMatrix*, *sampleNumber*)
object.**GetBootSampleEstimates** (*matrixID*, *theMatrix*, *sampleNumber*, *groupNumber*)

The **GetBootSampleEstimates** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.
theMatrix	A double precision array, dimensioned in the calling routine as **Dim theMatrix(,) As Double.**

	On return from **GetBootSampleEstimates** theMatrix contains the estimate from bootstrap sample number *sampleNumber*, of the matrix specified by matrixID.
sampleNumber	An integer specifying an individual bootstrap sample. The first bootstrap sample is sample number 1.
groupNumber	Optional integer specifying a group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement[156]: [3].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of

		residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

You have to use NeedBootSampleEstimates 325 to declare that estimates of a matrix will be needed before you can use **GetBootSampleEstimates** to obtain the estimates from a bootstrap sample. For example, you have to use

object.NeedBootSampleEstimates 325 (StandardizedTotalEffects)

before using

object.**GetBootSampleEstimates** (StandardizedTotalEffects, ...)

See Also

Bootstrap Method 188

GetBCLowerBounds, GetBCUpperBounds Methods 243

GetEstimates Method 260

GetPCLowerBounds, GetPCUpperBounds Methods 269

GetStandardErrors Method 278

NeedBootSampleEstimates Method 325

This example demonstrates the **GetBootSampleEstimates** method.

```
Imports System.Diagnostics
Imports AmosEngineLib.AmosEngine.TMatrixID
Imports Microsoft.VisualBasic
Module MainModule
    ' GetBootSampleEstimates Method Example
    Sub Main()
        Const NBootSamples As Integer = 3
        Dim i As Integer
        Dim X(,) As Double
        Dim RNames() As String
        Dim CNames() As String
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.NeedBootSampleEstimates(FactorScoreWeights)
        Sem.Bootstrap(NBootSamples)
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Row Names(FactorScoreWeights, RNames)
        Sem.ColumnNames(FactorScoreWeights, CNames)

        For i = 1 To NBootSamples
            Sem.GetBootSampleEstimates(FactorScoreWeights, X, i)
            Debug.WriteLine("")
            Debug.WriteLine("Factor score w eights from bootstrap sample #" & i)
            Debug.WriteLine("")
            PrintMatrix(X, CNames, RNames)
        Next

        Sem.Dispose()
    End Sub

    'Print a matrix in the debug window
    Sub PrintMatrix(ByVal TheMatrix(,) As Double, ByVal CNames$(), ByVal RNames$())
        Dim NRow s1 As Integer, NColumns1 As Integer
        Dim i As Integer, j As Integer
        NRow s1 = UBound(RNames)
        NColumns1 = UBound(CNames)

        Debug.Write(" ")
        For j = 0 To NColumns1
        Debug.Write(CNames(j).PadLeft(10))
        Next

        Debug.WriteLine("")
        For i = 0 To NRow s1
            Debug.Write(RNames(i).PadRight(8))
            For j = 0 To NColumns1
                Debug.Write(TheMatrix(i, j).ToString(".00000").PadLeft(10))
            Next
            Debug.WriteLine("")
        Next
    End Sub
End Module
```

4.5.2.2.2.51 GetDataFile Method

Gets information about the data file for a single group.

Syntax

object.**GetDataFile** (*groupNumber*, *dbFormat*, *fileName*, *tableName*, *groupingVariable*, *groupingValue*)

The **GetDataFile** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine** or **Pd**.
groupNumber	Number of the group for which data file information is wanted. The first group is group number 1.
dbFormat	(Integer) A database format specifier, as described in Settings.
fileName	(String) The name of the data file.
tableName	(String) The name of the data table within the data file (for data files that contain multiple data tables).
groupingVariable	(String) The name of the grouping variable. The empty string if there is no grouping variable.
groupingValue	(Object) The value of the grouping variable for this group.

Placement 156: [2].

Settings

The settings for *dbFormat* are:

Constant	Value	Description
mmDBASE3	0	Dbase III
mmDBASE4	1	Dbase IV
mmDBASE5	2	Dbase V
mmEXCEL3	3	Excel 3
mmEXCEL4	4	Excel 4

mmEXCEL5	5	Excel 5, Excel 7
mmEXCEL97	6	Excel 97, Excel 8
mmFOXPRO20	7	Foxpro 2.0
mmFOXPRO25	8	Foxpro 2.5
mmFOXPRO26	9	Foxpro 2.6
mmLOTUSWK1	11	Lotus *.wk1
mmLOTUSWK3	12	Lotus *.wk3
mmLOTUSWK4	13	Lotus *.wk4
mmAccess	14	Microsoft Access
mmSPSS	18	SPSS Statistics
mmText	19	Text

The following program specifies a two-group model and displays the data file, data table name, grouping variable name and grouping value for each group.

```
Imports System.Diagnostics
Module MainModule
   ' GetDataFile Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
      Sem.AStructure("academic <> attract")

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
      Sem.AStructure("academic <> attract")

      Dim dbformat As PXMLPersist.CDataTable.cDatabaseFormat
      Dim FileName As String
      Dim TableName As String
      Dim GroupingVariable As String
      Dim GroupingValue As Object
      Dim iGroup As Integer

      For iGroup = 1 To 2
         Debug.WriteLine("")
         Debug.WriteLine("Group: " & Sem.GetGroupName(iGroup))

         Sem.GetDataFile(iGroup, dbformat, FileName, TableName, GroupingVariable, GroupingValue)

         Debug.WriteLine("File Name = " & FileName)
         Debug.WriteLine("Table Name = " & TableName)
         Debug.WriteLine("Grouping Variable = " & GroupingVariable)
         Debug.WriteLine("Grouping Value = " & CStr(GroupingValue))
      Next

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.52 GetEstimate Method

Gets one element of a matrix of estimates.

Syntax

result = *object*.**GetEstimate** (*matrixID*, *rowVariableName*, *columnVariableName*)
result = *object*.**GetEstimate** (*matrixID*, *rowVariableName*, *columnVariableName*, *groupNumber*)

result = *object*.**GetEstimate** (*matrixID*, *row*, *column*)
result = *object*.**GetEstimate** (*matrixID*, *row*, *column*, *groupNumber*)

The **GetEstimates** method syntax has the following parts:

Part	Description
result	(Double) The scalar value in the specified row and column of the matrix.

object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.
rowVariableName	*The name of a variable that labels a row of the matrix.*
columnVariableName	*The name of a variable that labels a row of the matrix.*
row	*(Integer) A row number. The rows are numbered starting with 0.*
column	*(Integer) A column number. The columns are numbered starting with 0.*
groupNumber	Optional integer specifying a group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement 156: [3].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.

AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

You have to use NeedEstimates 329 to declare that estimates of a matrix will be needed before you can use **GetEstimate** to obtain the estimates. For example, you have to use

object.NeedEstimates 329 (StandardizedTotalEffects)

before using

object.**GetEstimate** (StandardizedTotalEffects, ...)

See Also

GetEstimates Method 260

GetEstimatesEx Method 265

GetStandardErrors Method 278

NeedEstimates Method 329

4.5.2.2.2.53 GetEstimates Method

Gets a matrix of estimates.

Syntax

object.**GetEstimates** (*matrixID*, *theMatrixBase0*)
object.**GetEstimates** (*matrixID*, *theMatrixBase0*, *groupNumber*)

The **GetEstimates** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.
theMatrixBase0	A double precision array, dimensioned in the calling routine as **Dim theMatrixBase0() As Double** or as **Dim theMatrixBase0(,) As Double** On return from **GetEstimates** theMatrixBase0 contains the estimate of the matrix specified by matrixID.
groupNumber	Optional integer specifying a group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement 156: [3].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.

ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

You have to use NeedEstimates 329 to declare that estimates of a matrix will be needed before you can use **GetEstimates** to obtain the estimates. For example, you have to use

object.NeedEstimates 329 (StandardizedTotalEffects)

before using

object.**GetEstimates** (StandardizedTotalEffects, ...)

See Also

GetEstimate Method 258

GetEstimatesEx Method 265

GetStandardErrors Method 278

NeedEstimates Method 329

The following program fits Models A and B of Example 11. It displays the matrix of total effects for each group and model.

```
Imports System.Diagnostics
Imports AmosEngineLib.AmosEngine.TMatrixID
Imports Microsoft.VisualBasic
Module MainModule
    ' GetEstimates Method Example
    Sub Main()
        Dim CNames() As String, RNames() As String, X(,) As Double
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.NeedEstimates(TotalEffects)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
        Sem.GroupName("girls")
        Sem.AStructure("academic = (g1) GPA + (g2) attract + (1) e1")
        Sem.AStructure( "attract = (g3) height + (g4) w eight + (g5) rating + (g6) academic + (1) e2")
        Sem.AStructure("e2 <--> e1")

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
        Sem.GroupName("boys")
        Sem.AStructure("academic = (b1) GPA + (b2) attract + (1) e1")
        Sem.AStructure( "attract = (b3) height + (b4) w eight + (b5) rating + (b6) academic + (1) e2 ")
        Sem.AStructure("e2 <--> e1")

        Sem.Model("Model_A")
        Sem.Model("Model_B", "g1=b1", "g2=b2", "g3=b3", "g4=b4", "g5=b5", "g6=b6")

        'Print implied covariances for each model and each group
        Dim ModelNumber As Integer
        Dim GroupNumber As Integer
        For ModelNumber = 1 To 2
            Sem.FitModel(ModelNumber)
            For GroupNumber = 1 To 2

                Sem.GetEstimates(TotalEffects, X, GroupNumber)

                Sem.ColumnNames(TotalEffects, CNames, GroupNumber)
                Sem.Row Names(TotalEffects, RNames, GroupNumber)
                Debug.WriteLine(vbCrLf & "Group " & GroupNumber & ", Model " & ModelNumber)
                PrintMatrix(X, CNames, RNames)
            Next
        Next

        Sem.Dispose()
    End Sub
    'Print a matrix in the debug w indow
    Sub PrintMatrix(ByVal TheMatrix(,) As Double, ByVal CNames$(), ByVal RNames$())
        Dim NRow s1 As Integer, NColumns1 As Integer
        Dim i As Integer, j As Integer
        NRow s1 = UBound(RNames)
        NColumns1 = UBound(CNames)

        Debug.Write(" ")
        For j = 0 To NColumns1
            Debug.Write(CNames(j).PadLeft(10))
        Next
        Debug.WriteLine("")

        For i = 0 To NRow s1
            Debug.Write(RNames(i).PadRight(8))
            For j = 0 To NColumns1
```

```
            Debug.Write(TheMatrix(i, j).ToString(".00000").PadLeft(10))
        Next
        Debug.WriteLine("")
    Next
  End Sub
End Module
```

4.5.2.2.2.54 GetEstimatesEx Method

Gets a matrix of estimates.

Syntax

object.**GetEstimatesEx** (*matrixID*, *theMatrix*)
object.**GetEstimatesEx** (*matrixID*, *theMatrix*, *groupNumber*)

The **GetEstimatesEx** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.
theMatrix	An object of type AmosMatrix 411. On return from **GetEstimatesEx** *theMatrix* contains the estimate of the matrix specified by *matrixID*.
groupNumber	Optional integer specifying a group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement 156: [3].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.

ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

You have to use NeedEstimates 329 to declare that estimates of a matrix will be needed before you can use **GetEstimatesEx** to obtain the estimates. For example, you have to use

object.NeedEstimates 329 (StandardizedTotalEffects)

before using

object.**GetEstimatesEx** (StandardizedTotalEffects, …)

GetEstimatesEx differs from GetEstimates[260] in the following way. GetEstimatesEx assigns values to the members of an AmosMatrix[411] object, which contains the matrix of estimates as well as the variable names and variable numbers associated with the matrix's rows and columns. GetEstimates[260], by contrast, merely sets a double array equal to the matrix of estimates. Additional calls to RowNames[382], RowNumbers[386], ColumnNames[199] and ColumnNumbers[203] are necessary if there is a need for the variable names and variable numbers associated with the matrix's rows and columns.

GetEstimatesEx is often more convenient, but GetEstimates[260] is faster.

See Also

GetEstimates Method[260]

GetStandardErrors Method[278]

NeedEstimates Method[329]

The following program fits Models A and B of Example 11. It displays the matrix of total effects for each group and model.

```
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    ' GetEstimatesEx Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Dim am As New AmosEngineLib.AmosMatrix
        Dim ad As New AmosDebug.AmosDebug
        ad.DecimalPlaces = 4

        Sem.NeedEstimates(TotalEffects)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
        Sem.GroupName("girls")
        Sem.AStructure("academic = (g1) GPA + (g2) attract + (1) e1")
        Sem.AStructure("attract = (g3) height + (g4) w eight + (g5) rating + (g6) academic + (1) e2")
        Sem.AStructure("e2 <--> e1")

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
        Sem.GroupName("boys")
        Sem.AStructure("academic = (b1) GPA + (b2) attract + (1) e1")
        Sem.AStructure("attract = (b3) height + (b4) w eight + (b5) rating + (b6) academic + (1) e2")
        Sem.AStructure("e2 <--> e1")

        Sem.Model("Model_A")
        Sem.Model("Model_B", "g1=b1", "g2=b2", "g3=b3", "g4=b4", "g5=b5", "g6=b6")

        'Print implied covariances for each model and each group
        Dim ModelNumber As Integer, GroupNumber As Integer
        For ModelNumber = 1 To 2
            Sem.FitModel(ModelNumber)
            For GroupNumber = 1 To 2
                Sem.GetEstimatesEx(TotalEffects, am, GroupNumber)
                ad.PrintX(am, "Model " & ModelNumber & ", Group " & GroupNumber)
            Next
        Next

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.55 GetGroupName Method

Gets the name of a group.

Syntax

object.**GetGroupName** ()
object.**GetGroupName** (*groupNumber*)

The **GetGroupName** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

groupNumber	Optional group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement[156]: [2] [3].

In the following program, the **GetGroupName** method is used to display the group names in a two-group analysis.

```
Imports System.Diagnostics
Module MainModule
   ' GetGroupName Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine

      Sem.TextOutput()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\Fels_fem.sav")
      Sem.GroupName("girls")
      Sem.AStructure("academic = GPA + attract + error1 (1)")
      Sem.AStructure("attract = height + w eight + rating + academic + error2 (1)")
      Sem.AStructure("error2 <--> error1")

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\Fels_mal.sav")
      Sem.GroupName("boys")
      Sem.AStructure("academic = GPA + attract + error1 (1)")
      Sem.AStructure("attract = height + w eight + rating + academic + error2 (1)")
      Sem.AStructure("error2 <--> error1")

      Dim i As Integer
      For i = 1 To Sem.NumberOfGroups
         Debug.WriteLine("Group " & i & " is called " & Sem.GetGroupName(i))
      Next

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.56 GetPCLow erBounds, GetPCUpperBounds Methods

Gets the lower and upper bounds of the bootstrap confidence intervals for the elements of a matrix of estimates, using the percentile method.

Syntax

object.**GetPCLowerBounds** (*matrixID*, *theMatrixBase0*)
object.**GetPCLowerBounds** (*matrixID*, *theMatrixBase0*, *groupNumber*)

object.**GetPCUpperBounds** (*matrixID*, *theMatrixBase0*)
object.**GetPCUpperBounds** (*matrixID*, *theMatrixBase0*, *groupNumber*)

The **GetPCLowerBounds** and **GetPCLowerBounds** method syntaxes have the following parts:

Part	Description

object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.
theMatrixBase0	A double precision array, dimensioned in the calling routine as **Dim theMatrixBase0() As Double** or as **Dim theMatrixBase0(,) As Double** On return from **GetPCLowerBounds** each element of *theMatrixBase0* contains the lower bound on the confidence interval for the corresponding element of the matrix of estimates specified by *matrixID*. On return from **GetPCUpperbounds**, *theMatrixBase0* contains upper bounds.
groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement 156: [3].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.

AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

In order to use **GetPCLowerBounds** (*matrixID*), you must first use NeedPCLowerBounds 332 (*matrixID*). For example, you have to use

object.NeedPCLowerBounds 332 (FactorScoreWeights)

before using

object.**GetPCLowerBounds** (FactorScoreWeights, ...)

Similarly, in order to use **GetPCUpperBounds** (*matrixID*), you must first use NeedPCUpperBounds 332 (*matrixID*). For example, you have to use

object.NeedPCUpperBounds 332 (FactorScoreWeights)

before using

object.**GetPCUpperBounds** (FactorScoreWeights, ...)

Bootstrap 188 can be used to specify the number of bootstrap samples. By default, 1000 bootstrap samples will be generated.

ConfidencePC [209] can be used to specify the confidence level. By default, 90% confidence intervals will be estimated.

This example demonstrates the **GetPCLowerBounds** and **GetPCUpperBounds** methods.

```
Imports System.Diagnostics
Imports AmosEngineLib.AmosEngine.TMatrixID
Imports Microsoft.VisualBasic
Module MainModule
    ' GetPCLowerBounds, GetPCUpperBounds Methods Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.Bootstrap(2000)
        Sem.ConfidencePC(90) '90% confidence intervals

        Sem.NeedPCLowerBounds(FactorScoreWeights)
        Sem.NeedPCUpperBounds(FactorScoreWeights)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("wordmean = verbal + (1) err_w")

        Dim X(,) As Double
        Dim RNames() As String
        Dim CNames() As String

        'Get the row and column variable names
        Sem.RowNames(FactorScoreWeights, RNames)
        Sem.ColumnNames(FactorScoreWeights, CNames)

        'Print the lower bounds
        Sem.GetPCLowerBounds(FactorScoreWeights, X)
        Debug.WriteLine(vbCrLf & "Confidence intervals on factor score weights -- lower bound")
        PrintMatrix(X, CNames, RNames)

        'Print the upper bounds
        Sem.GetPCUpperBounds(FactorScoreWeights, X)
        Debug.WriteLine(vbCrLf & "Confidence intervals on factor score weights -- upper bound")
        PrintMatrix(X, CNames, RNames)

        Sem.Dispose()
    End Sub

    'Print a matrix in the debug window
    Sub PrintMatrix(ByVal TheMatrix(,) As Double, ByVal CNames$(), ByVal RNames$())
        Dim NRows1 As Integer, NColumns1 As Integer
        Dim i As Integer, j As Integer
        NRows1 = UBound(RNames)
        NColumns1 = UBound(CNames)

        Debug.Write(" ")
        For j = 0 To NColumns1
            Debug.Write(CNames(j).PadLeft(10))
        Next

        Debug.WriteLine("")
        For i = 0 To NRows1
            Debug.Write(RNames(i).PadRight(8))
            For j = 0 To NColumns1
                Debug.Write(TheMatrix(i, j).ToString(".00000").PadLeft(10))
            Next
```

```
        Debug.WriteLine("")
    Next
  End Sub
End Module
```

4.5.2.2.2.57 GetPCLowerBoundsEx, GetPCUpperBoundsEx Methods

Gets the lower and upper bounds of bootstrap confidence intervals for the elements of a matrix of estimates, using the percentile method.

Syntax

object.**GetPCLowerBoundsEx** (*matrixID*, *am*)
object.**GetPCLowerBoundsEx** (*matrixID*, *am*, *groupNumber*)

object.**GetPCUpperBoundsEx** (*matrixID*, *am*)
object.**GetPCUpperBoundsEx** (*matrixID*, *am*, *groupNumber*)

The **GetPCLowerBoundsEx** and **GetPCLowerBoundsEx** method syntaxes have the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.
am	An object of type AmosMatrix 411. On return from **GetPCLowerBoundsEx** *am* contains lower bounds on confidence intervals for the estimates specified by *matrixID*. On return from **GetPCUpperBoundsEx**, *am* contains upper bounds.
groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement 156: [3].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.

SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

In order to use **GetPCLowerBoundsEx** (*matrixID*), you must first use NeedPCLowerBounds 332 (*matrixID*). For example, you have to use

```
object.NeedPCLowerBounds 332 (FactorScoreWeights)
```

before using

object.**GetPCLowerBoundsEx** (FactorScoreWeights, …)

Similarly, in order to use **GetPCUpperBoundsEx** (*matrixID*), you must first use NeedPCUpperBounds[332] (*matrixID*). For example, you have to use

object.NeedPCUpperBounds[332] (FactorScoreWeights)

before using

object.**GetPCUpperBoundsEx** (FactorScoreWeights, …)

Bootstrap[188] can be used to specify the number of bootstrap samples. By default, 1000 bootstrap samples are generated.

ConfidencePC[209] can be used to specify the confidence level. By default, 90% confidence intervals are estimated.

GetPCLowerBoundsEx and **GetPCUpperBoundsEx** differ from GetPCLowerBounds[269] and GetPCUpperBounds[269] in the following way. **GetPCLowerBoundsEx** and **GetPCUpperBoundsEx** assign values to the members of an AmosMatrix[411] object, which contains the matrix of lower bounds (or upper bounds) as well as the variable names and variable numbers associated with the matrix's rows and columns. GetPCLowerBounds[269] and GetPCUpperBounds[269] , by contrast, merely set a double array equal to the matrix of lower bounds (or upper bounds). Additional calls to RowNames[382], RowNumbers[386], ColumnNames[199] and ColumnNumbers[203] are necessary if there is a need for the variable names and variable numbers associated with the matrix's rows and columns.

GetPCLowerBoundsEx and **GetPCUpperBoundsEx** are often more convenient, but GetPCLowerBounds[269] and GetPCUpperBounds[269] are faster.

This example demonstrates the **GetPCLowerBoundsEx** and **GetPCUpperBoundsEx** methods.

```
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    ' GetPCLowerBoundsEx, GetPCUpperBoundsEx Methods Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.Bootstrap(2000)
        Sem.ConfidencePC(90) '90% confidence intervals

        Sem.NeedPCLowerBounds(FactorScoreWeights)
        Sem.NeedPCUpperBounds(FactorScoreWeights)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("wordmean = verbal + (1) err_w")

        Dim am As New AmosEngineLib.AmosMatrix
        Dim ad As New AmosDebug.AmosDebug

        Sem.GetPCLowerBoundsEx(FactorScoreWeights, am)
        ad.PrintX(am, "Confidence intervals on factor score weights -- lower bounds")

        Sem.GetPCUpperBoundsEx(FactorScoreWeights, am)
        ad.PrintX(am, "Confidence intervals on factor score weights -- upper bounds"
)
        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.58 GetStandardErrors Method

Gets bootstrap standard errors for the elements of a matrix of estimates.

Syntax

object.**GetStandardErrors** (*matrixID*, *theMatrixBase0*)
object.**GetStandardErrors** (*matrixID*, *theMatrixBase0*, *groupNumber*)

The **GetStandardErrors** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.
theMatrixBase0	A double precision array, dimensioned in the calling routine as

	Dim theMatrixBase0() As Double or as **Dim theMatrixBase0(,) As Double** On return from **GetStandardErrors** *theMatrixBase0* contains bootstrap standard errors for the elements of the matrix of estimates specified by *matrixID*.
groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement[156]: [3].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of

		residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

You have to use NeedStandardErrors 336 to declare that standard errors for a matrix will be needed before you can use **GetStandardErrors** to obtain the standard errors. For example, you have to use

object.NeedStandardErrors 336 (StandardizedTotalEffects)

before using

object.**GetStandardErrors** (StandardizedTotalEffects)

See Also

Bootstrap Method 188

GetEstimates Method 260

GetStandardErrorsEx Method 282

NeedStandardErrors Method 336

This example demonstrates the **GetStandardErrors** method.

```
Imports System.Diagnostics
Imports AmosEngineLib.AmosEngine.TMatrixID
Imports Microsoft.VisualBasic
Module MainModule
   ' GetStandardErrors Method Example
   Sub Main()
      Dim Sem As New AmosEngineLib.AmosEngine

      Sem.NeedStandardErrors(FactorScoreWeights)

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.AStructure("w ordmean = verbal + (1) err_w ")

      Dim X(,) As Double
      Dim RNames() As String
      Dim CNames() As String

      'Get the row and column variable names
      Sem.RowNames(FactorScoreWeights, RNames)
      Sem.ColumnNames(FactorScoreWeights, CNames)

      'Print the standard errors
      Sem.GetStandardErrors(FactorScoreWeights, X)
      Debug.WriteLine(vbCrLf & "Standard errors for factor score w eights")
      PrintMatrix(X, CNames, RNames)

      Sem.Dispose()
   End Sub

   'Print a matrix in the debug w indow
   Sub PrintMatrix(ByVal TheMatrix(,) As Double, ByVal CNames$(), ByVal RNames$())
      Dim NRow s1 As Integer, NColumns1 As Integer
      Dim i As Integer, j As Integer
      NRow s1 = UBound(RNames)
      NColumns1 = UBound(CNames)

      Debug.Write(" ")
      For j = 0 To NColumns1
         Debug.Write(CNames(j).PadLeft(10))
      Next

      Debug.WriteLine("")
      For i = 0 To NRow s1
         Debug.Write(RNames(i).PadRight(8))
         For j = 0 To NColumns1
            Debug.Write(TheMatrix(i, j).ToString(".00000").PadLeft(10))
         Next
         Debug.WriteLine("")
      Next
   End Sub
End Module
```

4.5.2.2.2.59 GetStandardErrorsEx Method

Gets bootstrap standard errors for the elements of a matrix of estimates.

Syntax

object.**GetStandardErrorsEx** (*matrixID*, *am*, *groupNumber*)

The **GetStandardErrorsEx** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.
am	An object of type AmosMatrix 411. On return from **GetStandardErrorsEx**, *am* contains bootstrap standard errors for the elements of the matrix of estimates specified by *matrixID*.
groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement 156: [3].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.

ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

You have to use NeedStandardErrors 336 to declare that standard errors for a matrix will be needed before you can use **GetStandardErrorsEx** to obtain the standard errors. For example, you have to use

object.NeedStandardErrors 336 (StandardizedTotalEffects)

before using

object.**GetStandardErrorsEx** (StandardizedTotalEffects, ...)

GetStandardErrorsEx differs from GetStandardErrors 278 in the following way. GetStandardErrorsEx assigns values to the members of an AmosMatrix 411 object, which contains the matrix of standard errors as well as the variable names and variable numbers associated with the matrix's rows and columns. GetStandardErrors 278 , by contrast, merely sets a double array equal to the matrix of standard errors. Additional calls to RowNames 382, RowNumbers 386, ColumnNames 199 and ColumnNumbers 203 are

necessary if there is a need for the variable names and variable numbers associated with the matrix's rows and columns.

GetStandardErrorsEx is often more convenient, but GetStandardErrors 278 is faster.

See Also

Bootstrap Method 188

GetEstimates Method 260

GetStandardErrors Method 278

NeedStandardErrors Method 336

This example demonstrates the **GetStandardErrorsEx** method.

```
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    ' GetStandardErrorsEx Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.NeedStandardErrors(FactorScoreWeights)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Dim am As New AmosEngineLib.AmosMatrix
        Dim ad As New AmosDebug.AmosDebug
        ad.DecimalPlaces = 5
        Sem.GetStandardErrorsEx(FactorScoreWeights, am)
        ad.PrintX(am, "Standard errors for factor score w eights")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.60 Gls Method

Requests a generalized least squares solution, obtained by minimizing (D1) together with (D3) in Appendix B.

Syntax

object.**Gls** ()

The **Gls** method syntax has the following parts:

Part	Description

object	An object of type **AmosEngine**.

Placement[156]: [1].

Default

When you do not specify an estimation criterion, the maximum likelihood criterion (Ml[305] method) is used.

See Also

Adf Method[163]

BootGls Method[183]

Ml Method[305]

Sls Method[392]

Uls Method[403]

This example demonstrates the **Gls** method.

```
Module MainModule
    ' Gls Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine

        Sem.Gls()

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.61 GroupName Method

Assigns a name to a group.

Syntax

object.**GroupName** (*theGroupName*)

The **GroupName** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
theGroupName	A name for the group whose data is specified by the most recent BeginGroup 173 or BeginGroupEx 175 method.

Placement 156: [2].

Default

The groups are called 'Group number 1', 'Group number 2', and so on.

See Also

BeginGroup Method 173

BeginGroupEx Method 175

This example demonstrates the **GroupName** method.

```
Module MainModule
    ' GroupName Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
        Sem.GroupName("girls")
        Sem.AStructure("academic = GPA + attract + e1 (1)")
        Sem.AStructure("attract = height + weight + rating + academic + e2 (1)")
        Sem.AStructure("e2 <--> e1")

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
        Sem.GroupName("boys")
        Sem.AStructure("academic = GPA + attract + e1 (1)")
        Sem.AStructure("attract = height + weight + rating + academic + e2 (1)")
        Sem.AStructure("e2 <--> e1")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.62 ImpliedMoments Method

Controls reporting of the implied covariance matrix for the observed variables. When means and intercepts are explicitly modeled, **ImpliedMoments** also controls the reporting of implied means.

Syntax

object.**ImpliedMoments** ()
object.**ImpliedMoments** (*tf*)

The **ImpliedMoments** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. A boolean value that controls the reporting of implied moments. True (default) requests the output. False suppresses it.

Placement 156: [1].

Default

Implied moments are not reported.

Remarks

The 'implied' variances, covariances and means are estimates of the corresponding population values under the assumption that the specified model is correct.

If you use both the Standardized 396 and the **ImpliedMoments** methods, the implied correlation matrix will be reported, in addition to the implied covariance matrix.

ImpliedMoments is identical to AllImpliedMoments 165, except that AllImpliedMoments 165 displays implied variances, covariances and means for all variables in the model, not just for the observed variables.

See Also

Admissible Method 164

AllImpliedMoments Method 165

ResidualMoments Method 377

SampleMoments Method 389

This example demonstrates the **ImpliedMoments** method.

```
Module MainModule
    ' ImpliedMoments Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.ImpliedMoments()
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.63 Initialize Method (AmosEngine)

Initializes the Amos engine and specifies file names and locations for default output.

Syntax

object.**Initialize** (*projectName*)

The **Initialize** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
projectName	A fully qualified path without any file extension. The output generated by the TextOutput 398 method is written to the file *projectName*.AmosOutput. The output file used for displaying results on the path diagram in Amos Graphics is called *projectName*.amp.

Placement 156: Before any other **AmosEngine** methods or properties.

Default

The use of **Initialize** is optional. By default, the output generated by TextOutput 398 is written to the file *temppath***AmosScratch.AmosOutput**, where *temppath* is the Windows system temporary directory. Results to be displayed on the path diagram in Amos Graphics are written to the file *temppath***AmosScratch.amp**.

Remarks

The TextOutputFileName 399 method returns the name of the file that contains the output from TextOutput 398.

See Also

Shutdown Method 391

In the following program, the text output that is displayed by TextOutput 398 is written to the file **c:\AnAmosExample.AmosOutput**.

```
Module MainModule
   ' Initialize Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine

      Sem.Initialize("c:\AnAmosExample")
      Sem.TextOutput()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.AStructure("w ordmean = verbal + (1) err_w ")

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.64 InputMLMoments Method

Specifies that any sample covariances that are read from a data file are (biased) maximum likelihood estimates of the corresponding population covariances. In other words, the input covariance matrix is assumed to be made up of sums of squares and cross products, divided by **N** (rather than by **N - 1**).

Syntax

object.**InputMLMoments** ()

The **InputMLMoments** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement 156: [1].

Default

Sample covariances that are read from a data file are assumed to be (biased) maximum likelihood estimates.

Remarks

FitMLMoments [235] and **InputMLMoments** have different effects. **InputMLMoments** specifies that any sample covariance matrix that is read from a data file is a maximum likelihood estimate. **FitMLMoments** [235], on the other hand, tells Amos to *fit the model* to the sample covariance matrix ($\mathbf{S}^{(\varepsilon)}$ in Appendices A and B) that is the maximum likelihood estimate.

See Also

FitMLMoments Method [235]

FitUnbiasedMoments Method [239]

InputUnbiasedMoments Method [291]

This example demonstrates the **InputMLMoments** method.

```
Module MainModule
    ' InputMLMoments Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.InputMLMoments()
        Sem.FitMLMoments()

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Wheaton")
        Sem.AStructure("anomia67 <--- 67_alienation (1)")
        Sem.AStructure("anomia67 <--- eps1 (1)")
        Sem.AStructure("pow les67 <--- 67_alienation (path_p)")
        Sem.AStructure("pow les67 <--- eps2 (1)")
        Sem.AStructure("anomia71 <--- 71_alienation (1)")
        Sem.AStructure("anomia71 <--- eps3 (1)")
        Sem.AStructure("pow les71 <--- 71_alienation (path_p)")
        Sem.AStructure("pow les71 <--- eps4 (1)")
        Sem.AStructure("67_alienation <--- ses")
        Sem.AStructure("67_alienation <--- zeta1 (1)")
        Sem.AStructure("71_alienation <--- 67_alienation")
        Sem.AStructure("71_alienation <--- ses")
        Sem.AStructure("71_alienation <--- zeta2 (1)")
        Sem.AStructure("education <--- ses (1)")
        Sem.AStructure("education <--- delta1 (1)")
        Sem.AStructure("SEI <--- ses")
        Sem.AStructure("SEI <--- delta2 (1)")
        Sem.AStructure("eps3 <--> eps1")
        Sem.AStructure("eps1 (var_a)")
        Sem.AStructure("eps2 (var_p)")
        Sem.AStructure("eps3 (var_a)")
        Sem.AStructure("eps4 (var_p)")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.65 InputUnbiasedMoments Method

Specifies that any sample covariances that are read from a data file are unbiased estimates of the corresponding population covariances. In other words, the input covariance matrix is assumed to be made up of sums of squares and cross products, divided by **N - 1** (rather than by **N**).

Syntax

object.**InputUnbiasedMoments** ()

The **InputUnbiasedMoments** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement[156]: [1].

Default

Sample covariances read from a data file are assumed to be (biased) maximum likelihood estimates.

Remarks

FitUnbiasedMoments[239] and InputUnbiasedMoments have different effects. InputUnbiasedMoments specifies that any sample covariance matrix that is read from a data file is an unbiased estimate. **FitUnbiasedMoments**[239], on the other hand, tells Amos to *fit the model* to the sample covariance matrix ($\mathbf{S}^{(g)}$ in Appendices A and B) that is an unbiased estimate.

See Also

FitMLMoments Method[235]

FitUnbiasedMoments Method[239]

InputMLMoments Method[289]

This example demonstrates the **InputUnbiasedMoments** method.

```
Module MainModule
    ' InputUnbiasedMoments Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.InputUnbiasedMoments()
        Sem.FitUnbiasedMoments()

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Wheaton")
        Sem.AStructure("anomia67 <--- 67_alienation (1)")
        Sem.AStructure("anomia67 <--- eps1 (1)")
        Sem.AStructure("powles67 <--- 67_alienation (path_p)")
        Sem.AStructure("powles67 <--- eps2 (1)")
        Sem.AStructure("anomia71 <--- 71_alienation (1)")
        Sem.AStructure("anomia71 <--- eps3 (1)")
        Sem.AStructure("powles71 <--- 71_alienation (path_p)")
        Sem.AStructure("powles71 <--- eps4 (1)")
        Sem.AStructure("67_alienation <--- ses")
        Sem.AStructure("67_alienation <--- zeta1 (1)")
        Sem.AStructure("71_alienation <--- 67_alienation")
        Sem.AStructure("71_alienation <--- ses")
        Sem.AStructure("71_alienation <--- zeta2 (1)")
        Sem.AStructure("education <--- ses (1)")
        Sem.AStructure("education <--- delta1 (1)")
        Sem.AStructure("SEI <--- ses")
        Sem.AStructure("SEI <--- delta2 (1)")
        Sem.AStructure("eps3 <--> eps1")
        Sem.AStructure("eps1 (var_a)")
        Sem.AStructure("eps2 (var_p)")
        Sem.AStructure("eps3 (var_a)")
        Sem.AStructure("eps4 (var_p)")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.66 InputVariableHasMissingValues Method

Gets a boolean value that indicates whether an observed variable has missing observations.

Syntax

object.**InputVariableHasMissingValues** (*variableNumber*)
object.**InputVariableHasMissingValues** (*variableNumber, groupNumber*)

The **InputVariableHasMissingValues** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

variableNumber	An integer specifying a variable in the data set for group number *groupNumber*. *variableNumber* is 1 for the first variable in the data set.
groupNumber	Optional. An integer specifying a group. *groupNumber* is 1 for the first group. If *groupNumber* is omitted, the first group is used.

Placement[156]: [2].

See Also

AnyMissingValues Method[168]

The following program fits the model from Example 8 to the **Grant** data, and then displays in the debug window some characteristics of the variables in the dataset.

```
Imports System.Diagnostics
Module MainModule
  ' InputVariableHasMissingValues Method Example
  Sub Main()
    Dim i As Integer
    Dim Sem As New AmosEngineLib.AmosEngine

    Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grant")
    Sem.AStructure("visperc = (1) spatial + (1) err_v")
    Sem.AStructure("cubes = spatial + (1) err_c")
    Sem.AStructure("lozenges = spatial + (1) err_l")
    Sem.AStructure("paragraph = (1) verbal + (1) err_p")
    Sem.AStructure("sentence = verbal + (1) err_s")
    Sem.AStructure("w ordmean = verbal + (1) err_w ")

    Debug.WriteLine("Number of variables: " & Sem.DataFileNVariables)
    For i = 1 To Sem.DataFileNVariables
      Debug.WriteLine("")
      Debug.WriteLine("Variable number " & i)
      Debug.WriteLine("Name: " & Sem.InputVariableName(i))
      Debug.WriteLine("Label: " & Sem.InputVariableLabel(i))

      If Sem.InputVariableIsNumeric(i) Then
        Debug.WriteLine("Numeric")
      Else
        Debug.WriteLine("Non-numeric")
      End If

      If Sem.InputVariableHasMissingValues(i) Then
        Debug.WriteLine("Has missing values")
      Else
        Debug.WriteLine("Doesn't have missing values")
      End If
    Next

    Sem.Dispose()
  End Sub
End Module
```

4.5.2.2.2.67 InputVariableIsNumeric Method

Gets a boolean value that indicates whether an observed variable is numeric.

Syntax

object.**InputVariableIsNumeric**(*variableNumber*)
object.**InputVariableIsNumeric**(*variableNumber*, *groupNumber*)

The **InputVariableIsNumeric** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
variableNumber	An integer specifying a variable in the data set for group number *groupNumber*. *variableNumber* is 1 for the first variable in the data set.
groupNumber	Optional. An integer specifying a group. *groupNumber* is 1 for the first group. If *groupNumber* is omitted, the first group is used.

Placement 156: [2].

The following program fits the model from Example 8 to the **Grant** data, and then displays in the debug window some characteristics of the variables in the dataset.

```
Imports System.Diagnostics
Module MainModule
    ' InputVariableIsNumeric Method Example
    Sub Main()
        Dim i As Integer
        Dim Sem As New  AmosEngineLib.AmosEngine

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grant")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Debug.WriteLine("Number of variables: " & Sem.DataFileNVariables)
        For i = 1 To Sem.DataFileNVariables
            Debug.WriteLine("")
            Debug.WriteLine("Variable number " & i)
            Debug.WriteLine("Name: " & Sem.InputVariableName(i))
            Debug.WriteLine("Label: " & Sem.InputVariableLabel(i))

            If Sem.InputVariableIsNumeric(i) Then
                Debug.WriteLine("Numeric")
            Else
                Debug.WriteLine("Non-numeric")
            End If

            If Sem.InputVariableHasMissingValues(i) Then
                Debug.WriteLine("Has missing values")
            Else
                Debug.WriteLine("Doesn't have missing values")
            End If
        Next

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.68 InputVariableLabel Method

Gets the label (not the name) of an observed variable.

Syntax

result = *object*.**InputVariableLabel**(*variableNumber*)
result = *object*.**InputVariableLabel**(*variableNumber*, *groupNumber*)

The **InputVariableLabel** method syntax has the following parts:

Part	Description
result	*A variable label.*
object	An object of type **AmosEngine**.

variableNumber	An integer specifying a variable in the data set for group number *groupNumber*. *variableNumber* is 1 for the first variable in the data set.
groupNumber	Optional. An integer specifying a group. *groupNumber* is 1 for the first group. If *groupNumber* is omitted, the first group is used.

Placement[156]: [2].

Remarks

Variable labels are only available for SPSS Statistics data files. For other data formats, **InputVariableLabel** returns an empty string.

See Also

InputVariableName Method[297]

The following program fits the model from Example 8 to the **Grant** data, and then displays in the debug window some characteristics of the variables in the dataset.

```
Imports System.Diagnostics
Module MainModule
   ' InputVariableLabel Method Example
   Sub Main()
      Dim i As Integer
      Dim Sem As New  AmosEngineLib.AmosEngine

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\Grant.sav")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.AStructure("w ordmean = verbal + (1) err_w ")

      Debug.WriteLine("Number of variables: " & Sem.DataFileNVariables)
      For i = 1 To Sem.DataFileNVariables
         Debug.WriteLine("")
         Debug.WriteLine("Variable number " & i)
         Debug.WriteLine("Name: " & Sem.InputVariableName(i))
         Debug.WriteLine("Label: " & Sem.InputVariableLabel(i))

         If Sem.InputVariableIsNumeric(i) Then
            Debug.WriteLine("Numeric")
         Else
            Debug.WriteLine("Non-numeric")
         End If

         If Sem.InputVariableHasMissingValues(i) Then
            Debug.WriteLine("Has missing values")
         Else
            Debug.WriteLine("Doesn't have missing values")
         End If
      Next

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.69 InputVariableName Method

Gets the name of an observed variable.

Syntax

result = *object*.**InputVariableName**(*variableNumber*)
result = *object*.**InputVariableName**(*variableNumber*, *groupNumber*)

The **InputVariableName** method syntax has the following parts:

Part	Description
result	A variable name.
object	An object of type **AmosEngine**.

variableNumber	An integer specifying a variable in the data set for group number *groupNumber*. *variableNumber* is 1 for the first variable in the data set.
groupNumber	Optional. An integer specifying a group. *groupNumber* is 1 for the first group. If *groupNumber* is omitted, the first group is used.

Placement[156]: [2].

See Also

InputVariableLabel Method[295]

The following program fits the model from Example 8 to the **Grant** data, and then displays in the debug window some characteristics of the variables in the dataset.

```
Imports System.Diagnostics
Module MainModule
    ' InputVariableName Method Example
    Sub Main()
        Dim i As Integer
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grant")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Debug.WriteLine("Number of variables: " & Sem.DataFileNVariables)
        For i = 1 To Sem.DataFileNVariables
            Debug.WriteLine("")
            Debug.WriteLine("Variable number " & i)
            Debug.WriteLine("Name: " & Sem.InputVariableName(i))
            Debug.WriteLine("Label: " & Sem.InputVariableLabel(i))

            If Sem.InputVariableIsNumeric(i) Then
                Debug.WriteLine("Numeric")
            Else
                Debug.WriteLine("Non-numeric")
            End If

            If Sem.InputVariableHasMissingValues(i) Then
                Debug.WriteLine("Has missing values")
            Else
                Debug.WriteLine("Doesn't have missing values")
            End If
        Next

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.70 Intercept Method

Specifies an intercept as a model parameter.

Syntax

object.**Intercept** (*variableName*)
object.**Intercept** (*variableName*, *parameterValue*)
object.**Intercept** (*variableName*, *parameterName*)

The **Intercept** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
variableName	The character string, *variableName*, is the name of an endogenous variable. The intercept in the regression equation for predicting *variableName* is a model parameter.
parameterValue	(Optional) Parameter value. If *parameterValue* = 3, say, then the intercept is fixed at 3.
parameterName	(Optional) Parameter name. If parameterName = "abc", say, then the intercept is named "abc". It is constrained to be equal to any other parameters named "abc". If parameterName = "3?", say then the intercept is given an initial value of 3, and is unconstrained. If parameterName = "abc:3", say, then the intercept is named "abc" and is given an initial value of 3. It is constrained to be equal to any other parameters named "abc". If parameterName is an empty string (""), the intercept is an unconstrained parameter.

Placement 156: [2].

Default

When the ModelMeansAndIntercepts 310 method is not used, then all intercepts are unconstrained. However, they are not estimated.

When the ModelMeansAndIntercepts 310 method is used, the following default assumptions are made about intercepts that are not constrained or fixed at constant values by use of the AStructure 169 or Intercept 299 methods.

Intercepts for the prediction of observed, endogenous variables are free parameters.

Intercepts for the prediction of unobserved, endogenous variables are fixed at zero.

Remarks

If neither parameterValue nor parameterName is present, the intercept is an unconstrained parameter.

See Also

Mean Method 303

ModelMeansAndIntercepts Method 310

The following program uses the Path 364, **Intercept** and Mean 303 methods to specify the model in Example 14.

```
Module MainModule
    ' Intercept Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.TextOutput()
        Sem.Standardized()
        Sem.Smc()
        Sem.ImpliedMoments()
        Sem.SampleMoments()

        Sem.ModelMeansAndIntercepts()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Warren5v")
        Sem.Path("performance", "know ledge")
        Sem.Path("performance", "value")
        Sem.Path("performance", "satisfaction")
        Sem.Path("performance", "error", 1)
        Sem.Intercept("performance")
        Sem.Mean("know ledge")
        Sem.Mean("value")
        Sem.Mean("satisfaction")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.71 Interrupt Method

Stops any ongoing calculations. **Interrupt** is equivalent to selecting **Analyze** → **Stop Calculating Estimates** from the Amos Graphics menu.

Syntax

object.**Interrupt** ()

The **Interrupt** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement[156]: [3].

4.5.2.2.2.72 IsModelingMeansAndIntercepts Method

True if means and intercepts are explicit model parameters.

Syntax

result = *object*.**IsModelingMeansAndIntercepts** ()

The **IsModelingMeansAndIntercepts** method syntax has the following parts:

Part	Description
result	True if means and intercepts are explicit model parameters.
object	An object of type **AmosEngine**.

Placement[156]: [3].

See Also

Use the AmosEngine class to test for scale- and location-invariance[501]

4.5.2.2.2.73 Iterations Method

Places a limit on the number of iterations Amos will perform. If this limit is reached, Amos will stop after reporting its current parameter estimates, even if the convergence criteria (see the Crit1[217] and Crit2[218] methods) have not been met.

Syntax

object.**Iterations** (*nIterations*)

The **Iterations** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
nIterations	Upper bound on the number of iterations.

Placement 156: [1].

Default

There is no limit on the number of iterations.

See Also

Crit1 Method 217

Crit2 Method 218

Fisher Method 233

Technical Method 397

Time Method 400

This example demonstrates the **Iterations** method.

```
Module MainModule
    ' Iterations Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine

        Sem.Iterations(20)

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.74 LineLength Method

The **LineLength** method has no effect. It is provided for compatibility with earlier versions of Amos.

Syntax

object.**LineLength** (*nCharacters*)

The **LineLength** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

nCharacters	Number of characters per line.

Placement[156]: [1].

4.5.2.2.2.75 MaxDecimalPlaces Method

The **MaxDecimalPlaces** method has no effect. It is provided for compatibility with earlier versions of Amos.

Syntax

object.**MaxDecimalPlaces** (*nDigits*)

The **MaxDecimalPlaces** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
nDigits	Maximum number of decimal places.

Placement[156]: [1].

4.5.2.2.2.76 Mean Method

Specifies a mean as a model parameter.

Syntax

object.**Mean** (*variableName*)
object.**Mean** (*variableName*, *parameterValue*)
object.**Mean** (*variableName*, *parameterName*)

The **Mean** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
variableName	The character string, variableName, is the name of an exogenous variable. The **Mean** method makes that variable's mean a model parameter.
parameterValue	(Optional) Parameter value. If *parameterValue* = 3, say, then the mean is fixed at 3.

parameterName	(Optional) Parameter name. If parameterName = "abc", say, then the mean is named "abc". It is constrained to be equal to any other parameters named "abc". If parameterName = "3?", say then the mean is given an initial value of 3, and is unconstrained. If parameterName = "abc:3", say, then the mean is named "abc" and is given an initial value of 3. It is constrained to be equal to any other parameters named "abc". If parameterName is an empty string (""), the mean is an unconstrained parameter.

Placement[156]: [2].

Default

When the ModelMeansAndIntercepts[310] method is not used, then the means of all exogenous variables are unconstrained. However, the means are not estimated.

When the ModelMeansAndIntercepts[310] method is used, the following default assumptions are made about means that are not constrained or fixed at constant values by use of the MStructure[315] or **Mean** methods.

The means of observed, exogenous variables are free parameters.

The means of unobserved, exogenous variables are fixed at zero.

Remarks

If *parameterValue* and *parameterName* are omitted, the mean is an unconstrained parameter.

See Also

Intercept Method[299]

ModelMeansAndIntercepts Method[310]

The following program uses the Path[364], Intercept[299] and **Mean** methods to specify the model in Example 14.

```
Module MainModule
    ' Mean Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine

        Sem.TextOutput()
        Sem.Standardized()
        Sem.Smc()
        Sem.ImpliedMoments()
        Sem.SampleMoments()

        Sem.ModelMeansAndIntercepts()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Warren5v")
        Sem.Path("performance", "know ledge")
        Sem.Path("performance", "value")
        Sem.Path("performance", "satisfaction")
        Sem.Path("performance", "error", 1)
        Sem.Intercept("performance")
        Sem.Mean("know ledge")
        Sem.Mean("value")
        Sem.Mean("satisfaction")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.77 MinDecimalPlaces Method

The **MinDecimalPlaces** method has no effect. It is provided for compatibility with earlier versions of Amos.

Syntax

object.**MinDecimalPlaces** (*nDigits*)

The **MinDecimalPlaces** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
nDigits	Minimum number of decimal places.

Placement[156]: [1].

4.5.2.2.2.78 Ml Method

Requests estimation by the method of maximum likelihood, minimizing (D1) together with (D2) in Appendix B.

Syntax

object.**Ml** ()

The **Ml** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement[156]: [1].

Default

When you do not specify an estimation criterion, the maximum likelihood criterion (Ml[305] method) is used.

See Also

Adf Method[163]

BootMl Method[185]

Gls Method[284]

Sls Method[392]

Uls Method[403]

This example demonstrates the **Ml** method.

```
Module MainModule
    ' Ml Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.Ml()

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.79 Model Method

Places equality constraints on model parameters.

Syntax

object.**Model** (*modelName*, *constraint1*)

object.**Model** (*modelName*, *constraint1*, *constraint2*)

object.**Model** (*modelName*, *constraint1*, *constraint2*, *constraint3*)

...

The **Model** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
modelName	A name for the complete set of constraints consisting of constraint1, constraint2, constraint3, etc.
constraint1	Either: A string of the form p1=p2=p3=..., where each pi is either a parameter name or a number. At most one of the pi can be a number. Or: The name of a model defined by another use of the **Model** method.
constraint2	Same as constraint1
constraint3	Same as constraint1
...	...

Placement 156: [2].

Default

No additional parameter constraints are imposed beyond those specified by the assignment of names and values to parameters by use of the Path 364, Cov 212, Var 405, Mean 303, Intercept 299, AStructure 169 and MStructure 315 methods.

Remarks

In order to use the **Model** method, you need to be able to refer to parameters by name. Parameters can be named by the Path 364, Cov 212, Var 405, Mean 303, Intercept 299, AStructure 169 and MStructure 315 methods.

You can use the **Model** method as many times as you want.

This example demonstrates the **Model** method.

```
Module MainModule
    ' Model Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Wheaton")
        Sem.Path("anomia67", "67_alienation", 1)
        Sem.Path("anomia67", "eps1", 1)
        Sem.Path("pow les67", "67_alienation")
        Sem.Path("pow les67", "eps2", 1)
        Sem.Path("anomia71", "71_alienation", 1)
        Sem.Path("anomia71", "eps3", 1)
        Sem.Path("pow les71", "71_alienation")
        Sem.Path("pow les71", "eps4", 1)
        Sem.Path("67_alienation", "ses")
        Sem.Path("67_alienation", "zeta1", 1)
        Sem.Path("71_alienation", "67_alienation")
        Sem.Path("71_alienation", "ses")
        Sem.Path("71_alienation", "zeta2", 1)
        Sem.Path("education", "ses", 1)
        Sem.Path("education", "delta1", 1)
        Sem.Path("SEI", "ses")
        Sem.Path("SEI", "delta2", 1)
        Sem.Var("eps1", "var1")
        Sem.Var("eps2", "var2")
        Sem.Var("eps3", "var3")
        Sem.Var("eps4", "var4")
        Sem.Cov("eps1", "eps3", "cov1")
        Sem.Cov("eps2", "eps4", "cov2")

        Sem.Model("B", "cov1 = cov2 = 0")
        Sem.Model("C", "cov2 = 0")
        Sem.Model("D")
        Sem.Model("E", "var1 = var3")
        Sem.Model("F", "var2 = var4")
        Sem.Model("G", "E", "F")

        Sem.Dispose()
    End Sub
End Module
```

Many modeling efforts require fitting several alternative models to the same data. You can fit many models at once provided that each model can be obtained by placing equality constraints on the parameters of one special, 'most general', model. The example shows how to do this. First the Path 364, Var 405 and Cov 212 methods are used to specify Jöreskog and Sörbom's (1989 517, p. 205) Model D for the data of Wheaton et al. (1977) 527.

```
Sub Main()
    Dim Sem As New AmosEngineLib.AmosEngine

    Sem.TextOutput()

    Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Wheaton")
    Sem.Path("anomia67", "67_alienation", 1)
    Sem.Path("anomia67", "eps1", 1)
    Sem.Path("pow les67", "67_alienation")
    Sem.Path("pow les67", "eps2", 1)
    Sem.Path("anomia71", "71_alienation", 1)
    Sem.Path("anomia71", "eps3", 1)
    Sem.Path("pow les71", "71_alienation")
    Sem.Path("pow les71", "eps4", 1)
    Sem.Path("67_alienation", "ses")
    Sem.Path("67_alienation", "zeta1", 1)
    Sem.Path("71_alienation", "67_alienation")
    Sem.Path("71_alienation", "ses")
    Sem.Path("71_alienation", "zeta2", 1)
    Sem.Path("education", "ses", 1)
    Sem.Path("education", "delta1", 1)
    Sem.Path("SEI", "ses")
    Sem.Path("SEI", "delta2", 1)
    Sem.Var("eps1", "var1")
    Sem.Var("eps2", "var2")
    Sem.Var("eps3", "var3")
    Sem.Var("eps4", "var4")
    Sem.Cov("eps1", "eps3", "cov1")
    Sem.Cov("eps2", "eps4", "cov2")

    Sem.Model("B", "cov1 = cov2 = 0")
    Sem.Model("C", "cov2 = 0")
    Sem.Model("D")
    Sem.Model("E", "var1 = var3")
    Sem.Model("F", "var2 = var4")
    Sem.Model("G", "E", "F")

    Sem.Dispose()
End Sub
```

Six parameters are named - cov1, cov2, var1, var2, var3, var4. However, since no two parameters share the same name, the presence of the names does not place any constraints on the parameters. The purpose of the names is to allow the **Model** method to place constraints on the named parameters.

Jöreskog and Sörbom proposed other models besides Model D. All but one of them can be obtained by constraining Model D. For instance, their Model C is just like Model D, but with the parameter named cov2 (the covariance between eps2 and eps4) fixed at zero. Their Model B goes even further. It assumes that two parameters (cov1 and cov2) are zero. Amos analyzes Models B and C along with Model D if you add the following lines to the program.

```
Sem.Model("B", "cov1 = cov2 = 0")
Sem.Model("C", "cov2 = 0")
Sem.Model("D")
```

The first two lines are self-explanatory - they name and describe Models B and C. You may be surprised that the third line is necessary. It declares that there is a model called Model D that employs no additional constraints beyond those specified by the Path 364, Var 405 and Cov 212 methods. This line is necessary if you want to analyze Model D. The rule is that, if you use the **Model** method at all, Amos will only analyze models explicitly defined through use of the **Model** method. This convention allows you to specify an unidentified model, and then to supply enough constraints with each use of the **Model** method to identify

the model. If you don't use the **Model** method at all, however, Amos will perform a single analysis — of the model as specified by the Path 364, Cov 212, Var 405, Mean 303, Intercept 299, AStructure 169 and MStructure 315 methods.

It may be possible to specify the same set of constraints in several equivalent ways. Model B, for instance, could have been specified in the following way.

```
Sem.Model("B", "cov1 = 0", "cov2 = 0")
```

Here is another, equivalent, variation.

```
Sem.Model("B", "cov1 = cov2", "cov2 = 0")
```

There is a shorthand for indicating that one model incorporates all of the constraints of another model. In the present example, Model B includes all of the constraints of Model C, as well as one additional constraint, so Model B could be specified this way:

```
Sem.Model("B", "C", "cov1 = 0")
```

The example specified three more models for the Wheaton data. Notice that var1 and var3 are unique variances associated with anomia measurements made in 1967 and 1971. It is a plausible hypothesis that the unique variance of anomia was the same in both years. This hypothesis was incorporated into a new model by adding this line to the program.

```
Sem.Model("E", "var1 = var3")
```

Similarly, since var2 and var4 are unique variances associated with powerlessness measurements made in 1967 and 1971, it is plausible to set up a model in which those two variances are required to be equal:

```
Sem.Model("F", "var2 = var4")
```

Finally, both of the models just described could be right. In other words, all of the 1971 parameter values could be the same as the corresponding 1967 values. The following model specification imposes both sets of constraints.

```
Sem.Model("G", "E", "F")
```

4.5.2.2.2.80 ModelMeansAndIntercepts Method

Specifies that means (of exogenous variables) and intercepts (in the equations for predicting endogenous variables) are explicit model parameters. The **ModelMeansAndIntercepts** method must be used in order to allow the use of the Intercept 299, Mean 303 or MStructure 315 methods or the specification of an intercept through use of the AStructure 169 method.

Syntax

object.**ModelMeansAndIntercepts** ()

The **ModelMeansAndIntercepts** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement 156: [1].

Default

When the **ModelMeansAndIntercepts** method is not used, means and intercepts are not constrained and not estimated.

Remarks

When the **ModelMeansAndIntercepts** method is used, means and intercepts are fixed at zero by default. Constraints on means can be changed with the Mean 303 and MStructure 315 methods. Constraints on intercepts can be changed with the Intercept 299 and AStructure 169 methods.

See Also

Intercept Method 299

Mean Method 303

MStructure Method 315

This example demonstrates the **ModelMeansAndIntercepts** method.

```
Module MainModule
    ' ModelMeansAndIntercepts Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine

        Sem.TextOutput()

        Sem.ModelMeansAndIntercepts()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
        Sem.MStructure("academic (4)")
        Sem.MStructure("athletic")
        Sem.MStructure("attract (abc)")
        Sem.MStructure("gpa (abc)")
        Sem.MStructure("height (20?)")
        Sem.MStructure("w eight (xyz : 10)")
        Sem.MStructure("rating (xyz : 10)")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.81 Mods Method

Displays the modification indices described by Jöreskog and Sörbom (1984) 517.

Syntax

object.**Mods** (*threshold*)

The **Mods** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

threshold	Optional. Only modification indices that exceed *threshold* are displayed. The default value for *threshold* is 4.

Placement[156]: [1].

Default

Modification indices are not reported.

Remarks

Amos computes a modification index for each parameter that is fixed at a constant value and for each parameter that is required to equal some other parameter. The modification index for a parameter is an estimate of the amount by which the discrepancy function would decrease if the analysis were repeated with the constraints on that parameter removed. The actual decrease that would occur may be much greater.

Amos computes modification indices not only for parameters that are explicitly constrained, but also for parameters that are implicitly assumed to be zero. For example, a modification index is computed for every covariance that is fixed at zero by default.

Amos also computes modification indices for paths that do not appear in a model, giving the approximate amount by which the discrepancy function would decrease if such a path were introduced. There are, however, two types of nonexistent paths for which Amos does not compute a modification index. First, Amos does not compute a modification index for a nonexistent path which, if introduced, would convert an exogenous variable into an endogenous variable. Second, Amos does not compute a modification index for a nonexistent path that, if introduced, would create an indirect path from a variable to itself where none already exists. In particular, Amos does not compute a modification index for a nonexistent path that, if introduced, would convert a recursive model to a nonrecursive one.

Each time Amos displays a modification index for a parameter, it also displays an estimate of the amount by which the parameter would change from its current, constrained value if the constraints on it were removed.

Specifying a small value for *threshold* can result in the output of a large number of modification indices.

This example demonstrates the **Mods** method.

```
Module MainModule
    ' Mods Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.Mods(4)

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Wheaton")
        Sem.AStructure("anomia67 <--- 67_alienation (1)")
        Sem.AStructure("anomia67 <--- eps1 (1)")
        Sem.AStructure("pow les67 <--- 67_alienation")
        Sem.AStructure("pow les67 <--- eps2 (1)")
        Sem.AStructure("anomia71 <--- 71_alienation (1)")
        Sem.AStructure("anomia71 <--- eps3 (1)")
        Sem.AStructure("pow les71 <--- 71_alienation")
        Sem.AStructure("pow les71 <--- eps4 (1)")
        Sem.AStructure("67_alienation <--- ses")
        Sem.AStructure("67_alienation <--- zeta1 (1)")
        Sem.AStructure("71_alienation <--- 67_alienation")
        Sem.AStructure("71_alienation <--- ses")
        Sem.AStructure("71_alienation <--- zeta2 (1)")
        Sem.AStructure("education <--- ses (1)")
        Sem.AStructure("education <--- delta1 (1)")
        Sem.AStructure("SEI <--- ses")
        Sem.AStructure("SEI <--- delta2 (1)")

        Sem.Dispose()
    End Sub
End Module
```

The example (which is the same as Model A in Example 6 of the *User's Guide*) yields the following modification indices.

Covariances: (Group number 1 - Default model)

			M.I.	Par Change
eps2	<-->	delta1	5.905	-.424
eps2	<-->	eps4	26.545	.825
eps2	<-->	eps3	32.071	-.988
eps1	<-->	delta1	4.609	.421
eps1	<-->	eps4	35.367	-1.069
eps1	<-->	eps3	40.911	1.253

Variances: (Group number 1 - Default model)

	M.I.	Par Change

Regression Weights: (Group number 1 - Default model)

			M.I.	Par Change
powles71	<---	powles67	5.457	.057
powles71	<---	anomia67	9.006	-.065
anomia71	<---	powles67	6.775	-.069
anomia71	<---	anomia67	10.352	.076
powles67	<---	powles71	5.612	.054
powles67	<---	anomia71	7.278	-.054
anomia67	<---	powles71	7.706	-.070
anomia67	<---	anomia71	9.065	.068

The largest modification index is 40.911, indicating that the chi-square statistic will drop by at least 40.911 if the covariance between **eps1** and **eps3** is allowed to depart from zero (the value at which it is fixed in Model A). The number 1.254 in the **Par Change** column indicates that the covariance will increase by about 1.254 if it is free to take on any value. Of course if the covariance (now zero) increases by 1.254 it will then be equal to 1.254. Actually, in Model B of Example 6, where the covariance between **eps1** and **eps3** is unconstrained, its estimate is 1.888. Kaplan (1989)[517] and Saris, Satorra and Sörbom (1987)[524] discuss the use of estimated parameter changes in exploratory analyses.

4.5.2.2.2.82 MonteCarlo Method

Controls whether a parametric bootstrap or a nonparametric bootstrap (Efron & Tibshirani, 1993[513]) is performed.

When the **MonteCarlo** method is used, bootstrap samples are drawn from a multivariate normal population whose means, variances and covariances are the same as the sample means, variances and covariances.

Syntax

object.**MonteCarlo** ()
object.**MonteCarlo** (*tf*)

The **MonteCarlo** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	True (default) specifies a parametric bootstrap. False specifies a nonparametric bootstrap.

Placement[156]: [1].

Default

A nonparametric bootstrap is performed. (Bootstrap samples are drawn with replacement from the original sample. Raw data input is required.)

Remarks

MonteCarlo allows bootstrapping to be carried out (with the assumption of normality) when raw data are not available. If you do not use the Bootstrap 188 method, the **MonteCarlo** method has no effect.

See Also

Bootstrap Method 188

Seed Method 390

This example demonstrates the **MonteCarlo** method.

```
Module MainModule
    ' MonteCarlo Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine

        Sem.MonteCarlo()
        Sem.Bootstrap(2000)

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.83 MStructure Method

Frees or constrains the mean of an exogenous variable.

The **Mean** method accomplishes the same thing, and is recommended for new Amos programs. The syntax of the MStructure 315 method resembles the syntax of the **$MStructure** command in previous versions of Amos, and is provided to assist in the translation of old Amos input files.

Syntax

object.**MStructure** (*s*)

The **MStructure** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

s	A character string in one of the following forms 1. variablename 2. variablename (number) 3. variablename (parametername) 4. variablename (number ?) 5. variablename (parametername : number) In the first form, the mean of the variable named variablename is a free parameter. In the second form, the mean of the variable named variablename is fixed at number. In the third form, the mean of the variable named variablename is given the name parametername. In the fourth form, the mean of the variable named variablename is a free parameter, and is given an initial value of number. In the fifth form, the mean of the variable named variablename is given the name parametername and an initial value of number.

Placement[156]: [2].

Default

When the ModelMeansAndIntercepts[310] method is not used, then all means and intercepts are unconstrained. However, no means or intercepts are estimated.

When the ModelMeansAndIntercepts[310] method is used, the following default assumptions are made about exogenous variables that are not constrained or fixed at constant values by use of the **MStructure** or Mean[303] methods.

The means of observed, exogenous variables are free parameters.

The means of unobserved, exogenous variables are fixed at zero.

Remarks

The ModelMeansAndIntercepts[310] method must be used before using the **MStructure** method.

It is possible to name an endogenous variable as an argument to the **MStructure** method. Doing so has the effect of freeing or constraining the intercept in the regression equation for predicting that variable. However, the following methods for specifying constraints on intercepts are recommended.

- Include the intercept in a linear equation through use of the AStructure[169] method.
- Use the Intercept[299] method.

See Also

Intercept Method 299

Mean Method 303

ModelMeansAndIntercepts Method 310

AStructure Method 169

In the following program, the mean of academic is fixed at 4. The mean of athletic is not constrained in any way. attract and gpa are required to have the same mean because both means are named 'abc'. The mean of height is given an initial value of 20. The means of weight and rating are required to be equal and are given an initial value of 10.

```
Module MainModule
    ' MStructure Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.TextOutput()
        Sem.ModelMeansAndIntercepts()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
        Sem.MStructure("academic (4)")
        Sem.MStructure("athletic")
        Sem.MStructure("attract (abc)")
        Sem.MStructure("gpa (abc)")
        Sem.MStructure("height (20?)")
        Sem.MStructure("w eight (xyz : 10)")
        Sem.MStructure("rating (xyz : 10)")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.84 Ncp, NcpLo, NcpHi Methods

Ncp gets a point estimate of the noncentrality parameter. **NcpLo** and **NcpHi** get the lower and upper boundaries of a 90% confidence interval for the noncentrality parameter.

Syntax

object.**Ncp** ()
object.**NcpLo** ()
object.**NcpHi** ()

The **Ncp**, **NcpLo** and **NcpHi** method syntaxes have the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement 156: [3].

Remarks

If you have used the Model 306 method to define more than one model, the **Ncp**, **NcpLo** and **NcpHi** methods return estimates for the most recently fitted model. The second example shows how to obtain estimates for multiple models.

The following program shows how to display various fit measures when only one model is defined (i.e., when the Model 306 method has been used only once or not at all).

```
Imports System.Diagnostics
Module MainModule
    ' Ncp, NcpLo and NcpHi Methods Example 1
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Debug.WriteLine("Chi Square = " & Sem.Cmin)
        Debug.WriteLine("Degrees of Freedom = " & Sem.df)
        Debug.WriteLine("p = " & Sem.p)
        Debug.WriteLine("Number of parameters = " & Sem.npar)
        Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & _
              Sem.NcpLo & ", " & Sem.NcpHi & ")")
        Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
        Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)

        Sem.Dispose()
    End Sub
End Module
```

The following program shows how to display various fit measures for multiple models (i.e., when the Model 306 method has been used more than once).

```
Imports System.Diagnostics
Module MainModule
   ' Ncp, NcpLo and NcpHi Methods Example 2
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Dim i As Integer

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (a) spatial + (1) err_v")
      Sem.AStructure("cubes = (b) spatial + (1) err_c")
      Sem.AStructure("lozenges = (c) spatial + (1) err_l")
      Sem.AStructure("paragraph = (d) verbal + (1) err_p")
      Sem.AStructure("sentence = (e) verbal + (1) err_s")
      Sem.AStructure("w ordmean = (f) verbal + (1) err_w ")
      Sem.Var("spatial", 1)
      Sem.Var("verbal", 1)

      Sem.Model("Congeneric")
      Sem.Model("tau-equivalent", "a = b = c", "d = e = f")

      For i = 1 To 2
         Debug.WriteLine("")
         Debug.WriteLine("Model number " & i)
         Sem.FitModel(i)
         Debug.WriteLine("Chi Square = " & Sem.Cmin)
         Debug.WriteLine("Degrees of Freedom = " & Sem.df)
         Debug.WriteLine("p = " & Sem.p)
         Debug.WriteLine("Number of parameters = " & Sem.npar)
         Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & _
               Sem.NcpLo & ", " & Sem.NcpHi & ")")
         Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
         Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)
      Next

      Sem.Dispose()
   End Sub
End Module
```

Controls whether the covariance matrix of estimates is estimated by inverting the expected second derivatives or the exact second derivatives.

Syntax

object.**ObservedInfo** ()

object.**ObservedInfo** (*tf*)

The **ObservedInfo** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. If *tf* is True (the default), the covariance matrix of estimates is estimated by inverting the matrix of exact second derivatives. If *tf* is False, the

	matrix of expected second derivatives is used.

Placement[156]: [1].

Default

The matrix of expected second derivatives is used.

Remarks

See Efron and Hinkley (1978)[513] for a discussion of exact versus expected second derivatives in the estimation of the covariance matrix of estimates.

This example demonstrates the **ObservedInfo** method.

```
Module MainModule
    ' ObservedInfo Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.ObservedInfo()

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.85 NeedBCLow erBounds, NeedBCUpperBounds Methods

Declares that GetBCLowerBounds[243] or GetBCUpperBounds[243] will be used later in the program.

Syntax

object.**NeedBCLowerBounds** (*matrixID*)
object.**NeedBCUpperBounds** (*matrixID*)

The **NeedBCLowerBounds** and **NeedBCLowerBounds** method syntaxes have the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.

Placement[156]: [1].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.

StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

In order to use GetBCLowerBounds 243 (*matrixID*), you must first use **NeedBCLowerBounds** (*matrixID*). For example, you have to use

object.**NeedBCLowerBounds** (FactorScoreWeights)

before using

object.GetBCLowerBounds 243 (FactorScoreWeights, …)

Similarly, in order to use GetBCUpperBounds 243 (*matrixID*), you must first use **NeedBCUpperBounds** (*matrixID*). For example, you have to use

object.**NeedBCUpperBounds** (FactorScoreWeights)

before using

object.GetBCUpperBounds 243 (FactorScoreWeights, …)

This example demonstrates the **NeedBCLowerBounds** and **NeedBCUpperBounds** methods.

```
Imports System.Diagnostics
Imports AmosEngineLib.AmosEngine.TMatrixID
Imports Microsoft.VisualBasic
Module MainModule
    ' NeedBCLowerBounds, NeedBCUpperBounds Methods Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.Bootstrap(2000)
        Sem.ConfidenceBC(90) '90% confidence intervals

        Sem.NeedBCLowerBounds(FactorScoreWeights)
        Sem.NeedBCUpperBounds(FactorScoreWeights)

        Sem.Standardized()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("wordmean = verbal + (1) err_w")

        Dim X(,) As Double
        Dim RNames() As String
        Dim CNames() As String

        'Get the row and column variable names
        Sem.RowNames(FactorScoreWeights, RNames)
        Sem.ColumnNames(FactorScoreWeights, CNames)

        'Print the lower bounds
        Sem.GetBCLowerBounds(FactorScoreWeights, X)
        Debug.WriteLine(vbCrLf & "Confidence intervals on factor score weights -- lower bound")
        PrintMatrix(X, CNames, RNames
)
        'Print the upper bounds
        Sem.GetBCUpperBounds(FactorScoreWeights, X)
        Debug.WriteLine(vbCrLf & "Confidence intervals on factor score weights -- upper bound")
        PrintMatrix(X, CNames, RNames)

        Sem.Dispose()
    End Sub

    'Print a matrix in the debug window
    Sub PrintMatrix(ByVal TheMatrix(,) As Double, ByVal CNames$(), ByVal RNames$())
        Dim NRows1 As Integer, NColumns1 As Integer
        Dim i As Integer, j As Integer
        NRows1 = UBound(RNames)
        NColumns1 = UBound(CNames)

        Debug.Write(" ")
        For j = 0 To NColumns1
            Debug.Write(CNames(j).PadLeft(10))
        Next

        Debug.WriteLine("")
        For i = 0 To NRows1
            Debug.Write(RNames(i).PadRight(8))
            For j = 0 To NColumns1
```

```
            Debug.Write(TheMatrix(i, j).ToString(".00000").PadLeft(10))
        Next
        Debug.WriteLine("")
    Next
  End Sub
End Module
```

4.5.2.2.2.86 NeedBootSampleEstimates Method

Declares that GetBootSampleEstimates [251] will be used to get a matrix of estimates from an individual bootstrap sample.

Syntax

object.**NeedBootSampleEstimates** (*matrixID*)

The **NeedBootSampleEstimates** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.

Placement [156]: [1].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.

AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

You have to use **NeedBootSampleEstimates** to declare that estimates of a matrix will be needed before you can use GetBootSampleEstimates 251 to obtain the estimates from a bootstrap sample. For example, you have to use

object.**NeedBootSampleEstimates** (StandardizedTotalEffects)

before using

object.GetBootSampleEstimates 251 (StandardizedTotalEffects, ...)

This example demonstrates the **NeedBootSampleEstimates** method.

```
Imports System.Diagnostics
Imports AmosEngineLib.AmosEngine.TMatrixID
Imports Microsoft.VisualBasic
Module MainModule
    ' NeedBootSampleEstimates Method Example
    Sub Main()
        Const NBootSamples As Integer = 3
        Dim i As Integer
        Dim X(,) As Double
        Dim RNames() As String
        Dim CNames() As String
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.NeedBootSampleEstimates(FactorScoreWeights)

        Sem.Bootstrap(NBootSamples)
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("wordmean = verbal + (1) err_w")

        Sem.RowNames(FactorScoreWeights, RNames)
        Sem.ColumnNames(FactorScoreWeights, CNames)
        For i = 1 To NBootSamples
            Sem.GetBootSampleEstimates(FactorScoreWeights, X, i)
            Debug.WriteLine("")
            Debug.WriteLine("Factor scores weights from bootstrap sample #" & i)
            Debug.WriteLine("")
            PrintMatrix(X, CNames, RNames)
        Next

        Sem.Dispose()
    End Sub

    'Print a matrix in the debug window
    Sub PrintMatrix(ByVal TheMatrix(,) As Double, ByVal CNames$(), ByVal RNames$())
        Dim NRows1 As Integer, NColumns1 As Integer
        Dim i As Integer, j As Integer
        NRows1 = UBound(RNames)
        NColumns1 = UBound(CNames)

        Debug.Write(" ")
        For j = 0 To NColumns1
            Debug.Write(CNames(j).PadLeft(10))
        Next

        Debug.WriteLine("")
        For i = 0 To NRows1
            Debug.Write(RNames(i).PadRight(8))
            For j = 0 To NColumns1
                Debug.Write(TheMatrix(i, j).ToString(".00000").PadLeft(10))
            Next
            Debug.WriteLine("")
        Next
    End Sub
```

```
End Module
```

4.5.2.2.2.87 NeedEstimates Method

Declares that GetEstimates [260] will be used to get a matrix of estimates.

Syntax

object.**NeedEstimates** (*matrixID*)

The **NeedEstimates** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.

Placement [156]: [1].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the

		exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

You have to use **NeedEstimates** to declare that estimates of a matrix will be needed before you can use GetEstimates 260 to obtain the estimates. For example, you have to use

object.**NeedEstimates** (StandardizedTotalEffects)

before using

object.GetEstimates 260 (StandardizedTotalEffects, ...)

The following program fits Models A and B of Example 11. It displays the matrix of total effects for each group and model.

```
Imports System.Diagnostics
Imports AmosEngineLib.AmosEngine.TMatrixID
Imports Microsoft.VisualBasic
Module MainModule
    ' NeedEstimates Method Example
    Sub Main()
        Dim CNames() As String, RNames() As String, X(,) As Double
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.NeedEstimates(TotalEffects)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
        Sem.GroupName("girls")
        Sem.AStructure("academic = (g1) GPA + (g2) attract + (1) e1")
        Sem.AStructure("attract = (g3) height + (g4) w eight + (g5) rating + (g6) academic + (1) e2")
        Sem.AStructure("e2 <--> e1")

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
        Sem.GroupName("boys")
        Sem.AStructure("academic = (b1) GPA + (b2) attract + (1) e1")
        Sem.AStructure("attract = (b3) height + (b4) w eight + (b5) rating + (b6) academic + (1) e2")
        Sem.AStructure("e2 <--> e1")

        Sem.Model("Model_A")
        Sem.Model("Model_B", "g1=b1", "g2=b2", "g3=b3", "g4=b4", "g5=b5", "g6=b6")

        'Print implied covariances for each model and each group
        Dim ModelNumber As Integer
        Dim GroupNumber As Integer
        For ModelNumber = 1 To 2
            Sem.FitModel(ModelNumber)
            For GroupNumber = 1 To 2
                Sem.GetEstimates(TotalEffects, X, GroupNumber)
                Sem.ColumnNames(TotalEffects, CNames, GroupNumber)
                Sem.Row Names(TotalEffects, RNames, GroupNumber)
                Debug.WriteLine(vbCrLf & "Group " & GroupNumber & ", Model " & ModelNumber)
                PrintMatrix(X, CNames, RNames)
            Next
        Next

        Sem.Dispose()
    End Sub

    'Print a matrix in the debug w indow
    Sub PrintMatrix(ByVal TheMatrix(,) As Double, ByVal CNames$(), ByVal RNames$())
        Dim NRow s1 As Integer, NColumns1 As Integer
        Dim i As Integer, j As Integer
        NRow s1 = UBound(RNames)
        NColumns1 = UBound(CNames)

        Debug.Write(" ")
        For j = 0 To NColumns1
            Debug.Write(CNames(j).PadLeft(10))
        Next

        Debug.WriteLine("")
        For i = 0 To NRow s1
            Debug.Write(RNames(i).PadRight(8))
            For j = 0 To NColumns1
                Debug.Write(TheMatrix(i, j).ToString(".00000").PadLeft(10))
```

```
        Next
        Debug.WriteLine("")
    Next
  End Sub
End Module
```

4.5.2.2.2.88 NeedPCLowerBounds, NeedPCUpperBounds Methods

Declares that GetPCLowerBounds 269 or GetPCUpperBounds 269 will be used later in the program.

Syntax

object.**NeedPCLowerBounds** (*matrixID*)
object.**NeedPCUpperBounds** (*matrixID*)

The **NeedPCLowerBounds** and **NeedPCLowerBounds** method syntaxes have the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.

Placement 156: [1].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the

		exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

In order to use GetPCLowerBounds 269 (*matrixID*), you must first use **NeedPCLowerBounds** (*matrixID*). For example, you have to use

object.**NeedPCLowerBounds** (FactorScoreWeights)

before using

object.GetPCLowerBounds 269 (FactorScoreWeights, …)

Similarly, in order to use GetPCUpperBounds 269 (*matrixID*), you must first use **NeedPCUpperBounds** (*matrixID*). For example, you have to use

object.**NeedPCUpperBounds** (FactorScoreWeights)

before using

object.GetPCUpperBounds 269 (FactorScoreWeights, …)

This example demonstrates the **NeedPCLowerBounds** and **NeedPCUpperBounds** methods.

```
Imports System.Diagnostics
Imports AmosEngineLib.AmosEngine.TMatrixID
Imports Microsoft.VisualBasic
Module MainModule
    ' NeedPCLowerBounds, NeedPCUpperBounds Methods Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.Bootstrap(2000)
        Sem.ConfidencePC(90) '90% confidence intervals

        Sem.NeedPCLowerBounds(FactorScoreWeights)
        Sem.NeedPCUpperBounds(FactorScoreWeights)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("wordmean = verbal + (1) err_w")

        Dim X(,) As Double
        Dim RNames() As String
        Dim CNames() As String

        'Get the row and column variable names
        Sem.RowNames(FactorScoreWeights, RNames)
        Sem.ColumnNames(FactorScoreWeights, CNames)

        'Print the lower bounds
        Sem.GetPCLowerBounds(FactorScoreWeights, X)
        Debug.WriteLine(vbCrLf & "Confidence intervals on factor score weights -- lower bound")
        PrintMatrix(X, CNames, RNames)

        'Print the upper bounds
        Sem.GetPCUpperBounds(FactorScoreWeights, X)
        Debug.WriteLine(vbCrLf & "Confidence intervals on factor score weights -- upper bound")
        PrintMatrix(X, CNames, RNames)

        Sem.Dispose()
    End Sub

    'Print a matrix in the debug window
    Sub PrintMatrix(ByVal TheMatrix(,) As Double, ByVal CNames$(), ByVal RNames$())
        Dim NRows1 As Integer, NColumns1 As Integer
        Dim i As Integer, j As Integer
        NRows1 = UBound(RNames)
        NColumns1 = UBound(CNames)

        Debug.Write(" ")
        For j = 0 To NColumns1
            Debug.Write(CNames(j).PadLeft(10))
        Next

        Debug.WriteLine("")
        For i = 0 To NRows1
            Debug.Write(RNames(i).PadRight(8))
            For j = 0 To NColumns1
                Debug.Write(TheMatrix(i, j).ToString(".00000").PadLeft(10))
            Next
```

```
        Debug.WriteLine("")
    Next
  End Sub
End Module
```

4.5.2.2.2.89 NeedStandardErrors Method

Declares that the GetStandardErrors[278] method will be used to obtain bootstrap standard errors for the elements of a matrix of estimates.

Syntax

object.**NeedStandardErrors** (*matrixID*)

The **NeedStandardErrors** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.

Placement[156]: [1].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.

AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

You have to use **NeedStandardErrors** to declare that standard errors for a matrix will be needed before you can use GetStandardErrors 278 to obtain the standard errors. For example, you have to use

object.**NeedStandardErrors** (StandardizedTotalEffects)

before using

object.GetStandardErrors 278 (StandardizedTotalEffects, ...)

This example demonstrates the **NeedStandardErrors** method.

```
Imports System.Diagnostics
Imports AmosEngineLib.AmosEngine.TMatrixID
Imports Microsoft.VisualBasic
Module MainModule
    ' NeedStandardErrors Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.NeedStandardErrors(FactorScoreWeights)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Dim X(,) As Double
        Dim RNames() As String
        Dim CNames() As String

        'Get the row  and column variable names
        Sem.RowNames(FactorScoreWeights, RNames)
        Sem.ColumnNames(FactorScoreWeights, CNames)

        'Print the standard errors
        Sem.GetStandardErrors(FactorScoreWeights, X)
        Debug.WriteLine(vbCrLf & "Standard errors for factor score w eights")
        PrintMatrix(X, CNames, RNames)

        Sem.Dispose()
    End Sub

    'Print a matrix in the debug w indow
    Sub PrintMatrix(ByVal TheMatrix(,) As Double, ByVal CNames$(), ByVal RNames$())
        Dim NRow s1 As Integer, NColumns1 As Integer
        Dim i As Integer, j As Integer
        NRow s1 = UBound(RNames)
        NColumns1 = UBound(CNames)

        Debug.Write(" ")
        For j = 0 To NColumns1
            Debug.Write(CNames(j).PadLeft(10))
        Next

        Debug.WriteLine("")
        For i = 0 To NRow s1
            Debug.Write(RNames(i).PadRight(8))
            For j = 0 To NColumns1
                Debug.Write(TheMatrix(i, j).ToString(".00000").PadLeft(10))
            Next
            Debug.WriteLine("")
        Next
    End Sub
End Module
```

4.5.2.2.2.90 NonPositive Method

Controls whether Amos attempts to obtain maximum likelihood estimates when a sample covariance matrix is not positive definite.

Syntax

object.**NonPositive** ()
object.**NonPositive** (*tf*)

The **NonPositive** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. If *tf* is True (default), Amos attempts to obtain maximum likelihood estimates even when a sample covariance matrix is not positive definite.

Placement[156]: [1].

Default

Amos reports an error if you attempt a maximum likelihood analysis when a sample covariance matrix fails to be positive definite.

Remarks

When you use the **NonPositive** method, Amos does not try to test the hypothesis that your model is correct against the usual alternative that the population moments are unconstrained.

Wothke (1993)[528] discusses the problem of covariance matrices that are not positive definite.

See Also

AllowUnidentified Method[167]

This example demonstrates the **NonPositive** method.

```
Module MainModule
    ' NonPositive Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.NonPositive()

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.91 NormalityCheck Method

Controls the reporting of statistics for assessing multivariate normality of the observed variables.

Syntax

object.**NormalityCheck** ()
object.**NormalityCheck** (*tf*)

The **NormalityCheck** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. If *tf* is True (the default), statistics for assessing normality are reported.

Placement 156: [1].

Default

Statistics for assessing normality are not reported.

This example demonstrates the **NormalityCheck** method.

```
Module MainModule
    ' NormalityCheck Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.NormalityCheck()

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

In the example 341, **NormalityCheck** produces the following output.

Assessment of normality (Group number 1)

Variable	min	max	skew	c.r.	kurtosis	c.r.
wordmean	2.000	41.000	.575	2.004	-.212	-.370
sentence	4.000	28.000	-.836	-2.915	.537	.936
paragraph	2.000	19.000	.374	1.305	-.239	-.416
lozenges	3.000	36.000	.833	2.906	.127	.221
cubes	9.000	37.000	-.131	-.457	1.439	2.510
visperc	11.000	45.000	-.406	-1.418	-.281	-.490
Multivariate					3.102	1.353

The first row of the table shows that the lowest **wordmean** score was 2 and the highest was 41. **wordmean** had a *sample skewness* of

$$\frac{\sum_{i=1}^{N}\left(x_i-\bar{x}\right)^3}{N\hat{s}^3}=.575,$$

where $\hat{s}^2$ is the unbiased variance estimate $\hat{s}^2=\sum\left(x_i-\bar{x}\right)^2/(N-1)$. Assuming normality, skewness has a mean of zero and a standard error of $\sqrt{6/N}=.287$. The critical ratio 2.004 in the **c.r.** column is the sample skewness divided by its standard error.

wordmean has a *sample kurtosis* of

$$\frac{\sum_{i=1}^{N}(x_i-\bar{x})^4}{N\hat{s}^4}-3=-.212$$

.

Assuming normality, kurtosis has a mean of zero and a standard error of $\sqrt{24/N}=.573$. The critical ratio, –.370, is the sample kurtosis divided by its standard error.

The table has a separate row for each observed variable. A final row, labeled 'multivariate', contains Mardia's (Mardia, 1970 [520]; Mardia, 1974 [521]) coefficient of *multivariate kurtosis*

$$\frac{1}{N}\sum_{i=1}^{N}\left[(\mathbf{x}_i-\bar{\mathbf{x}})'\hat{\mathbf{S}}^{-1}(\mathbf{x}_i-\bar{\mathbf{x}})\right]^2-\frac{p(p+2)(N-1)}{N+1}=3.102$$

,

where $\mathbf{x}_i$ is the *i*-th observation on the *p* observed variables, $\bar{\mathbf{x}}$ is the vector of their means and $\hat{\mathbf{S}}^{-1}$ is the unbiased estimate of their population covariance matrix. Assuming normality, this coefficient has a mean of zero and a standard error of $\sqrt{8p(p+2)/N}=2.294$. The critical ratio obtained by dividing the sample coefficient by its standard error is 1.353, as shown in the **c.r.** column.

Assuming normality in very large samples, each of the critical values shown in the table above is an observation on a standard normally distributed random variable. Even with a very large sample, however, the table is of limited use. All it does is to quantify the departure from normality in the sample and provide a rough test of whether the departure is statistically significant. Unfortunately, this is not enough. In order to make use of this information you also need to know how robust your chosen estimation method is against the departure from normality that you have discovered. A departure from normality that is big enough to be significant could still be small enough to be harmless.

The following table, also produced by **NormalityCheck** from the **Grnt_fem** data, provides additional evidence on the question of normality.

Observations farthest from the centroid (Mahalanobis distance) (Group number 1)

Observation number	Mahalanobis d-squared	p1	p2
42	18.747	.005	.286
20	17.201	.009	.130
3	13.264	.039	.546
35	12.954	.044	.397
28	12.730	.048	.266

Only the first five rows of the table are shown here. Specifically, the table focuses on the occurrence of outliers, individual observations that differ markedly from the general run of observations. The table lists the observations that are furthest from the centroid of all observations, using as the distance measure for the *i*-th observation the squared Mahalanobis distance, $d_i^2=(\mathbf{x}_i-\bar{\mathbf{x}})'\hat{\mathbf{S}}^{-1}(\mathbf{x}_i-\bar{\mathbf{x}})$. Mardia's coefficient

of multivariate kurtosis can be written $\sum d_i^4/N - p(p+2)(N-1)/(N+1)$. The first row of the table shows that observation number 42 is furthest from the centroid with $d_{42}^2 = 18.747$. The **p1** column shows that, assuming normality, the probability of d_{42}^2 (or any individual d_i^2) exceeding 18.747 is .005. The **p2** column shows, still assuming normality, that the probability is .268 that the largest d_i^2 would exceed 18.747. The second row of the table shows that: Observation number 20 is the second furthest observation from the centroid with $d_{20}^2 = 17.201$. The probability of any arbitrary d_i^2 exceeding 17.201 is .009. The probability of the second largest d_i^2 exceeding 17.201 is .130. Small numbers in the **p1** column are to be expected. Small numbers in the **p2** column, on the other hand, indicate observations that are improbably far from the centroid under the hypothesis of normality. For the Grnt_fem data, none of the probabilities in the **p2** column is very small, so there is no evidence that any of the five most unusual observations should be treated as outliers under the assumption of normality. See Bollen (1987) [507] for a discussion of the importance of checking for outliers.

4.5.2.2.2.92 Npar Method

Gets the number of model parameters.

Syntax

object.**Npar** ()

The **Npar** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement [156]: [3].

Remarks

If you have used the Model [306] method to define more than one model, the **Npar** method returns the number of parameters for the most recently fitted model. The second example shows how to obtain the number of parameters for multiple models.

See Also

Df Method [221]

The following program shows how to display various fit measures when only one model is defined (i.e., when the Model [306] method has been used only once or not at all).

```
Imports System.Diagnostics
Module MainModule
    ' Npar Method Example 1
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Debug.WriteLine("Chi Square = " & Sem.Cmin)
        Debug.WriteLine("Degrees of Freedom = " & Sem.df)
        Debug.WriteLine("p = " & Sem.p)
        Debug.WriteLine("Number of parameters = " & Sem.npar)
        Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & _
        Sem.NcpLo & ", " & Sem.NcpHi & ")")
        Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
        Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)

        Sem.Dispose()
    End Sub
End Module
```

The following program shows how to display various fit measures for multiple models (i.e., when the Model 306 method has been used more than once).

```
Imports System.Diagnostics
Module MainModule
   ' Npar Method Example 2
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Dim i As Integer

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (a) spatial + (1) err_v")
      Sem.AStructure("cubes = (b) spatial + (1) err_c")
      Sem.AStructure("lozenges = (c) spatial + (1) err_l")
      Sem.AStructure("paragraph = (d) verbal + (1) err_p")
      Sem.AStructure("sentence = (e) verbal + (1) err_s")
      Sem.AStructure("w ordmean = (f) verbal + (1) err_w ")
      Sem.Var("spatial", 1)
      Sem.Var("verbal", 1)

      Sem.Model("Congeneric")
      Sem.Model("tau-equivalent", "a = b = c", "d = e = f")

      For i = 1 To 2
         Debug.WriteLine("")
         Debug.WriteLine("Model number " & i)
         Sem.FitModel(i)
         Debug.WriteLine("Chi Square = " & Sem.Cmin)
         Debug.WriteLine("Degrees of Freedom = " & Sem.df)
         Debug.WriteLine("p = " & Sem.p)
         Debug.WriteLine("Number of parameters = " & Sem.npar)
         Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & Sem.NcpLo & _
                  ", " & Sem.NcpHi & ")")
         Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
         Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)
      Next

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.93 NumberOfColumns Method

Gets the number of columns in a matrix of estimates.

Syntax

object.**NumberOfColumns** (*matrixID*)

object.**NumberOfColumns** (*matrixID, groupNumber*)

The **NumberOfColumns** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

matrixID	An integer that specifies a matrix of estimates, as described in Settings.
groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement 156: [3].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.

TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

In order to use **NumberOfColumns** (*matrixID*), you must first use NeedEstimates[329] (*matrixID*). For example, you have to use

object.NeedEstimates[329] (FactorScoreWeights)

before using

object.**NumberOfColumns** (FactorScoreWeights, …)

4.5.2.2.2.94 NumberOfGroups Method

Gets the number of groups.

Syntax

object.**NumberOfGroups** ()

The **NumberOfGroups** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement[156]: [3].

This example demonstrates the **NumberOfGroups** method.

```
Imports System.Diagnostics
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    ' NumberOfGroups Method Example
    Sub Main()
        Dim CNumbers() As Long, RNumbers() As Long, X() As Double
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.NeedEstimates(TotalEffects)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
        Sem.GroupName("girls")
        Sem.AStructure("academic = (g1) GPA + (g2) attract + (1) e1")
        Sem.AStructure("attract = (g3) height + (g4) weight + (g5) rating + (g6) academic + (1) e2")
        Sem.AStructure("e2 <--> e1")

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
        Sem.GroupName("boys")
        Sem.AStructure("academic = (b1) GPA + (b2) attract + (1) e1")
        Sem.AStructure("attract = (b3) height + (b4) weight + (b5) rating + (b6) academic + (1) e2")
        Sem.AStructure("e2 <--> e1")

        Debug.WriteLine("Number of groups = " & Sem.NumberOfGroups)

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.95 NumberOfParameters Method

Gets the number of model parameters, not taking into account any parameter constraints specified with the Model 306 method.

Syntax

object.**NumberOfParameters** ()

The **NumberOfVariables** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement 156: [3].

This example demonstrates the **NumberOfParameters** method.

```
Imports System.Diagnostics
Module MainModule
   ' NumberOfParameters Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Dim i As Integer

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = (a) spatial + (1) err_c")
      Sem.AStructure("lozenges = (b) spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = (c) verbal + (1) err_s")
      Sem.AStructure("w ordmean = (d) verbal + (1) err_w ")

      Debug.WriteLine("There are " & Sem.NumberOfParameters & " parameters")
      Debug.WriteLine("The 3rd parameter is called " & Sem.ParameterName(3))
      Debug.WriteLine("'a' is parameter number " & Sem.ParameterNumber("a"))
      Debug.WriteLine("The value of 'a' is " & Sem.ParameterValue("a"))
      Debug.WriteLine("The value of the 3rd parameter is " & Sem.ParameterValue(3))

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.96 NumberOfRow s Method

Gets the number of rows in a matrix of estimates.

Syntax

object.**NumberOfRows** (*matrixID*)

object.**NumberOfRows** (*matrixID, groupNumber*)

The **NumberOfRows** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings.
groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement[156]: [3].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

In order to use **NumberOfRows** (*matrixID*), you must first use NeedEstimates[329] (*matrixID*). For example, you have to use

object.NeedEstimates[329] (FactorScoreWeights)

before using

object.**NumberOfRows** (FactorScoreWeights, …)

4.5.2.2.2.97 NumberOfVariables Method

Gets the number of variables in the model for one group.

Syntax

object.**NumberOfVariables** ()
object.**NumberOfVariables** (*groupNumber*)

The **NumberOfVariables** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
groupNumber	Optional group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement[156]: [3].

This example demonstrates the **NumberOfVariables** method.

```
Imports System.Diagnostics
Module MainModule
    ' NumberOfVariables Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        'List all of the variables in the model
        Dim i As Integer
        For i = 1 To Sem.NumberOfVariables
            Debug.WriteLine(i.ToString.PadLeft(3) & " " & Sem.Variablename(i))
        Next

        'What is the variable number of "cubes"?
        Debug.WriteLine("")
        Debug.WriteLine("""cubes"" is variable number " & Sem.VariableNumber("cubes"))

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.98 OVariableCount Method

Specifies the number of observed variables in the model.

Syntax

object.**OVariableCount** (*nVariables*)

The **OVariableCount** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
nVariables	The number of observed variables in the model.

Placement[156]: [2].

Default

When the **OVariableCount** method is not used, no error checking is done based on the number of observed variables.

Remarks

Amos checks *nVariables* for consistency with the model and the data file. If a discrepancy is found, Amos reports the discrepancy and quits. Spelling or typing errors are frequently detected by this check, since two variant spellings of a variable name will be treated as references to two distinct variables.

In a multiple-group analysis, the **OvariableCount** method can be used once per group.

See Also

UVariableCount Method 404

VariableCount Method 407

This example demonstrates the **OVariableCount** method.

```
Module MainModule
    ' OVariableCount Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.OVariableCount(6)
        Sem.UVariableCount(8)
        Sem.VariableCount(14)

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.99 P Method

Gets the "*p* value" for testing the null hypothesis that the specified model is correct.

Syntax

object.**P** ()

The **P** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement 156: [3].

Remarks

If you have used the Model 306 method to define more than one model, the **P** method gets the *p* value for the most recently fitted model. The second example shows how to obtain the *p* value for multiple models.

The following program shows how to obtain various fit measures when only one model is defined (i.e., when the Model 306 method has been used only once or not at all).

```
Imports System.Diagnostics
Module MainModule
    ' P Method Example 1
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("wordmean = verbal + (1) err_w")

        Debug.WriteLine("Chi Square = " & Sem.Cmin)
        Debug.WriteLine("Degrees of Freedom = " & Sem.df)
        Debug.WriteLine("p = " & Sem.p)
        Debug.WriteLine("Number of parameters = " & Sem.npar)
        Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & _
            Sem.NcpLo & ", " & Sem.NcpHi & ")")
        Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
        Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)

        Sem.Dispose()
    End Sub
End Module
```

The following program shows how to obtain various fit measures for multiple models (i.e., when the Model 306 method has been used more than once).

```
Imports System.Diagnostics
Module MainModule
   ' P Method Example 2
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Dim i As Integer

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (a) spatial + (1) err_v")
      Sem.AStructure("cubes = (b) spatial + (1) err_c")
      Sem.AStructure("lozenges = (c) spatial + (1) err_l")
      Sem.AStructure("paragraph = (d) verbal + (1) err_p")
      Sem.AStructure("sentence = (e) verbal + (1) err_s")
      Sem.AStructure("w ordmean = (f) verbal + (1) err_w ")
      Sem.Var("spatial", 1)
      Sem.Var("verbal", 1)

      Sem.Model("Congeneric")
      Sem.Model("tau-equivalent", "a = b = c", "d = e = f")

      For i = 1 To 2
         Debug.WriteLine("")
         Debug.WriteLine("Model number " & i)
         Sem.FitModel(i)
         Debug.WriteLine("Chi Square = " & Sem.Cmin)
         Debug.WriteLine("Degrees of Freedom = " & Sem.df)
         Debug.WriteLine("p = " & Sem.p)
         Debug.WriteLine("Number of parameters = " & Sem.npar)
         Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & _
               Sem.NcpLo & ", " & Sem.NcpHi & ")")
         Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
         Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)
      Next

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.100 PackSymmetricEstimates Method

Controls whether the GetEstimates 260 returns symmetric matrices as square two-dimensional arrays or as one-dimensional arrays that contain only the lower triangle.

Syntax

object.**PackSymmetricEstimates** (*tf*)

The **PackSymmetricEstimates** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	If *tf* is False, symmetric matrices are returned as square two-dimensional arrays. If *tf* is True, symmetric matrices are returned as one-dimensional

	arrays, with the elements in the lower triangle stored in the order x(0,0), x(1,0), x(1,1),....

Placement[156]: [1].

Default

If the **PackSymmetricEstimates** method is not used, symmetric matrices are returned as square two-dimensional arrays.

Remarks

PackSymmetricEstimates does not affect GetEstimatesEx[265].

4.5.2.2.2.101 PageLength Method

The **PageLength** method has no effect. It is provided for compatibility with earlier versions of Amos.

Syntax

object.**PageLength** (*nLines*)

The **PageLength** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
nLines	The number of lines per page.

Placement[156]: [1].

4.5.2.2.2.102 Paginate Method

The **Paginate** method has no effect. It is provided for compatibility with earlier versions of Amos.

Syntax

object.**Paginate** (*tf*)

The **Paginate** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	If *tf* is True (default), the text output file is paginated. Otherwise, not.

Placement[156]: [1].

4.5.2.2.2.103 ParameterCovariance Method

Gets the covariance between two parameter estimates.

Syntax 1

result = *object*.**ParameterCovariance** (*parameterIndex*)
result = *object*.**ParameterCovariance** (*parameterIndex1*, *parameterIndex2*)
result = *object*.**ParameterCovariance** (*parameterName*)
result = *object*.**ParameterCovariance** (*parameterName1*, *parameterName2*)

The **ParameterCovariance** method syntax 1 has the following parts:

Part	Description
result	The covariance between two parameters or the variance of a single parameter.
object	An object of type **AmosEngine**.
parameterIndex *parameterIndex1* *parameterIndex2*	Parameter indices. Parameters are numbered starting with 1.
parameterName *parameterName1* *parameterName2*	Parameter names.

Placement[156]: [3].

See Also

ParameterName Method[361]

ParameterNumber Method[362]

ParameterValue Method[362]

The following program, based on Example 8, displays the error variances and their standard errors. Then it displays the correlation between two of the error variances.

```
Imports System
Imports System.Diagnostics
Module MainModule
    ' ParameterCovariance Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.Standardized()
        Sem.Smc()

        Sem.BeginGroup(AmosEngine.AmosDir & "\Examples\English\userguide.xls", "grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("wordmean = verbal + (1) err_w")
        'Give parameter names to the error variances.
        Sem.Var("err_v", "verr_v")
        Sem.Var("err_c", "verr_c")
        Sem.Var("err_l", "verr_l")
        Sem.Var("err_p", "verr_p")
        Sem.Var("err_s", "verr_s")
        Sem.Var("err_w", "verr_w")

        Debug.WriteLine("Error variances, and their standard deviations:")
        Debug.Write("verr_v: " & Sem.ParameterValue("verr_v"))
        Debug.WriteLine(Math.Sqrt(Sem.ParameterCovariance("verr_v")))
        Debug.Write("verr_c: " & Sem.ParameterValue("verr_c"))
        Debug.WriteLine(Math.Sqrt(Sem.ParameterCovariance("verr_c")))
        Debug.Write("verr_l: " & Sem.ParameterValue("verr_l"))
        Debug.WriteLine(Math.Sqrt(Sem.ParameterCovariance("verr_l")))
        Debug.Write("verr_p: " & Sem.ParameterValue("verr_p"))
        Debug.WriteLine(Math.Sqrt(Sem.ParameterCovariance("verr_p")))
        Debug.Write("verr_s: " & Sem.ParameterValue("verr_s"))
        Debug.WriteLine(Math.Sqrt(Sem.ParameterCovariance("verr_s")))
        Debug.Write("verr_w: " & Sem.ParameterValue("verr_w"))
        Debug.WriteLine(Math.Sqrt(Sem.ParameterCovariance("verr_w")))
        Debug.WriteLine("")

        Debug.Write("The correlation between ""verr_v"" and ""verr_c"" is ")
        Dim Numerator As Double, Denominator As Double
        Numerator = sem.ParameterCovariance("verr_v", "verr_c")
        Denominator = Math.Sqrt(Sem.ParameterCovariance("verr_v") * _
                Sem.ParameterCovariance("verr_c"))
        Debug.WriteLine(Numerator / Denominator)

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.104 ParameterInfo Method

Get information about the parameter that is specified by an index number, based on an arbitrary ordering of model parameters. The first parameter is parameter number 1.

Syntax

object.**ParameterInfo** (*parameterNumber, groupNumber, parameterType, leftVariable, rightVariable*)

The **ParameterInfo** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
parameterNumber	Input: A whole number between 1 and n, where n is the number of parameters.
groupNumber	Output: When the parameter specified by *parameterNumber* is associated with one group only, then *groupNumber* is that group's number. The first group is group number 1.
parameterType	Output: An integer that tells what kind of parameter is specified by *parameterNumber*. The possible values are 1 (regression weight), 2 (mean), 3 (intercept), 4 (covariance) and 5 (variance).
leftVariable, rightVariable	Integers that specify two variables. (Use the VariableName 408 method to get their names.) If the parameter specified by *parameterNumber* is a regression weight, *leftVariable* refers to the dependent variable, *rightVariable* to the independent variable. If the parameter specified by *parameterNumber* is a covariance, it is the covariance between *leftVariable* and *rightVariable*. If the parameter specified by *parameterNumber* is a variance, it is the variance of *leftVariable*. (*rightVariable* is the same as *leftVariable*.) If the parameter specified by *parameterNumber* is a mean, it is the mean of *leftVariable*. (The value returned for *rightVariable* is undefined.)

Placement 156: [3].

See Also

ParameterName Method 361

ParameterNumber Method 362

ParameterValue Method 362

The following program fits the model of Example 8. Then it displays a list of the model parameters.

```
Imports System.Diagnostics
Module MainModule
    ' ParameterInfo Method Example
    Dim Sem As New AmosEngineLib.AmosEngine
    Sub Main()
        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\Grnt_fem.sav")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragrap = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Dim i As Integer
        For i = 1 To Sem.NumberOfParameters
            Debug.WriteLine(i & " " & ParameterPicture(i))
        Next

        Sem.Dispose()
    End Sub

    Function ParameterPicture(ByVal ParameterNumber As Integer) As String
        Dim GroupNumber As Integer
        Dim ParameterType As Integer
        Dim LeftVariable As Integer
        Dim RightVariable As Integer
        Dim S As String
        Dim vname1 As String
        Dim vname2 As String

        Sem.ParameterInfo(ParameterNumber, _
                GroupNumber, ParameterType, _
                LeftVariable, RightVariable)

        vname1 = Sem.Variablename(LeftVariable)
        vname2 = Sem.Variablename(RightVariable)
        Select Case ParameterType
            Case 1
                S = vname1 & "<--" & vname2
            Case 2
                S = vname1 & " mean"
            Case 3
                S = vname1 & " intercept"
            Case 4
                S = vname1 & "<->" & vname2
            Case 5
                S = vname1 & " variance"
        End Select
        If Sem.NumberOfGroups > 1 Then
            S = "Group " & GroupNumber & " " & S
        End If
        S = "[" & S & "]"
        ParameterPicture = S
    End Function
End Module
```

4.5.2.2.2.105 ParameterName Method

Gets the name of a parameter, given its parameter number.

Syntax

object.**ParameterName** (*parameterNumber*)

The **ParameterName** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
parameterNumber	A parameter number.

Placement[156]: [3].

See Also

ParameterNumber Method[362]

This example demonstrates the **ParameterName** method.

```
Imports System.Diagnostics
Module MainModule
   ' ParameterName Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Dim i As Integer

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = (a) spatial + (1) err_c")
      Sem.AStructure("lozenges = (b) spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = (c) verbal + (1) err_s")
      Sem.AStructure("w ordmean = (d) verbal + (1) err_w ")

      Debug.WriteLine("There are " & Sem.NumberOfParameters & " parameters")
      Debug.WriteLine("The 3rd parameter is called " & Sem.ParameterName(3))
      Debug.WriteLine("'a' is parameter number " & Sem.ParameterNumber("a"))
      Debug.WriteLine("The value of 'a' is " & Sem.ParameterValue("a"))
      Debug.WriteLine("The value of the 3rd parameter is " & Sem.ParameterValue(3))

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.106 ParameterNumber Method

Gets a parameter number, given its name.

A "parameter number" is a parameter's position on Amos's internal parameter list. The first parameter on the list is parameter number 1. Some methods refer to model parameters by number (*i.e.*, by list position).

Syntax

object.**ParameterNumber** (*ParameterName*)

The **ParameterNumber** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
ParameterName	A parameter name.

Placement[156]: [3].

This example demonstrates the **ParameterNumber** method.

```
Imports System.Diagnostics
Module MainModule
    ' ParameterNumber Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Dim i As Integer

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = (a) spatial + (1) err_c")
        Sem.AStructure("lozenges = (b) spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = (c) verbal + (1) err_s")
        Sem.AStructure("w ordmean = (d) verbal + (1) err_w ")

        Debug.WriteLine("There are " & Sem.NumberOfParameters & " parameters")
        Debug.WriteLine("The 3rd parameter is called " & Sem.ParameterName(3))
        Debug.WriteLine("'a' is parameter number " & Sem.ParameterNumber("a"))
        Debug.WriteLine("The value of 'a' is " & Sem.ParameterValue("a"))
        Debug.WriteLine("The value of the 3rd parameter is " & Sem.ParameterValue(3))

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.107 ParameterValue Method

Gets a parameter value, given either a parameter name or a parameter number.

Syntax

object.**ParameterValue** (*parameterIndex*)

object.**ParameterValue** (*parameterName*)

The **ParameterValue** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
parameterIndex	(Integer) A parameter index. Parameters are numbered starting with 1.
parameterName	(String) A parameter name.

Placement 156: [3].

See Also

ParameterName Method 361

ParameterNumber Method 362

This example demonstrates the **ParameterValue** method.

```
Imports System.Diagnostics
Module MainModule
   ' ParameterValue Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Dim i As Integer

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = (a) spatial + (1) err_c")
      Sem.AStructure("lozenges = (b) spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = (c) verbal + (1) err_s")
      Sem.AStructure("w ordmean = (d) verbal + (1) err_w ")

      Debug.WriteLine("There are " & Sem.NumberOfParameters & " parameters")
      Debug.WriteLine("The 3rd parameter is called " & Sem.ParameterName(3))
      Debug.WriteLine("'a' is parameter number " & Sem.ParameterNumber("a"))
      Debug.WriteLine("The value of 'a' is " & Sem.ParameterValue("a"))
      Debug.WriteLine("The value of the 3rd parameter is " & Sem.ParameterValue(3))

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.108 ParameterVector Method

Gets the values of all free parameters.

Syntax

object.**ParameterVector** (*x*)

The **ParameterVector** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
x	An array of type double dimensioned in the calling program as **Dim x() as Double** On return from **ParameterVector**, *x* is a 1-dimensional array containing the vector of free parameters.

Placement[156]: [3].

See Also

Use the AmosEngine class to evaluate derivatives numerically and display the results with the Amos Debug class[499]

4.5.2.2.2.109 Path Method

Specifies a regression weight as a model parameter.

Syntax

object.**Path** (*leftVariableName*, *rightVariableName*)

object.**Path** (*leftVariableName*, *rightVariableName*, *parameterValue*)

object.**Path** (*leftVariableName*, *rightVariableName*, *parameterName*)

The **Path** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
leftVariableName *rightVariableName*	The character strings, *leftVariableName* and *rightVariableName*, are the names of two variables. *leftVariableName* depends linearly on *rightVariableName*.

	That is, *leftVariableName* is on the left-hand side of a regression equation and *rightVariableName* appears on the right-hand side with a regression weight that is a model parameter.
parameterValue	(Optional) Parameter value. If *parameterValue* = 3, say, then the regression weight is fixed at 3.
parameterName	(Optional) Parameter name. If parameterName = "abc", say, then the regression weight is named "abc". It is constrained to be equal to any other parameters named "abc". If parameterName = "3?", say then the regression weight is given an initial value of 3, and is unconstrained. If parameterName = "abc:3", say, then the regression weight is named "abc" and is given an initial value of 3. It is constrained to be equal to any other parameters named "abc". If parameterName is an empty string (""), the regression weight is an unconstrained parameter.

Placement 156: [2].

Default

A variable is assumed not to depend directly on another variable unless a linearly dependency is specified by use of the **Path** or AStructure 169 method.

Remarks

If *parameterValue* and *parameterName* are omitted, the regression weight is an unconstrained parameter.

The following program uses the **Path**, Cov 212 and Var 405 methods to specify Model C of Example 6.

```
Module MainModule
    ' Path Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.TextOutput()
        Sem.Standardized()
        Sem.Smc()
        Sem.AllImpliedMoments()
        Sem.FactorScoreWeights()
        Sem.TotalEffects()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Wheaton")
        Sem.Path("anomia67", "67_alienation", 1)
        Sem.Path("anomia67", "eps1", 1)
        Sem.Path("powles67", "67_alienation", "path_p")
        Sem.Path("powles67", "eps2", 1)
        Sem.Path("anomia71", "71_alienation", 1)
        Sem.Path("anomia71", "eps3", 1)
        Sem.Path("powles71", "71_alienation", "path_p")
        Sem.Path("powles71", "eps4", 1)
        Sem.Path("67_alienation", "ses")
        Sem.Path("67_alienation", "zeta1", 1)
        Sem.Path("71_alienation", "67_alienation")
        Sem.Path("71_alienation", "ses")
        Sem.Path("71_alienation", "zeta2", 1)
        Sem.Path("education", "ses", 1)
        Sem.Path("education", "delta1", 1)
        Sem.Path("SEI", "ses")
        Sem.Path("SEI", "delta2", 1)
        Sem.Cov("eps3", "eps1")
        Sem.Var("eps1", "var_a")
        Sem.Var("eps2", "var_p")
        Sem.Var("eps3", "var_a")
        Sem.Var("eps4", "var_p")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.110 Pclose Method

Gets the "*p* value" for testing the null hypothesis that RMSEA is less than .05 in the population. (Browne & Cudeck, 1993 509)

Syntax

object.**Pclose** ()

The **Pclose** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement[156]: [3].

Remarks

If you have used the Model[306] method to define more than one model, the **Pclose** method gets the *p* value for the most recently fitted model. The second example shows how to obtain the *p* value for multiple models.

The following program shows how to display various fit measures when only one model is defined (i.e., when the Model[306] method has been used only once or not at all).

```
Imports System.Diagnostics
Module MainModule
    ' Pclose Method Example 1
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Debug.WriteLine("Chi Square = " & Sem.Cmin)
        Debug.WriteLine("Degrees of Freedom = " & Sem.df)
        Debug.WriteLine("p = " & Sem.p)
        Debug.WriteLine("Number of parameters = " & Sem.npar)
        Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & _
                Sem.NcpLo & ", " & Sem.NcpHi & ")")
        Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
        Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)

        Sem.Dispose()
    End Sub
End Module
```

The following program shows how to display various fit measures for multiple models (i.e., when the Model[306] method has been used more than once).

```
Imports System.Diagnostics
Module MainModule
   ' Pclose Method Example 2
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Dim i As Integer

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (a) spatial + (1) err_v")
      Sem.AStructure("cubes = (b) spatial + (1) err_c")
      Sem.AStructure("lozenges = (c) spatial + (1) err_l")
      Sem.AStructure("paragraph = (d) verbal + (1) err_p")
      Sem.AStructure("sentence = (e) verbal + (1) err_s")
      Sem.AStructure("w ordmean = (f) verbal + (1) err_w ")
      Sem.Var("spatial", 1)
      Sem.Var("verbal", 1)

      Sem.Model("Congeneric")
      Sem.Model("tau-equivalent", "a = b = c", "d = e = f")

      For i = 1 To 2
         Debug.WriteLine("")
         Debug.WriteLine("Model number " & i)
         Sem.FitModel(i)
         Debug.WriteLine("Chi Square = " & Sem.Cmin)
         Debug.WriteLine("Degrees of Freedom = " & Sem.df)
         Debug.WriteLine("p = " & Sem.p)
         Debug.WriteLine("Number of parameters = " & Sem.npar)
         Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & _
                  Sem.NcpLo & ", " & Sem.NcpHi & ")")
         Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
         Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)
      Next

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.111 Permute Method

Performs a permutation test (Arbuckle, 1994b [506]) of the specified model.

Syntax

object.**Permute** (*x*)

The **Permute** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
x	An integer. If *x* = 0, the permutation test is based on all permutations of the observed variables. If *x* is positive, the permutation test is based on *x* random permutations. If *x* is negative, no permutation test is performed.

Placement[156]: [1].

Default

No permutation test is performed.

Remarks

Bootstrapping cannot be performed at the same time as the permutation test. That is, you can't execute both the **Permute** and the Bootstrap[188] methods in the same program.

See Also

PermuteDetail Method[371]

The following program performs a permutation test of Jöreskog and Sörbom's (1989, p. 205)[517] Model D for the data of Wheaton et al. (1977)[527].

```
Module MainModule
    ' Permute Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.TextOutput()
        Sem.Iterations(100)
        Sem.Permute(0) 'All permutations
        Sem.PermuteDetail()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Wheaton")
        Sem.Path("anomia67", "67_alienation", 1)
        Sem.Path("anomia67", "eps1", 1)
        Sem.Path("pow les67", "67_alienation")
        Sem.Path("pow les67", "eps2", 1)
        Sem.Path("anomia71", "71_alienation", 1)
        Sem.Path("anomia71", "eps3", 1)
        Sem.Path("pow les71", "71_alienation")
        Sem.Path("pow les71", "eps4", 1)
        Sem.Path("67_alienation", "ses")
        Sem.Path("67_alienation", "zeta1", 1)
        Sem.Path("71_alienation", "67_alienation")
        Sem.Path("71_alienation", "ses")
        Sem.Path("71_alienation", "zeta2", 1)
        Sem.Path("education", "ses", 1)
        Sem.Path("education", "delta1", 1)
        Sem.Path("SEI", "ses")
        Sem.Path("SEI", "delta2", 1)
        Sem.Var("eps1", "var1")
        Sem.Var("eps2", "var2")
        Sem.Var("eps3", "var3")
        Sem.Var("eps4", "var4")
        Sem.Cov("eps1", "eps3", "cov1")
        Sem.Cov("eps2", "eps4", "cov2")

        Sem.Dispose()
    End Sub
End Module
```

Here is a portion of the output from the example.

```
Matrix Permutations Test (Model 1)
Summary (Model 1)
Of 719 permutations:
15 permutations improved the model fit or left it unchanged.
86 permutations resulted in a model that could not be fitted.
618 permutations resulted in a higher discrepancy function.

Of the remaining permutations:
0 resulted in inadmissible estimates and unstable systems.
0 resulted in inadmissible estimates.
0 resulted in unstable systems.
p = 16 / 720 = .022
```

With six observed variables, there are 720 possible permutations -- 719, if you don't count the permutation that leaves each observed variable in its original position. Of the 719 non-identity permutations, 15 made the discrepancy function smaller or left it unchanged. 617 of the permutations made the discrepancy function larger. 86 permutations resulted in a model for which Amos could not find a solution. As noted above, failures are to be expected in fitting a series of generally bad models. The question is, how do you classify the models for which no solution was found? Can it be assumed that each one of those models is worse than the original model? In other words, can you assume that, whenever Amos fails, it's the model's fault rather than Amos's?

Experience shows that Amos's failures to find solutions are *almost* always due to bad models (or samples that are too small). But not always. Therefore, there may be an objection to lumping the 86 permutations that produced an unfittable model together with the 617 permutations that produced a worse fitting model, on the grounds that doing so could result in an overcount of the number of permutations that make the model worse.

With these considerations in mind, Amos follows the convention that unfittable models are "worse than" the model being evaluated. Then out of 720 permutations (including the identity permutation), there are 16 permutations that produce a model that fits as well as or better than the original model. (The original model itself is one of those 16.). In other words, if you picked a model at random out of those generated by permuting the observed variables, there is a probability of 16/720 = .022 of getting a model as good as the one that Jöreskog and Sörbom proposed.

It is possible for an Amos solution to be inadmissible or to consist of an unstable linear system, although neither of these problems arose in the present example. There needs to be a policy on permutations that produce a model with a lower discrepancy function than was obtained for the original model, but for which an inadmissible solution or an unstable system occurs. Amos adheres to the following policy. First of all, if the original model results in an inadmissible solution, Amos disregards the admissibility status of estimates for models that are generated by permutations. Also, if the original model results in an unstable system, Amos ignores any instability that occurs in linear systems that result from permutations. If the original model yields an admissible solution with a stable system of linear equations, Amos reports the number of permutations that lower the discrepancy function while producing an inadmissible solution or an unstable system, and follows the convention that such permutations are harmful (*i.e.*, that they make a model worse).

The frequency of inadmissible solutions and unstable systems is summarized as follows for the present example.

Of the remaining permutations:
0 resulted in inadmissible estimates and unstable systems.
0 resulted in inadmissible estimates.
0 resulted in unstable systems.

Of the 15 permutations that resulted in a discrepancy function that was as good as or better than that of the original model, all were in fact exactly as good - none were better. Examination of the output from the PermuteDetail 371 method reveals that these 15 models are equivalent to the original model in the sense of Stelzl (1986) 526, Lee and Hershberger (1990) 518 and MacCallum, et al. (1993) 520.

In principal, it would be possible to reduce the computational requirements of the permutation test by fitting one representative model from each set of equivalent models. Amos does not do this, however. More importantly, the fact that the "permuted" models come in clusters of equivalent models has a bearing on the interpretation of the permutation test. In the current example, for instance, the proportion of permuted models that fit as well as or better than the original model cannot take on just any of the values 1/720, 2/720, 3/720,.... Instead, the proportion is restricted to the values 16/720, 32/720, 48/720,.... The number of possible *p* values is still 720/16 = 45, and so it remains an interesting question what the value of *p* is. However, a serious problem arises when the number of permutations that leave the fit of the model invariant is very large, so that the number of distinct discrepancy function values that can occur is very small. To take an extreme case, consider the common factor model with one common factor, and no parameter constraints other than those required to make the model identified. No permutation of the observed variables will affect the fit of the model, and it will not be possible to apply the permutation test in a meaningful way.

4.5.2.2.2.112 PermuteDetail Method

Gives detailed information about the solution obtained for each permutation when the Permute 368 method is used. First the permutation itself is reported. (That is, the new location of each observed variable in the model is shown after the permutation is carried out.) Then, if a solution is found, the value of the discrepancy function is reported along with a notation of whether the solution was admissible and whether the resulting linear system was stable.

Syntax

object.**PermuteDetail** ()
object.**PermuteDetail** (*tf*)

The **PermuteDetail** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

tf	Optional. A boolean value that controls the reporting of information about each permutation. True (default) requests the output. False suppresses it

Placement[156]: [1].

Default

Detailed information about each permutation is not reported.

The following program performs a permutation test of Jöreskog and Sörbom's (1989[517], p. 205) Model D for the data of Wheaton et al. (1977)[527].

```
Module MainModule
   ' PermuteDetail Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Sem.TextOutput()
      Sem.Iterations(100)

      Sem.Permute(0) 'All permutations
      Sem.PermuteDetail()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Wheaton")
      Sem.Path("anomia67", "67_alienation", 1)
      Sem.Path("anomia67", "eps1", 1)
      Sem.Path("pow les67", "67_alienation")
      Sem.Path("pow les67", "eps2", 1)
      Sem.Path("anomia71", "71_alienation", 1)
      Sem.Path("anomia71", "eps3", 1)
      Sem.Path("pow les71", "71_alienation")
      Sem.Path("pow les71", "eps4", 1)
      Sem.Path("67_alienation", "ses")
      Sem.Path("67_alienation", "zeta1", 1)
      Sem.Path("71_alienation", "67_alienation")
      Sem.Path("71_alienation", "ses")
      Sem.Path("71_alienation", "zeta2", 1)
      Sem.Path("education", "ses", 1)
      Sem.Path("education", "delta1", 1)
      Sem.Path("SEI", "ses")
      Sem.Path("SEI", "delta2", 1)
      Sem.Var("eps1", "var1")
      Sem.Var("eps2", "var2")
      Sem.Var("eps3", "var3")
      Sem.Var("eps4", "var4")
      Sem.Cov("eps1", "eps3", "cov1")
      Sem.Cov("eps2", "eps4", "cov2")

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.113 PutParameterValue Method

Assigns a value to a parameter.

Syntax

object.**PutParameterValue** (*parameterIndex, parameterValue*)

object.**PutParameterValue** (*parameterName, parameterValue*)

The **PutParameterValue** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
parameterIndex	(Integer) A parameter index. Parameters are numbered starting with 1.
parameterName	(String) A parameter name.
parameterValue	(Double) New parameter value.

Placement 156: [3].

See Also

Use the AmosEngine class to evaluate derivatives numerically and display the results with the Amos Debug class 499

4.5.2.2.2.114 PutParameterVector Method

Assigns values to the vector of free parameters.

Syntax

object.**PutParameterVector** (*x*)

The **PutParameterVector** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
x	A 1-dimensional array of type double. *x* must be dimensioned to have exactly one element for each parameter. The first parameter is in *x*(0). (The number of parameters can be obtained with the NumberOfParameters 348 method.)

Placement 156: [3].

See Also

Use the AmosEngine class to evaluate derivatives numerically and display the results with the Amos Debug class [499]

4.5.2.2.2.115 PutSampleCovariances Method

Assigns values to the matrix of sample covariances. This method should not be used if means/intercepts are explicit model parameters, because it sets the vector of sample means to zero. (Use PutSampleMoments [375] instead.)

Syntax

object.**PutSampleCovariances** (*covarianceMatrix*)
object.**PutSampleCovariances** (*covarianceMatrix, groupNumber*)

The **PutSampleCovariances** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
covarianceMatrix	A two-dimensional array of type double. The dimensions of *covarianceMatrix* must exactly match the dimensions of the sample covariance matrix.
groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement [156]: [3].

Remarks

One technique for performing sampling studies consists of the repeated use of **PutSampleCovariances** and FitModel [237].

See Also

Use the AmosEngine class to test for scale- and location-invariance [501]

4.5.2.2.2.116 PutSampleCovariancesPacked Method

Assigns values to the matrix of sample covariances, where sample covariances are stored as a one-dimensional array. This method should not be used if means/intercepts are explicit model parameters, because it sets the vector of sample means to zero. (Use PutSampleMomentsPacked [376] instead.) If your sample covariances are stored in a two-dimensional array, use PutSampleCovariances [374].

Syntax

object.**PutSampleCovariancesPacked** (*covarianceMatrix*)

object.**PutSampleCovariancesPacked** (*covarianceMatrix, groupNumber*)

The **PutSampleCovariancesPacked** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
covarianceMatrix	A one-dimensional array of type double. The lower triangular elements of *covarianceMatrix* should be stored in the order x(0,0), x(1,0), x(1,1),.....
groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement 156: [3].

Remarks

One efficient technique for performing a sampling study is to loop through the following sequence.

1. Generate a new sample covariance matrix.
2. Use **PutSampleCovariancesPacked**.
3. Use FitModel 237.

4.5.2.2.2.117 PutSampleMoments Method

Assigns values to the sample covariances and the sample means.

Syntax

object.**PutSampleMoments** (*covariancesBase0, meansBase0*)
object.**PutSampleMoments** (*covariancesBase0, meansBase0, groupNumber*)

The **PutSampleMoments** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
covariancesBase0	A two-dimensional array of type double. The dimensions of *covariancesBase0* must exactly match the dimensions of the sample covariance matrix.
meansBase0	A one-dimensional array of type double. The number of elements in *meansBase0* must equal the number of observed variables in the model.

groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement[156]: [3].

Remarks

One technique for performing sampling studies consists of the repeated use of **PutSampleMoments** and FitModel[237].

See Also

Use the AmosEngine class to test for scale- and location-invariance[501]

4.5.2.2.2.118 PutSampleMomentsPacked Method

Assigns values to the sample covariances and the sample means when sample covariances are stored as a one-dimensional array.

Syntax

object.**PutSampleMomentsPacked** (*covariances*, *means*)
object.**PutSampleMomentsPacked** (*covariances*, *means*, *groupNumber*)

The **PutSampleMomentsPacked** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
covariances	A one-dimensional array of type double. The lower triangular elements of *covariances* should be stored in the order x(0,0), x(1,0), x(1,1),.....
means	A one-dimensional array of type double. The number of elements in *means* must equal the number of observed variables in the model.
groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement[156]: [3].

Remarks

One efficient technique for performing a sampling study is to loop through the following sequence.

1. Generate a new sample covariance matrix.

2. Use **PutSampleMomentsPacked**.
3. Use FitModel [237].

4.5.2.2.2.119 ResidualMoments Method

Controls reporting of the difference between the sample covariance matrix and the implied covariance matrix. If means and intercepts are explicitly modeled, the **ResidualMoments** method also controls reporting of differences between sample means and implied means.

Syntax

object.**ResidualMoments** ()
object.**ResidualMoments** (*tf*)

The **ResidualMoments** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. A boolean value that controls the reporting of residual moments. True (default) requests the output. False suppresses it.

Placement [156]: [1].

Default

Residual moments are not reported.

See Also

Admissible Method [164]

ImpliedMoments Method [286]

SampleMoments Method [389]

This example demonstrates the **ResidualMoments** method.

```
Module MainModule
    ' ResidualMoments Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.ResidualMoments()
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.120 ReviseModel Method

Changes the constraints on a model that was previously defined by the Model 306 method. There are some limitations on the extent to which a model can be changed.

Syntax

object.**ReviseModel** (*modelName*, *constraint1*)
object.**ReviseModel** (*modelName*, *constraint1*, *constraint2*)
object.**ReviseModel** (*modelName*, *constraint1*, *constraint2*, *constraint3*)
...

The **ReviseModel** method syntax is identical to the Model 306 method syntax. The **ReviseModel** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
modelName	The name of a model previously defined by the Model 306 method.
constraint1	Either: A string of the form p1=p2=p3=..., where each pi is either a parameter name or a number. At most one of the pi can be a number. Or: The name of another model.

constraint2	Same as constraint1
constraint3	Same as constraint1
...	...

Placement[156]: [3].

Remarks

The constraints expressed by *constraint1*, *constraint2*,... replace the constraints of the original model. There are the following limitations on the changes model changes that can be made by **ReviseModel**.

The number of equality constraints in the revised model cannot exceed the number of constraints in the original model. For example, the **ReviseModel** method in following program fragment raises an error because it attempts to create four constraints on a model that originally had only three.

```
…
Dim Sem As New AmosEngine
…
Sem.Model "Model 1", "a=0", "b=0", "c=0"
…
Sem.ReviseModel "Model 1", "u=v=w =x=0"
…
```

The constraints in the revised model may only employ constants of 0, 1, or constants that have been previously been assigned to parameter values with the Model[306], AStructure[169], MStructure[315], Path[364], Cov[212], Var[405], Mean[303], or Intercept[299] methods.

The following program performs an all-possible-subsets regression analysis with the model of Example 4.

```
Imports System.Diagnostics
Module MainModule
    ' ReviseModel Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Warren5v")
        Sem.AStructure("performance = (b1) know ledge + (b2) value + (b3) satisfaction + (1) error")

        'Define a model w ith three constraints.
        'Subsequently, no revised model can have more than three constraints.
        Sem.Model("Model 1", "b1 = b2 = b3 = 0")
        Subset(Sem, "b1 = b2 = b3 = 0", "no predictors")
        Debug.WriteLine("--------------------------------")
        Subset(Sem, "b1 = b2 = 0", "satisfaction")
        Subset(Sem, "b2 = b3 = 0", "know ledge")
        Subset(Sem, "b1 = b3 = 0", "value")
        Debug.WriteLine("--------------------------------")
        Subset(Sem, "b1 = 0", "value, satisfaction")
        Subset(Sem, "b2 = 0", "know ledge, satisfaction")
        Subset(Sem, "b3 = 0", "know ledge, value")
        Debug.WriteLine("--------------------------------")
        Subset(Sem, "", "know ledge, value, satisfaction")

        Sem.Dispose()
    End Sub

    Sub Subset(ByVal Sem As AmosEngineLib.AmosEngine, ByVal Constraints As String, ByVal Predictors As String)
        Sem.ReviseModel("Model 1", Constraints)
        If Sem.FitModel = 0 Then
            Debug.Write(Sem.Cmin)
        Else
            Debug.Write("Failed")
        End If
        Debug.WriteLine(" " & Predictors)
    End Sub
End Module
```

4.5.2.2.2.121 Rmsea, RmseaLo, RmseaHi Methods

Gets a point estimate of the root mean square error of approximation (RMSEA). **RmseaLo** and **RmseaHi** get the lower and upper boundaries of a 90% confidence interval for the RMSEA.

Syntax

object.**Rmsea** ()
object.**RmseaLo** ()
object.**RmseaHi** ()

The **Rmsea, RmseaLo** and **RmseaHi** method syntaxes have the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement 156: [3].

Remarks

If you have used the Model 306 method to define more than one model, the **Rmsea**, **RmseaLo** and **RmseaHi** methods return estimates for the most recently fitted model. The second example shows how to obtain estimates for multiple models.

The following program shows how to display various fit measures when only one model is defined (i.e., when the Model 306 method has been used only once or not at all).

```
Imports System.Diagnostics
Module MainModule
    ' Rmsea, RmseaLo and RmseaHi Methods Example 1
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Debug.WriteLine("Chi Square = " & Sem.Cmin)
        Debug.WriteLine("Degrees of Freedom = " & Sem.df)
        Debug.WriteLine("p = " & Sem.p)
        Debug.WriteLine("Number of parameters = " & Sem.npar)
        Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & _
                Sem.NcpLo & ", " & Sem.NcpHi & ")")
        Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
        Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)

        Sem.Dispose()
    End Sub
End Module
```

The following program shows how to obtain various fit measures for multiple models (i.e., when the Model 306 method has been used more than once).

```
Imports System.Diagnostics
Module MainModule
    ' Rmsea, RmseaLo and RmseaHi Methods Example 2
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Dim i As Integer

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (a) spatial + (1) err_v")
        Sem.AStructure("cubes = (b) spatial + (1) err_c")
        Sem.AStructure("lozenges = (c) spatial + (1) err_l")
        Sem.AStructure("paragraph = (d) verbal + (1) err_p")
        Sem.AStructure("sentence = (e) verbal + (1) err_s")
        Sem.AStructure("w ordmean = (f) verbal + (1) err_w ")
        Sem.Var("spatial", 1)
        Sem.Var("verbal", 1)

        Sem.Model("Congeneric")
        Sem.Model("tau-equivalent", "a = b = c", "d = e = f")

        For i = 1 To 2
            Debug.WriteLine("")
            Debug.WriteLine("Model number " & i)
            Sem.FitModel(i)
            Debug.WriteLine("Chi Square = " & Sem.Cmin)
            Debug.WriteLine("Degrees of Freedom = " & Sem.df)
            Debug.WriteLine("p = " & Sem.p)
            Debug.WriteLine("Number of parameters = " & Sem.npar)
            Debug.WriteLine("Noncentrality parameter = " & Sem.Ncp & " (" & Sem.NcpLo & ", " & Sem.NcpHi & ")")
            Debug.WriteLine("Rmsea = " & Sem.Rmsea & " (" & Sem.RmseaLo & ", " & Sem.RmseaHi & ")")
            Debug.WriteLine("Test of close fit, p = " & Sem.Pclose)
        Next

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.122 Row Names Method

Obtains the variable names associated with the rows of a matrix of estimates.

Syntax

object.**RowNames** (*matrixID*, *theRowNames*)
object.**RowNames** (*matrixID*, *theRowNames*, *groupNumber*)

The **RowNames** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings

theRowNames	A string array declared in the calling program as **Dim theRowNames() as String** The Amos Engine will redimension the array.
groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement[156]: [3].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.
ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.

IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

The NeedEstimates 329 method must be used to give notice that a particular matrix of estimates will be needed before the **RowNames** method can be used to obtain the row names for that matrix. For example, you have to use

object.NeedEstimates 329 (ImpliedMeans)

before using

object.**RowNames** (ImpliedMeans, ...)

The following program fits Models A and B of Example 11. It displays the matrix of total effects for each group and model.

```
Imports System.Diagnostics
Imports AmosEngineLib.AmosEngine.TMatrixID
Imports Microsoft.VisualBasic
Module MainModule
    ' Row Names Method Example
    Sub Main()
        Dim CNames() As String, RNames() As String, X(,) As Double
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.NeedEstimates(TotalEffects)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
        Sem.GroupName("girls")
        Sem.AStructure("academic = (g1) GPA + (g2) attract + (1) e1")
        Sem.AStructure( _
        "attract = (g3) height + (g4) w eight + (g5) rating + (g6) academic + (1) e2")
        Sem.AStructure("e2 <--> e1")
        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
        Sem.GroupName("boys")
        Sem.AStructure("academic = (b1) GPA + (b2) attract + (1) e1")
        Sem.AStructure( _
        "attract = (b3) height + (b4) w eight + (b5) rating + (b6) academic + (1) e2")
        Sem.AStructure("e2 <--> e1")

        Sem.Model("Model_A")
        Sem.Model("Model_B", "g1=b1", "g2=b2", "g3=b3", "g4=b4", "g5=b5", "g6=b6")

        'Print total effects for each model and each group
        Dim ModelNumber As Integer
        Dim GroupNumber As Integer

        For ModelNumber = 1 To 2
            Sem.FitModel(ModelNumber)
            For GroupNumber = 1 To 2
                Sem.GetEstimates(TotalEffects, X, GroupNumber)
                Sem.ColumnNames(TotalEffects, CNames, GroupNumber)
                Sem.RowNames(TotalEffects, RNames, GroupNumber)
                Debug.WriteLine(vbCrLf & "Group " & GroupNumber & ", Model " & ModelNumber)
                PrintMatrix(X, CNames, RNames)
            Next
        Next

        Sem.Dispose()
    End Sub

    'Print a matrix in the debug w indow
    Sub PrintMatrix(ByVal TheMatrix(,) As Double, ByVal CNames$(), ByVal RNames$())
        Dim NRow s1 As Integer, NColumns1 As Integer
        Dim i As Integer, j As Integer
        NRow s1 = UBound(RNames)
        NColumns1 = UBound(CNames)

        Debug.Write(" ")
        For j = 0 To NColumns1
            Debug.Write(CNames(j).PadLeft(10))
        Next

        Debug.WriteLine("")
        For i = 0 To NRow s1
            Debug.Write(RNames(i).PadRight(8))
            For j = 0 To NColumns1
```

```
            Debug.Write(TheMatrix(i, j).ToString(".00000").PadLeft(10))
        Next
        Debug.WriteLine("")
    Next
  End Sub
End Module
```

4.5.2.2.2.123 Row Numbers Method

Obtains the variable numbers associated with the rows of a matrix.

Syntax

object.**RowNumbers** *matrixID*, *theVariableNumbers*, *groupNumber*

The **RowNumbers** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
matrixID	An integer that specifies a matrix of estimates, as described in Settings
theVariableNumbers	An integer array declared in the calling program as **Dim theVariableNumbers() as Integer** The Amos Engine will redimension the array.
groupNumber	Optional. A group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement[156]: [3].

Settings

The settings for *matrixID* are:

Constant	Value	Description
SampleCovariances	13	Sample covariances.
SampleCorrelations	14	Sample correlations.
SampleMeans	15	Sample means.
ImpliedCovariances	10	The implied covariances among the observed variables in the model.

ImpliedCorrelations	11	The implied correlations among the observed variables in the model.
ImpliedMeans	12	The implied means of the observed variables in the model.
AllImpliedCovariances	7	The implied covariances among all variables in the model, with the exception of residual variables.
AllImpliedCorrelations	8	The implied correlations among all variables in the model, with the exception of residual variables.
AllImpliedMeans	9	The implied means of all variables in the model, with the exception of residual variables.
DirectEffects	19	Direct effects.
IndirectEffects	20	Indirect effects.
TotalEffects	5	Total effects.
StandardizedDirectEffects	22	Standardized direct effects.
StandardizedIndirectEffects	23	Standardized indirect effects.
StandardizedTotalEffects	21	Standardized total effects.
FactorScoreWeights	6	Factor score weights.

Remarks

The NeedEstimates 329 method must be used to give notice that a particular matrix of estimates will be needed before the **RowNumbers** method can be used to obtain the row numbers for that matrix. For example, you have to use

object.NeedEstimates 329 (ImpliedMeans)

before using

object.**RowNumbers** (ImpliedMeans, ...)

The following program fits Models A and B of Example 11. It displays the matrix of total effects for each group and model. Rows and columns of the matrix are labeled with Amos's internal variable numbers.

```
Imports System.Diagnostics
Imports AmosEngineLib.AmosEngine.TMatrixID
Imports Microsoft.VisualBasic
Module MainModule
    ' Row Numbers Method Example
    Sub Main()
        Dim CNumbers() As Integer, RNumbers() As Integer, X(,) As Double
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.NeedEstimates(TotalEffects)
        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
        Sem.GroupName("girls")
        Sem.AStructure("academic = (g1) GPA + (g2) attract + (1) e1")
        Sem.AStructure("attract = (g3) height + (g4) w eight + (g5) rating + (g6) academic + (1) e2")
        Sem.AStructure("e2 <--> e1")

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
        Sem.GroupName("boys")
        Sem.AStructure("academic = (b1) GPA + (b2) attract + (1) e1")
        Sem.AStructure("attract = (b3) height + (b4) w eight + (b5) rating + (b6) academic + (1) e2")
        Sem.AStructure("e2 <--> e1")

        Sem.Model("Model_A")
        Sem.Model("Model_B", "g1=b1", "g2=b2", "g3=b3", "g4=b4", "g5=b5", "g6=b6")

        'Print total effects for each model and each group
        Dim ModelNumber As Integer
        Dim GroupNumber As Integer

        For ModelNumber = 1 To 2
            Sem.FitModel(ModelNumber)
            For GroupNumber = 1 To 2
                Sem.GetEstimates(TotalEffects, X, GroupNumber)
                Sem.ColumnNumbers(TotalEffects, CNumbers, GroupNumber)
                Sem.RowNumbers(TotalEffects, RNumbers, GroupNumber)
                Debug.WriteLine(vbCrLf & "Group " & GroupNumber & ", Model " & ModelNumber)
                PrintMatrix1(X, CNumbers, RNumbers)
            Next
        Next

        Sem.Dispose()
    End Sub

    'Print a matrix in the debug w indow
    Sub PrintMatrix1(ByVal TheMatrix(,) As Double, ByVal CNumbers() As Integer, ByVal RNumbers() As Integer)
        Dim NRow s1 As Integer, NColumns1 As Integer
        Dim i As Integer, j As Integer
        NRow s1 = UBound(RNumbers)
        NColumns1 = UBound(CNumbers)
        Debug.Write(" ")
        For j = 0 To NColumns1
            Debug.Write(CNumbers(j).ToString.PadLeft(10))
        Next
        Debug.WriteLine("")
        For i = 0 To NRow s1
            Debug.Write(RNumbers(i).ToString.PadLeft(8))
            For j = 0 To NColumns1
                Debug.Write(TheMatrix(i, j).ToString("#.00000").PadLeft(10))
            Next
            Debug.WriteLine("")
        Next
```

```
    End Sub
End Module
```

4.5.2.2.2.124 SampleMoments Method

Controls the reporting of the sample covariance matrix and (if means and intercepts are explicitly modeled) the sample means.

Syntax

object.**SampleMoments** ()
object.**SampleMoments** (*tf*)

The **SampleMoments** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. A boolean value that controls the reporting of sample moments. True (default) requests the output. False suppresses it.

Placement[156]: [1].

Default

Sample moments are not reported.

See Also

Admissible Method[164]

ImpliedMoments Method[286]

ResidualMoments Method[377]

InputUnbiasedMoments Method[291]

FitUnbiasedMoments Method[239]

This example demonstrates the **SampleMoments** method.

```
Module MainModule
    ' SampleMoments Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.SampleMoments()

        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.125 Seed Method

Specifies a seed for the random number generator used for bootstrapping and for the permutation test. Using Amos twice with the same seed guarantees getting the same sequence of random numbers both times.

Syntax

object.**Seed** (*theSeed*)

The Seed method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
theSeed	An integer between 1 and 29999

Placement[156]: [1].

Default

A seed of 1 is used.

See Also

BootAdf Method[177]

BootBS Method[179]

BootGls Method[183]

BootMl Method 185

BootSls Method 186

Bootstrap Method 188

BootSls Method 186

BootVerify Method 193

Permute Method 368

This example demonstrates the **Seed** method.

```
Module MainModule
   ' Seed Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Sem.Bootstrap(200)
      Sem.Seed(1)
      Sem.TextOutput()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.AStructure("w ordmean = verbal + (1) err_w ")

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.126 Shutdow n Method

The **Shutdown** method has no effect. It is provided for compatibility with earlier versions of Amos.

Syntax

object.**Shutdown**

The **Shutdown** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement 156: [3].

4.5.2.2.2.127 SignificantFigures Method

The **SignificantFigures** method has no effect. It is provided for compatibility with earlier versions of Amos.

Syntax

object.**SignificantFigures** *Ndigits*

The **SignificantFigures** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
Ndigits	Number of significant figures to be used in displaying matrices, within the bounds specified by MinDecimalPlaces 305 and MaxDecimalPlaces 303.

4.5.2.2.2.128 Sls Method

Requests the 'scale free' least squares solution obtained by minimizing (D1) together with (D5) in Appendix B.

Syntax

object.**Sls** ()

The **Sls** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement 156: [1].

Default

When you do not specify an estimation criterion, the maximum likelihood criterion (Ml 305 method) is used.

See Also

Adf Method 163

BootSls Method 186

Gls Method 284

Ml Method 305

Uls Method 403

This example demonstrates the **Sls** method.

```
Module MainModule
    ' Sls Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.Sls()
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.129 Smc Method

Controls reporting of the squared multiple correlation between each endogenous variable and the variables (other than residual variables) that directly affect it.

Syntax

object.**Smc** ()
object.**Smc** (*tf*)

The **Smc** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. If *tf* is True, squared multiple correlations are reported. Otherwise, not.

Placement[156]: [1].

Default

Squared multiple correlations are not reported.

This example demonstrates the **Smc** method.

```
Module MainModule
    ' Smc Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.Smc()
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("wordmean = verbal + (1) err_w")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.130 Specran Method

Controls whether Amos uses a special random number generator that is common to all versions of Amos (since the beginning of time).

Syntax

object.**Specran** ()
object.**Specran** (*tf*)

The **Specran** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. If *tf* is True (default), the special random number generator is used. If *tf* is False, the random number generator of Wichman and Hill (1982)[528] is used.

Placement[156]: [1].

Default

The random number generator of Wichman and Hill (1982)[528] is used.

Remarks

The random number generator that the **Specran** method invokes is not very good. It should not be used except to replicate an example in which the **Specran** method (or the **$specran** command of Amos 3.61 or earlier) was used.

This example demonstrates the **Specran** method.

```
Module MainModule
    ' Specran Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.Bootstrap(500)
        Sem.Specran()
        Sem.Standardized()
        Sem.Smc()
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.FitModel(1)

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.131 Stable Method

Returns True if the solution is a stable linear system for every group.

Syntax

object.**Stable()**

The **Stable** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement[156]: [3].

See Also

Admissible Method[164]

This example demonstrates the **Stable** method.

```
Imports System.Diagnostics
Module MainModule
    ' Stable Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_fem")
        Sem.GroupName("girls")
        Sem.AStructure("academic = GPA + attract + e1 (1)")
        Sem.AStructure("attract = height + w eight + rating + academic + e2 (1)")
        Sem.AStructure("e2 <--> e1")
        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Fels_mal")
        Sem.GroupName("boys")
        Sem.AStructure("academic = GPA + attract + e1 (1)")
        Sem.AStructure("attract = height + w eight + rating + academic + e2 (1)")
        Sem.AStructure("e2 <--> e1")

        If Sem.Admissible Then
            Debug.WriteLine("Admissible")
        Else
            Debug.WriteLine("Inadmissible")
        End If

        If Sem.Stable Then
            Debug.WriteLine("Stable")
        Else
            Debug.WriteLine("Unstable")
        End If

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.132 Standardized Method

Controls reporting of standardized parameter estimates (correlations among exogenous variables, and standardized regression weights). When used with the SampleMoments 389 method, it controls reporting of sample correlations. When used with the ImpliedMoments 286 or AllImpliedMoments 165 methods, it controls reporting of implied correlations. When used with TotalEffects 402, it controls reporting of standardized direct effects, indirect effects and total effects.

Syntax

object.**Standardized** ()
object.**Standardized** (*tf*)

The **Standardized** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

tf	Optional. If *tf* is True (default), standardized estimates are reported. Otherwise, not.

Placement[156]: [1].

Default

Standardized estimates are not reported.

This example demonstrates the **Standardized** method.

```
Module MainModule
    ' Standardized Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.Standardized()
        Sem.Smc()
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.133 TableOutput Method

The **TableOutput** method is obsolete. Use the TextOutput[398] method instead.

4.5.2.2.2.134 Technical Method

Controls the reporting of information about the progress of minimization of the discrepancy function.

Syntax

object.**Technical** ()

object.**Technical** (*tf*)

The **Technical** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
tf	Optional. If *tf* is True, the minimization history is reported. Otherwise, not.

Placement 156: [1].

Default

The minimization history is reported.

See Also

Crit1 Method 217

Crit2 Method 218

Fisher Method 233

Iterations Method 301

Time Method 400

This example demonstrates the **Technical** method.

```
Module MainModule
    ' Technical Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.Technical()
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.135 TextOutput Method

Controls whether results are displayed at the end of the analysis.

Syntax

object.**TextOutput** ()
object.**TextOutput** (*tf*)

The **TextOutput** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

tf	Optional. If *tf* is True (default), results are written to a html file that is displayed at the conclusion of the analysis. If *tf* is False, results are written to a html file, but the file is not automatically displayed at the conclusion of the analysis.

Placement[156]: [1], [2] or [3].

Default

Results are written to a html file, but the file is not automatically displayed at the conclusion of the analysis.

Remarks

By default, the html output file is created in the Windows temporary directory with the name **AmosScratch.AmosOutput**. Use Initialize[288] to change the name and location of the output file.

This example demonstrates the **TextOutput** method.

```
Module MainModule
    ' TextOutput Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.SampleMoments()
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.136 TextOutputFileName Method

Gets the fully qualified path to the file that contains output displayed by the TextOutput[398] method.

Syntax

object.**TextOutputFileName** ()

The **TextOutputFileName** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement[156]: [1].

This example demonstrates the **TextOutputFileName** method.

```
Imports System.Diagnostics
Module MainModule
    ' TextOutputFileName Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Debug.WriteLine("Text output w ill be w ritten to: " & Sem.TextOutputFileName)

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.137 Time Method

Sets an execution time limit.

Syntax

object.**Time** (*seconds*)

The **Time** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
seconds	Time limit in seconds. If the time limit is reached, Amos displays the most recent estimates of the parameters and quits, even if the convergence criteria (see the Crit1[217] and Crit2[218] methods) have not been met.

Placement[156]: [1].

Default

There is no time limit.

Remarks

If the time limit specified by the **Time** method is reached, it will take additional time to complete the output of results. You should allow for this by specifying a value with the **Time** method that is smaller than any time limit placed on your analysis by the operating system.

Permitted values for the time limit range from one to 2,147,483 seconds.

See Also

Crit1 Method 217

Crit2 Method 218

Fisher Method 233

Iterations Method 301

Technical Method 397

This example demonstrates the **Time** method.

```
Module MainModule
   ' Time Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Sem.Time(20)
      Sem.TextOutput()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.AStructure("w ordmean = verbal + (1) err_w ")

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.138 Title Method

Specifies a title for the analysis.

Syntax

object.**Title** (*theTitle*)

The **Title** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

theTitle	A title. The title is displayed in the output generated by TextOutput[398].

Placement[156]: [1].

Default

The title is an empty string.

This example demonstrates the **Title** method.

```
Imports Microsoft.VisualBasic
Module MainModule
   ' Title Method Example
   Sub Main()
      Dim Sem As New AmosEngineLib.AmosEngine

      Sem.title("Example 8:" _
         & vbLf & "factor analysis" _
         & vbLf _
         & vbLf & "Holzinger and Swineford (1939) Grant-White sample." _
         & vbLf & "Intelligence factor study. Raw data of 73 female" _
         & vbLf & "students from the Grant-White high school, Chicago.")

      Sem.TextOutput()
      Sem.SampleMoments()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.AStructure("wordmean = verbal + (1) err_w")

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.139 TotalEffects Method

Controls reporting of direct, indirect and total effects (Fox, 1980[514]).

Syntax

object.**TotalEffects** ()
object.**TotalEffects** (*tf*)

The **TotalEffects** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

tf	Optional. If *tf* is True (the default), then direct, indirect and total effects are reported. Otherwise, not.

Placement[156]: [1].

Default

Direct, indirect and total effects are not reported.

Remarks

When you use the Standardized[396] method, standardized direct, indirect and total effects are reported along with direct, indirect and total effects.

```
This example demonstrates the TotalEffects method.
Module MainModule
   ' TotalEffects Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine
      Sem.TotalEffects()
      Sem.TextOutput()

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.AStructure("w ordmean = verbal + (1) err_w ")

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.140 Uls Method

Requests an unweighted least squares solution, obtained by minimizing (D1) together with (D6) in the *User's Guide Appendix B* .

Syntax

object.**Uls** ()

The **Uls** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.

Placement[156]: [1].

Default

When you do not specify an estimation criterion, the maximum likelihood criterion (*Ml* 305 method) is used.

See Also

BootUls Method 192

Adf Method 163

Gls Method 284

Ml Method 305

Sls Method 392

This example demonstrates the **Uls** method.

```
Module MainModule
    ' Uls Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.Uls()
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.141 UVariableCount Method

Specifies the number of unobserved variables in the model.

Syntax

object.**UVariableCount** (*nVariables*)

The **UVariableCount** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
nVariables	The number of unobserved variables in the model.

Placement 156: [2].

Default

When the **UVariableCount** method is not used, no error checking is done based on the number of unobserved variables.

Remarks

Amos checks *nVariables* for consistency with the model and the data file. If a discrepancy is found, Amos reports the discrepancy and quits. Spelling or typing errors are frequently detected by this check, since two variant spellings of a variable name will be treated as references to two distinct variables.

In a multiple-group analysis, the **UvariableCount** method can be used once per group.

See Also

OVariableCount Method 352

VariableCount Method 407

This example demonstrates the **UVariableCount** method.

```
Module MainModule
    ' UVariableCount Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.OVariableCount(6)
        Sem.UVariableCount(8)
        Sem.VariableCount(14)

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.142 Var Method

Specifies a variance as a model parameter.

Syntax

object.**Var** (*variableName*)

object.**Var** (*variableName, parameterValue*)

object.**Var** (*variableName, parameterName*)

The **Var** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
variableName	The character string, *variableName*, is the name of an exogenous variable. The **Var** method makes its variance a model parameter.
parameterValue	(Optional) Parameter value. If *parameterValue* = 3, say, then the variance is fixed at 3.
parameterName	(Optional) Parameter name. If parameterName = "abc", say, then the variance is named "abc". It is constrained to be equal to any other parameters named "abc". If parameterName = "3?", say then the variance is given an initial value of 3, and is unconstrained. If parameterName = "abc:3", say, then the variance is named "abc" and is given an initial value of 3. It is constrained to be equal to any other parameters named "abc". If parameterName is an empty string (""), the variance is an unconstrained parameter.

Placement[156]: [2].

Default

The variance of an exogenous variable is an unconstrained parameter unless it has been constrained or fixed at a constant by use of the **Var** or AStructure[169] method.

Remarks

If *parameterValue* and *parameterName* are omitted, the variance is an unconstrained parameter.

The following program uses the Path 364, Cov 212 and **Var** methods to specify Model C of Example 6.

```
Module MainModule
    ' Var Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine
        Sem.TextOutput()
        Sem.Standardized()
        Sem.Smc()
        Sem.AllImpliedMoments()
        Sem.FactorScoreWeights()
        Sem.TotalEffects()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Wheaton")
        Sem.Path("anomia67", "67_alienation", 1)
        Sem.Path("anomia67", "eps1", 1)
        Sem.Path("powles67", "67_alienation", "path_p")
        Sem.Path("powles67", "eps2", 1)
        Sem.Path("anomia71", "71_alienation", 1)
        Sem.Path("anomia71", "eps3", 1)
        Sem.Path("powles71", "71_alienation", "path_p")
        Sem.Path("powles71", "eps4", 1)
        Sem.Path("67_alienation", "ses")
        Sem.Path("67_alienation", "zeta1", 1)
        Sem.Path("71_alienation", "67_alienation")
        Sem.Path("71_alienation", "ses")
        Sem.Path("71_alienation", "zeta2", 1)
        Sem.Path("education", "ses", 1)
        Sem.Path("education", "delta1", 1)
        Sem.Path("SEI", "ses")
        Sem.Path("SEI", "delta2", 1)
        Sem.Cov("eps3", "eps1")
        Sem.Var("eps1", "var_a")
        Sem.Var("eps2", "var_p")
        Sem.Var("eps3", "var_a")
        Sem.Var("eps4", "var_p")

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.143 VariableCount Method

Specifies the number of variables in the model.

Syntax

object.**VariableCount** (*nVariables*)

The **VariableCount** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
nVariables	The number of variables in the model.

Placement[156]: [2].

Default

When the **VariableCount** method is not used, no error checking is done based on the number of variables.

Remarks

Amos checks *nVariables* for consistency with the model and the data file. If a discrepancy is found, Amos reports the discrepancy and quits. Spelling or typing errors are frequently detected by this check, since two variant spellings of a variable name will be treated as references to two distinct variables.

In a multiple-group analysis, the **VariableCount** method can be used once per group.

See Also

OVariableCount Method[352]

UVariableCount Method[404]

This example demonstrates the **VariableCount** method.

```
Module MainModule
    ' VariableCount Method Example
    Sub Main()
        Dim Sem As New  AmosEngineLib.AmosEngine
        Sem.TextOutput()

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        Sem.OVariableCount(6)
        Sem.UVariableCount(8)
        Sem.VariableCount(14)

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.144 VariableName Method

Gets the name of a variable.

Syntax

object.**VariableName** (*variableNumber*)
object.**VariableName** (*variableNumber*, *groupNumber*)

The **VariableName** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
variableNumber	A variable number. Variable numbers start at 1 in each group.
groupNumber	Optional group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement[156]: [3].

This example demonstrates the **VariableName** method.

```
Imports System.Diagnostics
Module MainModule
   ' VariableName Method Example
   Sub Main()
      Dim Sem As New  AmosEngineLib.AmosEngine

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = spatial + (1) err_c")
      Sem.AStructure("lozenges = spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = verbal + (1) err_s")
      Sem.AStructure("w ordmean = verbal + (1) err_w ")

      'List all of the variables in the model
      Dim i As Integer
      For i = 1 To Sem.NumberOfVariables
         Debug.WriteLine(i.ToString.PadLeft(5) & " " & Sem.Variablename(i))
      Next

      'What is the variable number of "cubes"?
      Debug.WriteLine("")
      Debug.WriteLine("""cubes"" is variable number " & Sem.VariableNumber("cubes"))

      Sem.Dispose()
   End Sub
End Module
```

4.5.2.2.2.145 VariableNumber Method

Gets a variable number.

A "variable number" is a variable's position on Amos's internal variable list. The first variable on the list is variable number 1. Some methods refer to model variables by number (*i.e.*, by list position).

Syntax

object.**VariableNumber** (*variableName*)
object.**VariableNumber** (*variableName*, *groupNumber*)

The **VariableNumber** method syntax has the following parts:

Part	Description
object	An object of type **AmosEngine**.
variableName	A variable name.
groupNumber	Optional group number. The first group is group number 1. If *groupNumber* is omitted, the first group is used.

Placement 156: [3].

This example demonstrates the **VariableNumber** method.

```
Imports System.Diagnostics
Module MainModule
    ' VariableNumber Method Example
    Sub Main()
        Dim Sem As New AmosEngineLib.AmosEngine

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = spatial + (1) err_c")
        Sem.AStructure("lozenges = spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = verbal + (1) err_s")
        Sem.AStructure("w ordmean = verbal + (1) err_w ")

        'List all of the variables in the model
        Dim i As Integer
        For i = 1 To Sem.NumberOfVariables
            Debug.WriteLine(i.ToString.PadLeft(5) & " " & Sem.Variablename(i))
        Next

        'What is the variable number of "cubes"?
        Debug.WriteLine("")
        Debug.WriteLine("""cubes"" is variable number " & Sem.VariableNumber("cubes"))

        Sem.Dispose()
    End Sub
End Module
```

4.5.2.2.2.146 WasInverted Method

True if the matrix of (approximate or exact) second derivatives was successfully inverted by the most recent use of EvaluateEx2a 227 or EvaluateEx2e 229.

Syntax

result = *object*.**WasInverted** ()

The **WasInverted** method syntax has the following parts:

Part	Description
result	True if the matrix of (approximate or exact) second derivatives was successfully inverted. False otherwise.
object	An object of type **AmosEngine**.

Placement 156: [3].

See Also

Evaluate2a and EvaluateEx2a methods example 228

4.5.3 AmosMatrix Class Reference

An **AmosMatrix** object contains a matrix (such as a sample covariance or direct effects matrix) along with other properties, including row and column names. Several Amos Matrix object properties are provided for retrieving matrix-type output of Amos Engine results.

If you are not using Amos's built-in program editor, you need to provide a reference to **AmosEngineLib.dll** in order to use the **AmosMatrix** class. In Visual Studio 2003:

- Click **Project -> Add Reference**.
- In the **Add Reference** dialog, click **Browse**.
- When the **Select Component** dialog opens, select AmosEngineLib.dll from the Amos program directory and click **Open**.
- In the **Add Reference** dialog, click **OK**.

4.5.3.1 Properties

This section documents the properties of the AmosMatrix class.

4.5.3.1.1 ColumnName Property

Name of the variable associated with a matrix column. Read-only.

Syntax

result = *object*.**ColumnName**(*i*)

The **ColumnName** property syntax has the following parts:

Part	Description
result	The name of the variable associated with column i.
object	An object of type **AmosMatrix**.

i	An integer specifying a column number. Column 0 is the first column.

Placement[156]: [3].

4.5.3.1.1.1 ColumnName Property Example

The following program displays the names of the observed variables in the model of Example 4.

```
Imports System.Diagnostics
Imports AmosEngineLib
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    Sub Main()
        Dim Sem As New AmosEngine
        Dim AM As New AmosMatrix
        Dim i As Integer
        Sem.NeedEstimates(SampleCovariances)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Warren5v")
        Sem.AStructure("performance = knowledge + value + satisfaction + error (1)")

        Sem.GetEstimatesEx(SampleCovariances, AM)
        For i = 0 To AM.NColumns - 1
            Debug.WriteLine(AM.ColumnName(i))
        Next

        Sem.Dispose()
    End Sub
End Module
```

4.5.3.1.2 ColumnNumber Property

Variable number (maintained internally by the Amos engine) of the variable associated with a matrix column. Read-only.

Syntax

result = *object*.**ColumnNumber**(*i*)

The **ColumnNumber** property syntax has the following parts:

Part	Description
result	The Amos engine's internal variable number for the variable associated with column i.
object	An object of type **AmosMatrix**.
i	An integer specifying a column number. Column 0 is the first column.

Placement[156]: [3].

4.5.3.1.2.1 ColumnNumber Property Example

The following program displays the Amos engine's internal variable numbers for the observed variables in the model of Example 4.

```
Imports System.Diagnostics
Imports AmosEngineLib
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    Sub Main()
        Dim Sem As New AmosEngine
        Dim AM As New AmosMatrix
        Dim i As Integer
        Sem.NeedEstimates(SampleCovariances)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Warren5v")
        Sem.AStructure("performance = know ledge + value + satisfaction + error (1)")

        Sem.GetEstimatesEx(SampleCovariances, AM)
        For i = 0 To AM.NColumns - 1
            Debug.WriteLine(AM.ColumnNumber(i))
        Next

        Sem.Dispose()
    End Sub
End Module
```

4.5.3.1.3 NColumns Property

The number of columns in a matrix. Read-only.

Syntax

result = *object*.**NColumns**

The **NColumns** property syntax has the following parts:

Part	Description
result	The number of columns in object's matrix.
object	An object of type **AmosMatrix**.

Placement[156]: [3].

4.5.3.1.3.1 NColumns Property Example

The following program displays the names of the observed variables in the model of Example 4.

```
Imports System.Diagnostics
Imports AmosEngineLib
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    Sub Main()
        Dim Sem As New AmosEngine
        Dim AM As New AmosMatrix
        Dim i As Integer
        Sem.NeedEstimates(SampleCovariances)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Warren5v")
        Sem.AStructure("performance = knowledge + value + satisfaction + error (1)")

        Sem.GetEstimatesEx(SampleCovariances, AM)
        For i = 0 To AM.NColumns - 1
            Debug.WriteLine(AM.ColumnName(i))
        Next

        Sem.Dispose()
    End Sub
End Module
```

4.5.3.1.4 NRows Property

The number of rows in a matrix. Read-only.

Syntax

result = *object*.**nRows**

The **nRows** property syntax has the following parts:

Part	Description
result	The number of rows in object's matrix.
object	An object of type **AmosMatrix**.

Placement[156]: [3].

4.5.3.1.4.1 NRows Property Example

The following program displays the names of the observed variables in the model of Example 4.

```
Imports System.Diagnostics
Imports AmosEngineLib
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    Sub Main()
        Dim Sem As New AmosEngine
        Dim AM As New AmosMatrix
        Dim i As Integer
        Sem.NeedEstimates(SampleCovariances)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Warren5v")
        Sem.AStructure("performance = knowledge + value + satisfaction + error (1)")

        Sem.GetEstimatesEx(SampleCovariances, AM)
        For i = 0 To AM.NRows - 1
            Debug.WriteLine(AM.RowName(i))
        Next

        Sem.Dispose()
    End Sub
End Module
```

4.5.3.1.5 RowName Property

Name of the variable associated with a matrix row. Read-only.

Syntax

result = *object*.**RowName**(*i*)

The **RowName** property syntax has the following parts:

Part	Description
result	The name of the variable associated with row i.
object	An object of type **AmosMatrix**.
i	An integer specifying a row number. Row 0 is the first row.

Placement[156]: [3].

4.5.3.1.5.1 Row Name Property Example

The following program displays the names of the observed variables in the model of Example 4.

```
Imports System.Diagnostics
Imports AmosEngineLib
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    Sub Main()
        Dim Sem As New AmosEngine
        Dim AM As New AmosMatrix
        Dim i As Integer
        Sem.NeedEstimates(SampleCovariances)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Warren5v")
        Sem.AStructure("performance = knowledge + value + satisfaction + error (1)")

        Sem.GetEstimatesEx(SampleCovariances, AM)
        For i = 0 To AM.NRows - 1
            Debug.WriteLine(AM.RowName(i))
        Next

        Sem.Dispose()
    End Sub
End Module
```

4.5.3.1.6 Row Number Property

Variable number (maintained internally by the Amos engine) of the variable associated with a matrix row. Read-only.

Syntax

result = *object*.**RowNumber**(*i*)

The **RowNumber** property syntax has the following parts:

Part	Description
result	The Amos engine's internal variable number for the variable associated with row i.
object	An object of type **AmosMatrix**.
i	An integer specifying a row number. Row 0 is the first row.

Placement[156]: [3].

4.5.3.1.6.1 Row Number Property Example

The following program displays the Amos engine's internal variable numbers for the observed variables in the model of Example 4.

```
Imports System.Diagnostics
Imports AmosEngineLib
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    Sub Main()
        Dim Sem As New AmosEngine
        Dim AM As New AmosMatrix
        Dim i As Integer
        Sem.NeedEstimates(SampleCovariances)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Warren5v")
        Sem.AStructure("performance = knowledge + value + satisfaction + error (1)")

        Sem.GetEstimatesEx(SampleCovariances, AM)
        For i = 0 To AM.NRows - 1
            Debug.WriteLine(AM.RowNumber(i))
        Next

        Sem.Dispose()
    End Sub
End Module
```

4.5.3.1.7 X Property

A matrix element. Read-only.

Syntax

result = *object*.**X**(*i*, *j*)

The **X** property syntax has the following parts:

Part	Description
result	The (i, j) element of object's matrix.
object	An object of type **AmosMatrix**.
i	An integer specifying a row number. Row 0 is the first row.
j	An integer specifying a column number. Column 0 is the first row.

Placement 156: [3].

4.5.3.1.7.1 X Property Example

The following program fits the model of Example 4 and displays the sample covariance matrix.

```
Imports System.Diagnostics
Imports AmosEngineLib
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    Sub Main()
        Dim Sem As New AmosEngine
        Dim AM As New AmosMatrix
        Dim i As Integer, j As Integer
        Sem.NeedEstimates(SampleCovariances)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Warren5v")
        Sem.AStructure("performance = knowledge + value + satisfaction + error (1)")

        Sem.GetEstimatesEx(SampleCovariances, AM)
        For i = 0 To AM.NRows - 1
            For j = 0 To AM.NColumns - 1
                Debug.Write(AM.X(i, j).ToString("#.0000").PadLeft(10))
            Next
            Debug.WriteLine("")
        Next

        Sem.Dispose()
    End Sub
End Module
```

4.5.4 AmosDebug Class Reference

This class displays output in a debug window. There are several methods and properties for manipulating the debug window and formatting its contents.

If you are not using Amos's built-in program editor, you need to provide a reference to **AmosDebug.dll** in order to use the **AmosDebug** class. In Visual Studio 2003:

- Click **Project -> Add Reference**.
- In the **Add Reference** dialog, click **Browse**.
- When the **Select Component** dialog opens, select AmosDebug.dll from the Amos program directory and click **Open**.
- In the **Add Reference** dialog, click **OK**.

4.5.4.1 Properties

This section documents the properties of the AmosDebug class.

4.5.4.1.1 DecimalPlaces Property

Gets or sets the number of decimal places used to display numbers.

Syntax

object.**DecimalPlaces** *[=value]*

The **DecimalPlaces** property syntax has the following parts:

Part	Description
object	An object of type **AmosDebug**.
value	The number of decimal places used to display numbers.

4.5.4.1.1.1 DecimalPlaces Property Example

The following program fits the model of Example 4 and displays the sample covariance matrix with 5 decimal places of precision. Each matrix element is displayed in a field that is 12 characters wide.

```
Imports AmosEngineLib
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    Sub Main()
        Dim Sem As New AmosEngine
        Dim AM As New AmosMatrix
        Dim AD As New AmosDebug.AmosDebug
        Dim i As Long, j As Long
        Sem.NeedEstimates(SampleCovariances)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Warren5v")
        Sem.AStructure("performance = know ledge + value + satisfaction + error (1)")

        Sem.GetEstimatesEx(SampleCovariances, AM)

        AD.FieldWidth = 12
        AD.DecimalPlaces = 5
        AD.PrintX(AM, "Sample Covariances")

        Sem.Dispose()
    End Sub
End Module
```

4.5.4.1.2 FieldWidth Property

Sets or gets the number of characters used to display each matrix element.

Syntax

object.**FieldWidth** *[=value]*

The **FieldWidth** property syntax has the following parts:

Part	Description
object	An object of type **AmosDebug**.

value	The number of characters used to display each matrix element.

4.5.4.1.2.1 FieldWidth Property Example

The following program fits the model of Example 4 and displays the sample covariance matrix with 5 decimal places of precision. Each matrix element is displayed in a field that is 12 characters wide.

```
Imports AmosEngineLib
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    Sub Main()
        Dim Sem As New AmosEngine
        Dim AM As New AmosMatrix
        Dim AD As New AmosDebug.AmosDebug
        Dim i As Long, j As Long
        Sem.NeedEstimates(SampleCovariances)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Warren5v")
        Sem.AStructure("performance = know ledge + value + satisfaction + error (1)")

        Sem.GetEstimatesEx(SampleCovariances, AM)

        AD.FieldWidth = 12
        AD.DecimalPlaces = 5
        AD.PrintX(AM, "Sample Covariances")

        Sem.Dispose()
    End Sub
End Module
```

4.5.4.2 Methods

This section documents the methods of the AmosDebug class.

4.5.4.2.1 Clear Method

The **Clear** method is no longer supported because output from the **AmosDebug** class is now written to the trace listeners in the Listeners collection, not to a separate **AmosDebug** window.

In earlier versions, the **Clear** method cleared (erased) the contents of the **AmosDebug** window.

Syntax

object.**Clear** ()

The **Clear** method syntax has the following parts:

Part	Description
object	An object of type **AmosDebug**.

4.5.4.2.2 Fixed Method

Displays elements of numeric matrices in fixed point format.

Syntax

object.**Fixed** ()

The **Fixed** method syntax has the following parts:

Part	Description
object	An object of type **AmosDebug**.

Default

Elements of numeric matrices are displayed in fixed point format with 3 decimal places and a field width of 10.

4.5.4.2.2.1 Fixed Method Example

The following program fits the model of Example 4 and displays the sample covariance matrix with 5 decimal places of precision in fixed point format. Each matrix element is displayed in a field that is 12 characters wide.

```
Imports AmosEngineLib
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    Sub Main()
        Dim Sem As New AmosEngine
        Dim AM As New AmosMatrix
        Dim AD As New AmosDebug.AmosDebug
        Dim i As Long, j As Long
        Sem.NeedEstimates(SampleCovariances)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Warren5v")
        Sem.AStructure("performance = know ledge + value + satisfaction + error (1)")

        Sem.GetEstimatesEx(SampleCovariances, AM)

        AD.Fixed()
        AD.FieldWidth = 12
        AD.DecimalPlaces = 5
        AD.PrintX(AM, "Sample Covariances")

        Sem.Dispose()
    End Sub
End Module
```

4.5.4.2.3 Hide Method

The **Hide** method is no longer supported because output from the **AmosDebug** class is now written to the trace listeners in the Listeners collection, not to a separate **AmosDebug** window.

In earlier versions, the **Hide** method made the **AmosDebug** window invisible.

Syntax

object.**Hide** ()

The **Hide** method syntax has the following parts:

Part	Description
object	An object of type **AmosDebug**.

Default

The **AmosDebug** window is visible.

4.5.4.2.4 KeepOnTop Method

The **KeepOnTop** method is no longer supported because output from the **AmosDebug** class is now written to the trace listeners in the Listeners collection, not to a separate **AmosDebug** window.

In earlier versions, the **KeepOnTop** method kept the **AmosDebug** window on top of other windows (if called with an argument of True) or allowed other windows to be on top (if called with an argument of False).

Syntax

object.**KeepOnTop** ()

The **KeepOnTop** method syntax has the following parts:

Part	Description
object	An object of type **AmosDebug**.

4.5.4.2.5 PrintTranspose Method

Displays the transpose of a 1- or 2-dimensional array.

Syntax

object.**PrintTranspose** (*item*)
object.**PrintTranspose** (*item, title*)

The **PrintTranspose** method syntax has the following parts:

Part	Description
object	An object of type **AmosDebug**.

item	A 1- or 2-dimensional array of Double or String, or an AmosMatrix[411] object.
title	(Optional) A string that describes *item*.

Remarks

PrintTranspose transposes 2-dimensional arrays and AmosMatrix[411] objects.

PrintTranspose displays a 1-dimensional array as a single row.

See Also

PrintX, PrintTranspose, PrintTriangle Methods Example[424]

4.5.4.2.6 PrintTriangle Method

Displays a 1-dimensional array as a lower triangular matrix.

Syntax

object.**PrintTriangle** (*theMatrix*)
object.**PrintTriangle** (*theMatrix, title*)

The **PrintTriangle** method syntax has the following parts:

Part	Description
object	An object of type **AmosDebug**.
theMatrix	A 1-dimensional array of Double or String.
title	(Optional) A string that describes *theMatrix*.

Remarks

PrintTriangle displays a vector *x* as

x(0)

x(1) x(2)

x(3) x(4) x(5)

...

Thus, the **PrintTriangle** method may be used when the vector contains the lower triangular portion of a symmetric matrix (see Evaluate2a and EvaluateEx2a methods example[228]), and also when the vector contains the lower triangular portion of a lower triangular matrix.

See Also

PrintX, PrintTranspose, PrintTriangle Methods Example [424]

4.5.4.2.7 PrintX Method

Displays a number, a string, a 1- or 2-dimensional array of numbers or strings, or AmosMatrix [411] object.

Syntax

object.**PrintX** (*item*)
object.**PrintX** (*item, title*)

The **PrintX** method syntax has the following parts:

Part	Description
object	An object of type **AmosDebug**.
item	A Double, a String, a 1- or 2-dimensional array of Double or String, or AmosMatrix [411] object.
title	(Optional) A string that describes *item*.

Remarks

PrintX displays a 1-dimensional array as a single column.

4.5.4.2.7.1 PrintX, PrintTranspose, PrintTriangle Methods Example

The following program creates a 1-dimensional array of strings and displays it three times — as a row vector, as a column vector and as a lower triangular matrix. The program then fits the model of Example 4 and displays the ML chi square statistic and degrees of freedom. Finally, it displays the matrix of total effects and its transpose.

```
Imports AmosEngineLib
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
   Sub Main()
      Dim Sem As New  AmosEngine
      Dim AM As New  AmosMatrix
      Dim AD As New  AmosDebug.AmosDebug
      AD.FieldWidth = 8
      AD.PrintX("AmosDebug Demonstration")
      Dim A() As String
      ReDim A(5)
      A(0) = "aa"
      A(1) = "bb"
      A(2) = "cc"
      A(3) = "dd"
      A(4) = "ee"
      A(5) = "ff"
      AD.PrintX(A, "PrintX Demonstration")
      AD.PrintTranspose(A, "PrintTranspose Demonstration")
      AD.PrintTriangle(A, "PrintTriangle Demonstration")

      Dim Result As Integer
      Sem.NeedEstimates(DirectEffects)

      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Warren5v")
      Sem.AStructure("performance = know ledge + value + satisfaction + error (1)")

      Result = Sem.FitModel
      If Result = 0 Then
         AD.PrintX(Sem.Cmin, "Chi square")
         AD.PrintX("Degrees of freedom is " & Sem.df)
         Sem.GetEstimatesEx(TotalEffects, AM)
         AD.FieldWidth = 12
         AD.PrintX(AM, "Total Effects")
         AD.PrintTranspose(AM, "Total Effects Transposed")
      Else
         AD.PrintX("Minimization failed")
      End If

      Sem.Dispose()
   End Sub
End Module
```

4.5.4.2.8 Scientific Method

Displays elements of numeric matrices in scientific format.

Syntax

object.**Scientific**

The **Scientific** method syntax has the following parts:

Part	Description
object	An object of type **AmosDebug**.

Default

Elements of numeric matrices are displayed in fixed point format with 3 decimal places and a field width of 10.

4.5.4.2.8.1 Scientific Method Example

The following program fits the model of Example 4 and displays the sample covariance matrix with 5 decimal places of precision in scientific format. Each matrix element is displayed in a field that is 12 characters wide.

```
Imports AmosEngineLib
Imports AmosEngineLib.AmosEngine.TMatrixID
Module MainModule
    Sub Main()
        Dim Sem As New AmosEngine
        Dim AM As New AmosMatrix
        Dim AD As New AmosDebug.AmosDebug
        Sem.NeedEstimates(SampleCovariances)

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Warren5v")
        Sem.AStructure("performance = know ledge + value + satisfaction + error (1)")

        Sem.GetEstimatesEx(SampleCovariances, AM)

        AD.Scientific()
        AD.FieldWidth = 12
        AD.DecimalPlaces = 5
        AD.PrintX(AM, "Sample Covariances")

        Sem.Dispose()
    End Sub
End Module
```

4.5.4.2.9 Show Method

The **Show** method is no longer supported because output from the **AmosDebug** class is now written to the trace listeners in the Listeners collection, not to a separate **AmosDebug** window.

In earlier versions, the **Show** method made the **AmosDebug** window visible.

Syntax

object.**Show** ()

The **Show** method syntax has the following parts:

Part	Description
object	An object of type **AmosDebug**.

Default

The **AmosDebug** window is visible.

4.5.4.2.10 Unload Method

The **Unload** method is no longer supported because output from the **AmosDebug** class is now written to the trace listeners in the Listeners collection, not to a separate **AmosDebug** window.

In earlier versions, the **Unload** method closed the **AmosDebug** window.

Syntax

object.**Unload** ()

The **Unload** method syntax has the following parts:

Part	Description
object	An object of type **AmosDebug**.

4.5.5 AmosRanGen Class Reference

The **AmosRanGen** class generates multivariate normal random numbers with prescribed means and covariances. The class also generates uniform random numbers and performs some related matrix operations.

If you are not using Amos's built-in program editor, you need to provide a reference to **AmosRandom.dll** in order to use the **AmosRanGen** class. In Visual Studio 2003:

- Click **Project -> Add Reference**.
- In the **Add Reference** dialog, click **Browse**.
- When the **Select Component** dialog opens, select AmosRandom.dll from the Amos program directory and click **Open**.
- In the **Add Reference** dialog, click **OK**.

4.5.5.1 AmosRanGen Class Members

Properties 427

Methods 431

4.5.5.1.1 Properties

This section documents the properties of the AmosRanGen 427 class.

4.5.5.1.1.1 CholeskyEpsilon Property

The threshold used in the Cholesky decomposition of the population covariance matrix to decide whether a variable is linearly dependent on previous variables. The *k*-th variable is judged to be linearly dependent on variables 1 through *k*-1 if the *k*-th pivot is less than **CholeskyEpsilon**.

The InstantSqrt 437 and SpecifyPopulation 444 methods make use of **CholeskyEpsilon**.

Syntax

value= *object*.**CholeskyEpsilon**
object.**CholeskyEpsilon** = *value*

The **CholeskyEpsilon** property syntax has the following parts:

Part	Description
object	An object of type **AmosRanGen**.
value	The threshold.

Default

The default threshold is 1.0e-10.

The following program sets CholeskyEpsilon to 1.0e-15. Then it displays the Cholesky square root of the matrix $\begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$ as well as the rank of the matrix and the square root of its determinant.

```
Module MainModule
   Sub Main()
      Dim arand As New AMOSRANDOMLib6.AmosRanGen
      Dim Cov(2) As Double
      Cov(0) = 3
      Cov(1) = 1
      Cov(2) = 2
      Dim Rank As Integer
      Dim sqrdet As Double

      arand.CholeskyEpsilon = 0.0000000001

      Call arand.InstantSqrt(2, Cov(0), Rank, sqrdet)
      Dim ad As New AmosDebug.AmosDebug
      Call ad.PrintTriangle(Cov, "Square root of matrix")
      Call ad.PrintX(Rank, "Rank")
      Call ad.PrintX(sqrdet, "Square root of determinant")
   End Sub
End Module
```

4.5.5.1.1.2 Rank Property

Gets the rank of the population covariance matrix already specified with SpecifyPopulation [444].

Syntax

result = *object*.**Rank**

The **Rank** property syntax has the following parts:

Part	Description
result	Output of type Integer. The rank of the covariance matrix.
object	An object of type **AmosRanGen**.

Remarks

The **Rank** property is read-only.

The following program specifies a multivariate normal population with covariance matrix and mean given by

$$\Sigma = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}, \qquad \mu = \begin{bmatrix} 4 \\ 5 \end{bmatrix}.$$

Then it displays the rank of the covariance matrix.

```
Imports System.Diagnostics
Module MainModule
   Sub Main()
      Dim arand As New AMOSRANDOMLib6.AmosRanGen
      Dim Covariances(2) As Double
      Dim Means(1) As Double
      Covariances(0) = 3
      Covariances(1) = 1
      Covariances(2) = 2
      Means(0) = 4
      Means(1) = 5

      Call arand.SpecifyPopulation(Covariances, Means)

      Debug.WriteLine("Rank = " & arand.Rank)
   End Sub
End Module
```

4.5.5.1.1.3 SecondMomentsType Property

The type of second order moments returned by RandomMoments 441.

Syntax

value = *object*.**SecondMomentsType**
object.**SecondMomentsType** = *value*

The **SecondMomentsType** property syntax has the following parts:

Part	Description
value	One of the values specified in Settings.

object	An object of type **AmosRanGen**.

Settings

The settings for *value* are:

Constant	Value	Description
SSCP	1	RandomMoments() returns sums of squares and cross products about the mean.
ML	2	RandomMoments() returns the (biased) maximum likelihood estimate of the population covariances.
UNBIASED	3	RandomMoments() returns the unbiased estimate of the population covariances.

The following program specifies a multivariate normal population with covariance matrix and mean given by

$$\Sigma = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}, \qquad \mu = \begin{bmatrix} 4 \\ 5 \end{bmatrix}.$$

Then it draws a sample of 500 and displays the (biased) maximum likelihood estimate of the covariance matrix and mean vector.

```
Module MainModule
    Sub Main()
        Dim arand As New AMOSRANDOMLib6.AmosRanGen
        Dim PopCovariances(2) As Double
        Dim PopMeans(1) As Double
        PopCovariances(0) = 3
        PopCovariances(1) = 1
        PopCovariances(2) = 2
        PopMeans(0) = 4
        PopMeans(1) = 5

        Call arand.SpecifyPopulation(PopCovariances, PopMeans)

        Dim SampleCovariances(2) As Double
        Dim SampleMeans(1) As Double

        arand.SecondMomentsType = AMOSRANDOMLib6.SECONDMOMENTS.ML

        Call arand.RandomMoments(SampleCovariances(0), SampleMeans(0), 2, 500)

        Dim ad As New AmosDebug.AmosDebug
        Call ad.PrintTriangle(SampleCovariances, "Sample Covariances")
        Call ad.PrintX(SampleMeans, "Sample Means")
    End Sub
End Module
```

4.5.5.1.2 Methods

This section documents the methods of the AmosRanGen [427] class.

4.5.5.1.2.1 ChversInPlace Method

Replaces a positive definite symmetric matrix with its inverse.

Syntax

object.**ChversInPlace** (*N, SymMatrix, Work Vector, Determinant, Eps, errorflag*)

The **ChversInPlace** property syntax has the following parts:

Part	Description
object	An object of type **AmosRanGen**.
N	Number of rows in the matrix.
SymMatrix	An array of length N*(N+1)/2, of type Double. On input, the matrix to be inverted. On output, the inverse. SymMatrix contains the first element of the first row, the first two elements of the second row, and so on. The first element of SymMatrix is passed by reference.
Work Vector	An array of length N, of type Double. Used as work space. The first element of WorkVector is passed by reference.
Determinant	Output, of type Double. The determinant.
Eps	Input, of type Double. Each pivot element must exceed Eps.
errorflag	Output, of type Integer. If the matrix was successfully inverted, errorflag=0. Otherwise, errorflag=1.

The following program displays the inverse and the determinant of $\begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$.

```
Module MainModule
    Sub Main()
        Const eps As Double = 0.000000000000001
        Dim arand As New AMOSRANDOMLib6.AmosRanGen
        Dim ad As New AmosDebug.AmosDebug
        Dim Cov(2) As Double
        Cov(0) = 3
        Cov(1) = 1
        Cov(2) = 2
        Call ad.PrintTriangle(Cov, "The matrix")
        Dim w(1) As Double
        Dim det As Double
        Dim ef As Integer

        Call arand.ChversInPlace(2, Cov(0), w(0), det, eps, ef)

        If ef = 0 Then
            Call ad.PrintTriangle(Cov, "The inverse")
            Call ad.PrintX(det, "Determinant")
        Else
            Call ad.PrintX("Error")
        End If
    End Sub
End Module
```

4.5.5.1.2.2 Initialize Method (AmosRanGen)

Initializes the random number generator.

Syntax

object.**Initialize** (*Seed*)

The **Initialize** method syntax has the following parts:

Part	Description
object	An object of type **AmosRanGen**.
Seed	Input, of type Integer. Seed for the random number generator.

The following program displays three pairs of random numbers. The second pair is identical to the first pair.

```
Imports System.Diagnostics
Module MainModule
  Sub Main()
    Dim arand As New AMOSRANDOMLib6.AmosRanGen

    arand.Initialize(5)
    Debug.WriteLine(arand.NextNormal() & arand.NextNormal())

    arand.Initialize(5)
    Debug.WriteLine(arand.NextNormal() & arand.NextNormal())

    arand.Initialize(6)
    Debug.WriteLine(arand.NextNormal() & arand.NextNormal())

  End Sub
End Module
```

4.5.5.1.2.3 InstantRandomVector Method

Gets a random multivariate normal vector where the mean vector and the square root of the covariance matrix are specified.

Syntax

object.**InstantRandomVector** (*N, X, mean, s_sqrt*)

The **InstantRandomVector** property syntax has the following parts:

Part	Description
object	An object of type **AmosRanGen**.
N	Input of type Integer. Number of elements in the vector.
X	Output array of length N, of type Double. The random vector. The first element of X is passed by reference. X must be dimensioned by the caller.
mean	Input array of length N, of type Double. The population mean. The first element of mean is passed by reference.
s_sqrt	Input array of length N*(N+1)/2, of type Double. The square root of the population covariance matrix. s_sqrt contains the first element of the first row, the first two elements of the second row, and so on. The first element of s_sqrt is passed by reference.

See Also

InstantRandomVectorEx Method 434

RandomVector Method 442

The following program displays 5 observations from a multivariate normal population with covariance matrix and mean given by

$$\Sigma = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}, \qquad \mu = \begin{bmatrix} 4 \\ 5 \end{bmatrix}.$$

```
Imports System.Diagnostics
Module MainModule
  Sub Main()
    Dim arand As New AMOSRANDOMLib6.AmosRanGen
    Dim Cov(2) As Double
    Dim Mean(1) As Double
    Cov(0) = 3
    Cov(1) = 1
    Cov(2) = 2
    Mean(0) = 4
    Mean(1) = 5
    Dim Rank As Integer
    Dim sqrdet As Double
    Call arand.InstantSqrt(2, Cov(0), Rank, sqrdet)

    Dim rx(1) As Double
    Dim i As Long

    For i = 1 To 5
      Call arand.InstantRandomVector(2, rx(0), Mean(0), Cov(0))
      Debug.WriteLine(rx(0) & rx(1))
    Next
  End Sub
End Module
```

4.5.5.1.2.4 InstantRandomVectorEx Method

Generates a random multinormal vector and calculates its squared Mahalanobis distance from the mean.

Syntax

object.**InstantRandomVectorEx** (*N, X, mean, s_sqrt, MahalanobisD2*)

The **InstantRandomVectorEx** property syntax has the following parts:

Part	Description
object	An object of type **AmosRanGen**.
N	Input of type Integer. Number of elements in the vector.
X	Output array of length N, of type Double. The random vector. The first element of X is passed by reference. X must be dimensioned by the caller.

mean	Input array of length N, of type Double. The population mean. The first element of mean is passed by reference.
s_sqrt	Input array of length N*(N+1)/2, of type Double. The square root of the population covariance matrix. s_sqrt contains the first element of the first row, the first two elements of the second row, and so on. The first element of s_sqrt is passed by reference.
MahalanobisD2	Output of type Double. The squared Mahalanobis distance between X and mean.

See Also

InstantRandomVector Method 433

RandomVector Method 442

The following program displays 5 observations from a multivariate normal population with covariance matrix and mean given by

$$\Sigma = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}, \qquad \mu = \begin{bmatrix} 4 \\ 5 \end{bmatrix}.$$

The program also displays the squared Mahalanobis distance of each observation from the mean.

```
Imports System.Diagnostics
Module MainModule
   Sub Main()
      Dim arand As New AMOSRANDOMLib6.AmosRanGen
      Dim Cov(2) As Double
      Dim Mean(1) As Double
      Cov(0) = 3
      Cov(1) = 1
      Cov(2) = 2
      Mean(0) = 4
      Mean(1) = 5
      Dim Rank As Integer
      Dim sqrdet As Double
      Call arand.InstantSqrt(2, Cov(0), Rank, sqrdet)

      Dim D2 As Double
      Dim rx(1) As Double
      Dim i As Long

      For i = 1 To 5
         Call arand.InstantRandomVectorEx(2, rx(0), Mean(0), Cov(0), D2)
         Debug.WriteLine(rx(0) & rx(1) & "D2 = " & D2)
      Next
   End Sub
End Module
```

4.5.5.1.2.5 InstantSolve Method

Solves **Ax**=**y** for **x** where **A** is lower triangular.

Syntax

object.**InstantSolve** (*N, A, x, errorflag*)

The **InstantSolve** property syntax has the following parts:

Part	Description
object	An object of type **AmosRanGen**.
N	Input of type Integer. The number of rows in the matrix a.
A	Input array of length N*(N+1)/2, of type Double. A contains the first element of the first row, the first two elements of the second row, and so on. The first element of A is passed by reference.
x	Input and output array of length N, of type double. The first element of x is passed by reference.
errorflag	Output, of type Integer. If the system of linear equations was successfully solved, errorflag=0. Otherwise, errorflag=1.

The following program displays $\begin{bmatrix} 3 & 0 \\ 1 & 2 \end{bmatrix}^{-1}\begin{bmatrix} 4 \\ 5 \end{bmatrix}=\begin{bmatrix} 1.333 \\ 1.833 \end{bmatrix}$.

```
Module MainModule
   Sub Main()
      Dim arand As New AMOSRANDOMLib6.AmosRanGen
      Dim A(2) As Double
      A(0) = 3
      A(1) = 1
      A(2) = 2
      Dim x(1) As Double
      x(0) = 4
      x(1) = 5
      Dim errorflag As Integer

      Call arand.InstantSolve(2, A(0), x(0), errorflag)

      Dim ad As New AmosDebug.AmosDebug
      ad.PrintX(x)
   End Sub
End Module
```

4.5.5.1.2.6 InstantSqrt Method

Replaces a nonnegative definite symmetric matrix with its Cholesky square root.

Syntax

object.**InstantSqrt** (*n, x, rank, sqrdet*)

The **InstantSqrt** property syntax has the following parts:

Part	Description
object	An object of type **AmosRanGen**.
n	Input of type Integer. The number of rows in the matrix.
x	Input and output array of length N*(N+1)/2, of type Double. x contains the first element of the first row, the first two elements of the second row, and so on. The first element of x is passed by reference.
rank	Output of type Integer. The number of pivot elements greater than CholeskyEpsilon 427.
sqrdet	Output of type double. The square root of the determinant.

Remarks

On output, *x* and *sqrdet* are meaningful only if *rank* = *n*.

See Also

CholeskyEpsilon Property 427

The following program displays the Cholesky square root of the matrix $\begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$ as well as the rank of the matrix and the square root of its determinant.

```
Module MainModule
    Sub Main()
        Dim arand As New AMOSRANDOMLib6.AmosRanGen
        Dim Cov(2) As Double
        Cov(0) = 3
        Cov(1) = 1
        Cov(2) = 2
        Dim Rank As Integer
        Dim sqrdet As Double

        Call arand.InstantSqrt(2, Cov(0), Rank, sqrdet)

        Dim ad As New AmosDebug.AmosDebug
        Call ad.PrintTriangle(Cov, "Square root of matrix")
        Call ad.PrintX(Rank, "Rank")
        Call ad.PrintX(sqrdet, "Square root of determinant")
    End Sub
End Module
```

4.5.5.1.2.7 MahalanobisD2 Method

Gets the squared Mahalanobis distance of an observation from the mean.

Syntax

object.**MahalanobisD2** (*N, covsqrt, mean, X, D2, errorflag*)

The **MahalanobisD2** property syntax has the following parts:

Part	Description
object	An object of type **AmosRanGen**.
N	Input of type Integer. The number of elements in the vector.
covsqrt	Input array of length N*(N+1)/2, of type Double. The square root of the covariance matrix. covsqrt contains the first element of the first row, the first two elements of the second row, and so on. The first element of covsqrt is passed by reference.
mean	Input array of length N, of type Double. The mean vector. The first element of mean is passed by reference.
X	Input array of length N, of type Double. The observation. The first element of X is passed by reference.
D2	Output of type Double. The squared Mahalanobis distance between X and mean.

errorflag	Output of type Integer. errorflag=1 if an error occurred, errorflag=0 otherwise.

The following program evaluates the squared Mahalanobis distance,

$(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$,

of the vector $\mathbf{x}=\begin{bmatrix}6\\7\end{bmatrix}$ from the mean of a multivariate normal distribution with covariance matrix and mean given by

$\boldsymbol{\Sigma}=\begin{bmatrix}3 & 1\\1 & 2\end{bmatrix}$, $\boldsymbol{\mu}=\begin{bmatrix}4\\5\end{bmatrix}$.

```
Imports System.Diagnostics
Module MainModule
   Sub Main()
      Dim arand As New AMOSRANDOMLib6.AmosRanGen
      Dim Cov(2) As Double
      Dim Mean(1) As Double
      Cov(0) = 3
      Cov(1) = 1
      Cov(2) = 2
      Mean(0) = 4
      Mean(1) = 5
      Dim Rank As Integer
      Dim sqrdet As Double
      Call arand.InstantSqrt(2, Cov(0), Rank, sqrdet)
      Dim D2 As Double
      Dim ef As Integer
      Dim x(1) As Double
      x(0) = 6
      x(1) = 7

      Call arand.MahalanobisD2(2, Cov(0), Mean(0), x(0), D2, ef)

      If ef = 0 Then
         Debug.WriteLine("Squared Mahalanobis distance = " & D2)
      Else
         Debug.WriteLine("Error")
      End If
   End Sub
End Module
```

4.5.5.1.2.8 NextNormal Method

Gets a normally distributed random number that has mean 0 and standard deviation 1.

Syntax

result = *object*.**NextNormal ()**

The **NextNormal** method syntax has the following parts:

Part	Description
result	Output of type Double. The random number.
object	An object of type **AmosRanGen**.

Remarks

NextNormal uses the NextUniform 440 method.

The following program generates 20 random numbers.

```
Imports System.Diagnostics
Module MainModule
    Sub Main()
        Dim arand As New AMOSRANDOMLib6.AmosRanGen
        Dim i As Long
        For i = 1 To 20
            Debug.WriteLine(arand.NextNormal())
        Next
    End Sub
End Module
```

4.5.5.1.2.9 NextUniform Method

Gets a random number that is uniformly distributed over the interval from 0 to 1.

Syntax

result = *object*.**NextUniform** ()

The **NextUniform** method syntax has the following parts:

Part	Description
result	Output of type Double. The random number.
object	An object of type **AmosRanGen**.

The following program generates 20 random numbers.

```
Imports System.Diagnostics
Module MainModule
   Sub Main()
      Dim arand As New AMOSRANDOMLib6.AmosRanGen
      Dim i As Long
      For i = 1 To 20
         Debug.WriteLine(arand.NextUniform())
      Next
   End Sub
End Module
```

4.5.5.1.2.10 RandomMoments Method

Gets a sample mean vector and covariance matrix from the population specified by SpecifyPopulation [444].

Syntax

object.**RandomMoments** (*Covariances*, *Means*, *NCases*)

The **RandomMoments** method syntax has the following parts:

Part	Description
object	An object of type **AmosRanGen**.
Covariances	Output array of type Double. Second order moment matrix from a sample of *NCases* observations. Use SecondMomentsType [429] to specify whether *Covariances* contains unbiased estimates of the population covariances, maximum likelihood estimates of the population covariances, or sums of squares and cross products about the sample mean. *Covariances* contains the first element of the first row, the first two elements of the second row, and so on. The first element of *Covariances* is passed by reference. *Covariances* must be dimensioned by the caller.
Means	Output array of type Double. The vector of sample means from a sample of *NCases* observations. The first element of *Means* is passed by reference. Means must be dimensioned by the caller.
NCases	Input of type Integer. The number of observations.

The following program specifies a multivariate normal population with covariance matrix and mean given by

$$\Sigma = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}, \qquad \mu = \begin{bmatrix} 4 \\ 5 \end{bmatrix}.$$

Then it draws a sample of 500 and displays the sample covariance matrix and mean vector. The unbiased estimate of the covariance matrix is displayed because the default value of SecondMomentsType 429 is UNBIASED.

```
Module MainModule
   Sub Main()
      Dim arand As New AMOSRANDOMLib6.AmosRanGen
      Dim PopCovariances(2) As Double
      Dim PopMeans(1) As Double
      PopCovariances(0) = 3
      PopCovariances(1) = 1
      PopCovariances(2) = 2
      PopMeans(0) = 4
      PopMeans(1) = 5
      Call arand.SpecifyPopulation(PopCovariances, PopMeans)
      Dim SampleCovariances(2) As Double
      Dim SampleMeans(1) As Double

      Call arand.RandomMoments(SampleCovariances(0), SampleMeans(0), 2, 500)

      Dim ad As New AmosDebug.AmosDebug
      Call ad.PrintTriangle(SampleCovariances, "Sample Covariances")
      Call ad.PrintX(SampleMeans, "Sample Means")
   End Sub
End Module
```

4.5.5.1.2.11 RandomVector Method

Gets a random vector from the population specified by SpecifyPopulation 444.

Syntax

object.**RandomVector** (*X*)

The **RandomVector** method syntax has the following parts:

Part	Description
object	An object of type **AmosRanGen**.
X	Output array of type Double. The random vector. The first element of *X* is passed by reference. It must be dimensioned by the caller.

The following program specifies a multivariate normal population with covariance matrix and mean given by

$$\Sigma = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}, \qquad \mu = \begin{bmatrix} 4 \\ 5 \end{bmatrix}.$$

Then it generates a sample of 20 cases from that population.

```
Imports System.Diagnostics
Module MainModule
   Sub Main()
      Dim arand As New AMOSRANDOMLib6.AmosRanGen
      Dim Covariances(2) As Double
      Dim Means(1) As Double
      Covariances(0) = 3
      Covariances(1) = 1
      Covariances(2) = 2
      Means(0) = 4
      Means(1) = 5
      Call arand.SpecifyPopulation(Covariances, Means)

      Dim x(1) As Double
      Dim i As Long
      For i = 1 To 20
         Call arand.RandomVector(x(0), 2)
         Debug.WriteLine(x(0) & x(1))
      Next
   End Sub
End Module
```

4.5.5.1.2.12 RestoreState Method

The **RestoreState** method is no longer supported. The random number generator can be placed in a reproducible state by using the NextSeed 449 method of the **CAmosSeedManager** class to obtain a seed, and then using the Initialize 432 method of the **AmosRanGen** class to initialize the random number generator.

4.5.5.1.2.13 RestoreStateFromFile Method

The **RestoreStateFromFile** method is no longer supported. The random number generator can be placed in a reproducible state by using the NextSeed 449 method of the **CAmosSeedManager** class to obtain a seed, and then using the Initialize 432 method of the **AmosRanGen** class to initialize the random number generator.

4.5.5.1.2.14 SaveState Method

The **SaveState** method is no longer supported. The random number generator can be placed in a reproducible state by using the NextSeed 449 method of the **CAmosSeedManager** class to obtain a seed, and then using the Initialize 432 method of the **AmosRanGen** class to initialize the random number generator.

4.5.5.1.2.15 SaveStateToFile Method

The **SaveStateToFile** method is no longer supported. The random number generator can be placed in a

reproducible state by using the NextSeed 449 method of the **CAmosSeedManager** class to obtain a seed, and then using the Initialize 432 method of the **AmosRanGen** class to initialize the random number generator.

4.5.5.1.2.16 SpecifyPopulation Method

Specifies the population covariance matrix and mean vector used by RandomMoments 441, RandomVector 442, Rank 428, SqrDeterminant 445. and Sqrt 446.

The specified population covariance matrix must be non-negative definite.

Syntax

object.**SpecifyPopulation** (*CovarianceMatrix, MeanVector*)

The **SpecifyPopulation** method syntax has the following parts:

Part	Description
object	An object of type **AmosRanGen**.
CovarianceMatrix	Input. One-dimensional array of type Double. CovarianceMatrix is the population covariance matrix. CovarianceMatrix contains the first element of the first row, the first two elements of the second row, and so on. CovarianceMatrix is passed as a 1-dimensional array of type Double.
MeanVector	(Optional) Input. One-dimensional array of type Double. MeanVector is the population mean vector. MeanVector is passed as a 1-dimensional array of type Double. If MeanVector is omitted, the population mean vector is assumed to be zero.

The following program specifies a multivariate normal population with covariance matrix and mean given by

$$\Sigma = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}, \qquad \mu = \begin{bmatrix} 4 \\ 5 \end{bmatrix}.$$

Then it displays the square root of the determinant of the covariance matrix.

```
Imports System.Diagnostics
Module MainModule
   Sub Main()
      Dim arand As New AMOSRANDOMLib6.AmosRanGen
      Dim Covariances(2) As Double
      Dim Means(1) As Double
      Covariances(0) = 3
      Covariances(1) = 1
      Covariances(2) = 2
      Means(0) = 4
      Means(1) = 5

      Call arand.SpecifyPopulation(Covariances, Means)

      Debug.WriteLine(arand.SqrDeterminant())
   End Sub
End Module
```

4.5.5.1.2.17 SqrDeterminant Method

Gets the square root of the determinant of the covariance matrix previously specified with SpecifyPopulation (444).

Syntax

result = *object*.**SqrDeterminant** ()

The **SqrDeterminant** property syntax has the following parts:

Part	Description
result	The square root of the determinant
object	An object of type **AmosRanGen**.

The following program specifies a multivariate normal population with covariance matrix and mean given by

$$\Sigma = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}, \qquad \mu = \begin{bmatrix} 4 \\ 5 \end{bmatrix}.$$

Then it displays the square root of the determinant of the covariance matrix.

```
Imports System.Diagnostics
Module MainModule
   Sub Main()
      Dim arand As New  AMOSRANDOMLib6.AmosRanGen
      Dim Covariances(2) As Double
      Dim Means(1) As Double
      Covariances(0) = 3
      Covariances(1) = 1
      Covariances(2) = 2
      Means(0) = 4
      Means(1) = 5

      Call arand.SpecifyPopulation(Covariances, Means)

      Debug.WriteLine(arand.SqrDeterminant())
   End Sub
End Module
```

4.5.5.1.2.18 Sqrt Method

Gets the Cholesky square root of the population covariance matrix previously specified with SpecifyPopulation 444.

Syntax

object.**Sqrt** (X)

The **Sqrt** method syntax has the following parts:

Part	Description
X	Output array of type Double. The Cholesky square root. The first element of *X* is passed by reference. It must be dimensioned by the caller.
object	An object of type **AmosRanGen**.

The following program specifies a multivariate normal population with covariance matrix and mean given by

$$\Sigma = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}, \qquad \mu = \begin{bmatrix} 4 \\ 5 \end{bmatrix}.$$

Then it displays the Cholesky square root of the covariance matrix.

```
Module MainModule
    Sub Main()
        Dim arand As New AMOSRANDOMLib6.AmosRanGen
        Dim ad As New AmosDebug.AmosDebug
        Dim Covariances(2) As Double
        Dim Means(1) As Double
        Covariances(0) = 3
        Covariances(1) = 1
        Covariances(2) = 2
        Means(0) = 4
        Means(1) = 5

        Call arand.SpecifyPopulation(Covariances, Means)

        Dim SquareRoot(2) As Double

        Call arand.Sqrt(SquareRoot(0), 2)

        Call ad.PrintTriangle(SquareRoot)
    End Sub
End Module
```

4.5.5.1.2.19 SVMult Method

Multiplies a symmetric matrix times a vector.

Syntax

object.**SVMult**(*N, matrix, invector, outvector*)

The **SVMult** property syntax has the following parts:

Part	Description
object	An object of type **AmosRanGen**.
N	Input of type Integer. The number of elements in the vector.
matrix	Input array of array of length N*(N+1)/2, of type Double. matrix contains the first element of the first row, the first two elements of the second row, and so on. The first element of matrix is passed by reference.
invector	Input array of length N, of type Double. The first element of invector is passed by reference.
outvector	Output array of length N, of type Double. The product of matrix times invector. The first element of outvector is passed by reference. outvector must be dimensioned by the caller.

Remarks

outvector must be distinct from *invector*.

The following program displays the result of the matrix multiplication, $\begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} 4 \\ 5 \end{bmatrix}$.

```
Module MainModule
   Public Sub Main()
      Dim arand As New AMOSRANDOMLib6.AmosRanGen
      Dim ad As New AmosDebug.AmosDebug
      Dim A(2) As Double
      Dim x(1) As Double
      Dim y(1) As Double
      A(0) = 3
      A(1) = 1
      A(2) = 2
      x(0) = 4
      x(1) = 5

      Call arand.SVMult(2, A(0), x(0), y(0))

      ad.PrintX(y)
   End Sub
End Module
```

4.5.5.1.2.20 TimingTest Method

Generates N uniform random numbers for performance testing.

Syntax

object.**TimingTest**(*N*)

The **TimingTest** property syntax has the following parts:

Part	Description
object	An object of type **AmosRanGen**.
N	The number of random numbers to generate.

The following program reports the time needed to generate 100,000,000 random numbers.

```
Imports System
Imports System.Diagnostics
Imports Microsoft.VisualBasic
Module MainModule
   Sub Main()
      Dim arand As New AMOSRANDOMLib6.AmosRanGen
      Dim Time0 As DateTime
      Time0 = Now
      Call arand.TimingTest(100000000)
      Debug.WriteLine("Elapsed time = " & Now .Subtract(Time0).ToString)
   End Sub
End Module
```

4.5.6 CAmosSeedManager Class Reference

The **CAmosSeedManager** class provides a means of generating seeds for use with the random number generators of the **AmosRanGen** 427 class, and for maintaining a record of the seeds used by those random number generators. When you use **AmosRanGen** 427 in your programs, you can use **CAmosSeedManager** to make sure that your programs always use the same random number seed, or to make sure that they always use a different random number seed. You can also use **CAmosSeedManager** to find the random number seeds used by your programs in the past.

If you are not using Amos's built-in program editor, you need to provide a reference to **SeedManager3.exe** in order to use the **CAmosSeedManager** class. In Visual Studio 2003:

- Click **Project -> Add Reference**.
- In the **Add Reference** dialog, click **Browse**.
- When the **Select Component** dialog opens, select SeedManager3.exe from the Amos program directory and click **Open**.
- In the **Add Reference** dialog, click **OK**.

4.5.6.1 CAmosSeedManager Class Members

Methods 449

4.5.6.1.1 Methods

This section documents the methods of the CAmosSeedManager class.

4.5.6.1.1.1 NextSeed Method

Gets a 32-bit integer value that your program can use as a random number seed. The value returned by **NextSeed** is determined by a rule that you specify by clicking **Tools→Seed Manager** on the Amos Graphics menu, or by opening the Windows **Start** menu and searching for **IBM SPSS Amos 32 Seed Manager**.

Syntax

object.**NextSeed** (*AppName*)

The **NextSeed** property syntax has the following parts:

Part	Description
object	An object of type **CAmosSeedManager**.
AppName	A string that identifies your program.

The following program uses **NextSeed** to obtain a random number seed and then displays ten random numbers from a standard normal distribution. The random number seed will be saved in the **CAmosSeedManager** history log with the date and time and with the identifying string "My Program".

```
Imports System.Diagnostics
Module MainModule
    Sub Main()
        Dim SeedManager As New AmosSeedManager3.CAmosSeedManager3
        Dim ARand As New AMOSRANDOMLib6.AmosRanGen
        Dim Seed As Integer
        Seed = SeedManager.NextSeed("My Program")
        Call ARand.Initialize(Seed)

        Dim i As Long
        For i = 1 To 10
            Debug.WriteLine(ARand.NextNormal)
        Next
    End Sub
End Module
```

4.5.6.1.1.2 PersistFile Method

Gets the name of the file in which **CAmosSeedManager** keeps a history of random number seeds returned by **NextSeed**. The file may be examined in a text editor, although the history of seed values is more easily viewed by clicking **Tools→Seed Manager** on the Amos Graphics menu, or by opening the Windows **Start** menu and searching for **IBM SPSS Amos 32 Seed Manager**.

Syntax

result = *object*.**PersistFile** ()

The **PersistFile** method syntax has the following parts:

Part	Description
result	The name of the file that contains the history of random number seeds.
object	An object of type **CAmosSeedManager**.

The following program uses **PersistFile** to display the name of the file in which **CAmosSeedManager** keeps a history of random number seeds

```
Imports System.Diagnostics
Module MainModule
   Sub Main()
      Dim SeedManager As New PAmosSeedManager3.CAmosSeedManager3
      Debug.WriteLine(SeedManager.PersistFile())
   End Sub
End Module
```

4.5.7 CValue Class Reference

An object of type **CValue** is passed to the **Value** method in the **IUserValue** interface. (The **Value** method is the method that the user writes to specify a user-defined estimand.)

The **CValue** object that is passed to the **Value** method can be used to obtain information about a single sample and about estimates obtained by fitting the model to that sample. For example, a **CValue** object "knows" (for a particular sample) what the sample covariance matrix is, as well as the estimated values of all the model parameters, all the direct and indirect effects, the factor score weights, and so on. During some calls to the **Value** method, the **CValue** object provides information about the original sample. During other calls to the **Value** method, the **CValue** object provides information about a bootstrap sample.

CValue Class Members 451

4.5.7.1 CValue Class Members

The CValue 451 class members consist only of methods 451. The class has no public properties.

4.5.7.1.1 Methods

This section documents the methods of the CValue 451 class.

4.5.7.1.1.1 GetAllImpliedCorrelationsElement Method

Gets one element of the implied correlation matrix for all observed and latent variables, giving the implied correlation between two observed or latent variables. As an alternative, you can use GetAllImpliedCorrelationsMatrix 452 to get the entire matrix of implied correlations.

Syntax

result = *object*.**GetAllImpliedCorrelationsElement** (*rowVariableName, columnVariableName*)
result = *object*.**GetAllImpliedCorrelationsElement** (*rowVariable, columnVariable*)

The **GetAllImpliedCorrelationsElement** method syntax has the following parts:

Part	Description

result	The implied correlation.
object	An object of type CValue [451].
rowVariableName	(String) The name of an observed or latent variable.
columnVariableName	(String) The name of an observed or latent variable.
rowVariable	(An object of type Variable [488]) An observed or latent variable.
columnVariable	(An object of type Variable [488]) An observed or latent variable.

4.5.7.1.1.2 GetAllImpliedCorrelationsMatrix Method

Gets the implied correlation matrix for all observed and latent variables. As an alternative, you can use GetAllImpliedCorrelationsElement [451] to get the implied correlation between two observed or latent variables.

Syntax

object.**GetAllImpliedCorrelationsMatrix** (*theSymmetricMatrix, variables*)

The **GetAllImpliedCorrelationsMatrix** method syntax has the following parts:

Part	Description
object	An object of type CValue [451].
theSymmetricMatrix	The implied correlation matrix. The returned matrix contains only the non-redundant diagonal and subdiagonal elements.
variables	A list of the variables [488] that correspond to the rows (and also the columns) of the implied correlation matrix.

4.5.7.1.1.3 GetAllImpliedCovariancesElement Method

Gets one element of the implied covariance matrix for all observed and latent variables, giving the implied covariance between two observed or latent variables. As an alternative, you can use GetAllImpliedCovariancesMatrix [453] to get the entire matrix of implied covariances.

Syntax

result = *object*.**GetAllImpliedCovariancesElement** (*rowVariableName, columnVariableName*)

result = *object*.**GetAllImpliedCovariancesElement** (*rowVariable, columnVariable*)

The **GetAllImpliedCovariancesElement** method syntax has the following parts:

Part	Description
result	The implied covariance.
object	An object of type CValue [451].
rowVariableName	(String) The name of an observed or latent variable.
columnVariableName	(String) The name of an observed or latent variable.
rowVariable	(An object of type Variable [488]) An observed or latent variable.
columnVariable	(An object of type Variable [488]) An observed or latent variable.

4.5.7.1.1.4 GetAllImpliedCovariancesMatrix Method

Gets the implied covariance matrix for all observed and latent variables. As an alternative, you can use GetAllImpliedCovariancesElement [452] to get the implied covariance between two observed or latent variables or the implied variance of a single observed or latent variable.

Syntax

object.**GetAllImpliedCovariancesMatrix** (*theSymmetricMatrix, variables*)

The **GetAllImpliedCovariancesMatrix** method syntax has the following parts:

Part	Description
object	An object of type CValue [451].
theSymmetricMatrix	The implied covariance matrix. The returned matrix contains only the non-redundant diagonal and subdiagonal elements.
variables	A list of the variables [488] that correspond to the rows (and also the columns) of the implied covariance matrix.

4.5.7.1.1.5 GetAllImpliedMeansElement Method

Gets one element of the implied means vector for all observed and latent variables, giving the implied mean for a single observed or latent variable. As an alternative, you can use GetAllImpliedMeansVector [454] to get the entire vector of implied means.

Syntax

result = *object*.**GetAllImpliedMeansElement** (*variableName*)
result = *object*.**GetAllImpliedMeansElement** (*theVariable*)

The **GetAllImpliedMeansElement** method syntax has the following parts:

Part	Description
result	The implied mean.
object	An object of type CValue 451.
variableName	(String) The name of an observed or latent variable whose implied mean you want.
theVariable	(An object of type Variable 488) An observed or latent variable whose implied mean you want.

4.5.7.1.1.6 GetAllImpliedMeansVector Method

Gets the implied means vector for all observed and latent variables. As an alternative, you can use GetAllImpliedMeansElement 453 to get the implied mean of a single observed or latent variable.

Syntax

object.**GetAllImpliedMeansVector** (*theVector, variables*)

The **GetAllImpliedMeansVector** method syntax has the following parts:

Part	Description
object	An object of type CValue 451.
theVector	A vector of implied means.
variables	A list of the variables 488 that correspond to the elements of the implied means vector.

4.5.7.1.1.7 GetCorrelationList Method

Gets the correlations among exogenous variables.

Syntax

result = *object*.**GetCorrelationList** ()

The **GetCorrelationList** method syntax has the following parts:

Part	Description
result	A list of UnorderedPairAndValue [487] objects. Each UnorderedPairAndValue [487] object consists of two variables [488] and a correlation.
object	An object of type CValue [451].

4.5.7.1.1.8 GetCovarianceList Method

Gets the covariances among exogenous variables.

Syntax

result = *object*.**GetCovarianceList** ()

The **GetCovarianceList** method syntax has the following parts:

Part	Description
result	A list of UnorderedPairAndValue [487] objects. Each UnorderedPairAndValue [487] object consists of two variables [488] and a covariance.
object	An object of type CValue [451].

4.5.7.1.1.9 GetDirectEffectsElement Method

Gets one element of the matrix of direct effects, giving the direct effect of one variable on another. As an alternative, you can use GetDirectEffectsMatrix [456] to get the entire matrix of direct effects.

Syntax

result = *object*.**GetDirectEffectsElement** (*rowVariableName, columnVariableName*)
result = *object*.**GetDirectEffectsElement** (*rowVariable, columnVariable*)

The **GetDirectEffectsElement** method syntax has the following parts:

Part	Description
result	One element of the matrix of direct effects, giving the direct effect of one variable on another.
object	An object of type CValue [451].

rowVariableName	(String) The name of the "dependent" variable for which you want to estimate the direct effect of some other variable.
columnVariableName	(String) The name of the "independent" variable whose direct effect on some other variable you want to estimate.
rowVariable	(An object of type Variable 488) The "dependent" variable for which you want to estimate the direct effect of some other variable.
columnVariable	(An object of type Variable 488) The "independent" variable whose direct effect on some other variable you want to estimate.

4.5.7.1.1.10 GetDirectEffectsMatrix Method

Gets the matrix of direct effects. As an alternative, you can use GetDirectEffectsElement 455 to get the direct effect of one variable on another.

Syntax

object.**GetDirectEffectsMatrix** (*theDirectEffects*, *rowVariables*, *columnVariables*)

The **GetDirectEffectsMatrix** method syntax has the following parts:

Part	Description
object	An object of type CValue 451.
theDirectEffects	The matrix of direct effects. Each row and each column corresponds to a single model variable. The elements of the matrix give the effects of the column variables on the row variables. In other words, the column variables affect the row variables.
rowVariables	A list of the variables 488 that correspond to the rows of the matrix of direct effects.
columnVariables	A list of the variables 488 that correspond to the columns of the matrix of direct effects.

4.5.7.1.1.11 GetFactorScoreWeightsElement Method

Gets one element of the matrix of factor score weights, giving the regression weight applied to one of the observed variables in predicting one of the latent variables. As an alternative, you can use GetFactorScoreWeightsMatrix[457] to get the entire matrix of factor score weights.

Syntax

result = *object*.**GetFactorScoreWeightsElement** (*rowVariableName, columnVariableName*)
result = *object*.**GetFactorScoreWeightsElement** (*rowVariable, columnVariable*)

The **GetFactorScoreWeightsElement** method syntax has the following parts:

Part	Description
result	The regression weight.
object	An object of type CValue[451].
rowVariableName	(String) The name of one (predicted) latent variable.
columnVariableName	(String) The name of one observed variable (which acts as a predictor).
rowVariable	(An object of type Variable[488]) A (predicted) latent variable.
columnVariable	(An object of type Variable[488]) An observed variable (which acts as a predictor).

4.5.7.1.1.12 GetFactorScoreWeightsMatrix Method

Gets the matrix of factor score weights. As an alternative, you can use GetFactorScoreWeightsElement[457] to get one element of the matrix of factor score weights.

Syntax

object.**GetFactorScoreWeightsMatrix** (*theFactorScoreWeights, rowVariables, columnVariables*)

The **GetFactorScoreWeightsMatrix** method syntax has the following parts:

Part	Description
object	An object of type CValue[451].
theFactorScoreWeights	The matrix of factor score weights. The rows of the matrix correspond to the unobserved variables in the model. The

	columns correspond to the observed variables. The elements of the matrix give the regression weights for using the observed variables to predict the unobserved variables.
rowVariables	A list of the variables[488] that correspond to the rows of the matrix of factor score weights.
columnVariables	A list of the variables[488] that correspond to the columns of the matrix of factor score weights.

4.5.7.1.1.13 GetGroupName Method

Gets the name of the group for which estimates and sample moments are available during this call to the Value method.

Syntax

result = *object*.**GetGroupName** ()
result = *object*.**GetGroupName** (*groupNumber*)

The **GetGroupName** method syntax has the following parts:

Part	Description
result	The group name.
object	An object of type CValue[451].
groupNumber	A group number, where groupNumber=1 for the first group.

4.5.7.1.1.14 GetImpliedCorrelationsElement Method

Gets one element of the implied correlation matrix for the observed variables, giving the implied correlation between two observed variables. As an alternative, you can use GetImpliedCorrelationsMatrix[459] to get the entire matrix of implied correlations.

Syntax

result = *object*.**GetImpliedCorrelationsElement** (*rowVariableName, columnVariableName*)
result = *object*.**GetImpliedCorrelationsElement** (*rowVariable, columnVariable*)

The **GetImpliedCorrelationsElement** method syntax has the following parts:

Part	Description

result	The implied correlation.
object	An object of type CValue [451].
rowVariableName	(String) The name of an observed variable.
columnVariableName	(String) The name of an observed variable.
rowVariable	(An object of type Variable [488]) An observed variable.
columnVariable	(An object of type Variable [488]) An observed variable.

4.5.7.1.1.15 GetImpliedCorrelationsMatrix Method

Gets the implied correlation matrix for the observed variables in the model. As an alternative, you can use GetImpliedCorrelationsElement [458] to get the implied correlation between two observed variables.

Syntax

object.**GetImpliedCorrelationsMatrix** (*theSymmetricMatrix, variables*)

The **GetImpliedCorrelationsMatrix** method syntax has the following parts:

Part	Description
object	An object of type CValue [451].
theSymmetricMatrix	The implied correlation matrix. The returned matrix contains only the non-redundant diagonal and subdiagonal elements.
variables	A list of the variables [488] that correspond to the rows (and also the columns) of the implied correlation matrix.

4.5.7.1.1.16 GetImpliedCovariancesElement Method

Gets one element of the implied covariance matrix for the observed variables, giving the implied covariance between two observed variables. As an alternative, you can use GetImpliedCovariancesMatrix [460] to get the entire matrix of implied covariances.

Syntax

result = *object*.**GetImpliedCovariancesElement** (*rowVariableName, columnVariableName*)
result = *object*.**GetImpliedCovariancesElement** (*rowVariable, columnVariable*)

The **GetImpliedCovariancesElement** method syntax has the following parts:

Part	Description
result	The implied covariance.
object	An object of type CValue [451].
rowVariableName	(String) The name of an observed variable.
columnVariableName	(String) The name of an observed variable.
rowVariable	(An object of type Variable [488]) An observed variable.
columnVariable	(An object of type Variable [488]) An observed variable.

4.5.7.1.1.17 GetImpliedCovariancesMatrix Method

Gets the implied covariance matrix for the observed variables. As an alternative, you can use GetImpliedCovariancesElement [459] to get the implied covariance between two observed variables or the implied variance of a single observed variable.

Syntax

object.**GetImpliedCovariancesMatrix** (*theSymmetricMatrix, variables*)

The **GetImpliedCovariancesMatrix** method syntax has the following parts:

Part	Description
object	An object of type CValue [451].
theSymmetricMatrix	The implied covariance matrix. The returned matrix contains only the non-redundant diagonal and subdiagonal elements.
variables	A list of the variables [488] that correspond to the rows (and also the columns) of the implied covariance matrix.

4.5.7.1.1.18 GetImpliedMeansElement Method

Gets one element of the implied means vector for the observed variables, giving the implied mean for a single observed variable. As an alternative, you can use GetImpliedMeansVector [461] to get the entire vector of implied means.

Syntax

result = *object*.**GetImpliedMeansElement** (*variableName*)

result = *object*.**GetImpliedMeansElement** (*theVariable*)

The **GetImpliedMeansElement** method syntax has the following parts:

Part	Description
result	The implied mean.
object	An object of type CValue 451.
variableName	(String) The name of an observed variable whose implied mean you want.
theVariable	(An object of type Variable 488) An observed variable whose implied mean you want.

4.5.7.1.1.19 GetImpliedMeansVector Method

Gets the implied means vector for the observed variables. As an alternative, you can use GetImpliedMeansElement 460 to get the implied mean of a single observed variable.

Syntax

object.**GetImpliedMeansVector** (*theVector, variables*)

The **GetImpliedMeansVector** method syntax has the following parts:

Part	Description
object	An object of type CValue 451.
theVector	A vector of implied means.
variables	A list of the variables 488 that correspond to the elements of the implied means vector.

4.5.7.1.1.20 GetIndirectEffectsElement Method

Gets one element of the matrix of indirect effects, giving the indirect effect of one variable on another. As an alternative, you can use GetIndirectEffectsMatrix 462 to get the entire matrix of indirect effects.

Syntax

result = *object*.**GetIndirectEffectsElement** (*rowVariableName, columnVariableName*)
result = *object*.**GetIndirectEffectsElement** (*rowVariable, columnVariable*)

The **GetIndirectEffectsElement** method syntax has the following parts:

Part	Description
result	One element of the matrix of indirect effects, giving the indirect effect of one variable on another.
object	An object of type CValue [451].
rowVariableName	(String) The name of the "dependent" variable for which you want to estimate the indirect effect of some other variable.
columnVariableName	(String) The name of the "independent" variable whose indirect effect on some other variable you want to estimate.
rowVariable	(An object of type Variable [488]) The "dependent" variable for which you want to estimate the indirect effect of some other variable.
columnVariable	(An object of type Variable [488]) The "independent" variable whose indirect effect on some other variable you want to estimate.

4.5.7.1.1.21 GetIndirectEffectsMatrix Method

Gets the matrix of indirect effects. As an alternative, you can use GetIndirectEffectsElement [461] to get the indirect effect of one variable on another.

Syntax

object.**GetIndirectEffectsMatrix** (*theIndirectEffects, rowVariables, columnVariables*)

The **GetIndirectEffectsMatrix** method syntax has the following parts:

Part	Description
object	An object of type CValue [451].
theIndirectEffects	The matrix of indirect effects. Each row and each column corresponds to a single model variable. The elements of the matrix give the effects of the column variables on the row variables. In other words, the column variables affect the row variables.
rowVariables	A list of the variables [488] that correspond to the rows of the matrix of indirect effects.

columnVariables	A list of the variables[488] that correspond to the columns of the matrix of indirect effects.

4.5.7.1.1.22 GetInterceptList Method

Gets estimates of the intercepts in the regression equations for predicting the endogenous variables.

Syntax

result = *object*.**GetInterceptList** ()

The **GetInterceptList** method syntax has the following parts:

Part	Description
result	A list of VariableAndValue[489] objects. Each VariableAndValue[489] object consists of an endogenous Variable[488] and an intercept. Intercepts that are fixed at a constant, and therefore not estimated, are not included on the list.
object	An object of type CValue[451].

4.5.7.1.1.23 GetMeanList Method

Gets the estimated means for the exogenous variables.

Syntax

result = *object*.**GetMeanList** ()

The **GetMeanList** method syntax has the following parts:

Part	Description
result	A list of VariableAndValue[489] objects. Each VariableAndValue[489] object consists of an exogenous Variable[488] and a mean.
object	An object of type CValue[451].

4.5.7.1.1.24 GetRegressionWeightList Method

Gets the estimated regression weights.

Syntax

result = *object*.**GetRegressionWeightList** ()

The **GetRegressionWeightList** method syntax has the following parts:

Part	Description
result	A list of OrderedPairAndValue 485 objects. Each OrderedPairAndValue 485 object consists of a dependent Variable 488 (the 'to' variable), an independent Variable 488 (the 'from' variable), and a regression weight.
object	An object of type CValue 451.

4.5.7.1.1.25 GetSampleCorrelationsElement Method

Gets one element of the sample correlation matrix, giving the sample correlation between two observed variables. As an alternative, you can use GetSampleCorrelationsMatrix 464 to get the entire matrix of sample correlations.

Syntax

result = *object*.**GetSampleCorrelationsElement** (*rowVariableName, columnVariableName*)
result = *object*.**GetSampleCorrelationsElement** (*rowVariable, columnVariable*)

The **GetSampleCorrelationsElement** method syntax has the following parts:

Part	Description
result	The sample correlation.
object	An object of type CValue 451.
rowVariableName	(String) The name of an observed variable.
columnVariableName	(String) The name of an observed variable.
rowVariable	(An object of type Variable 488) An observed variable.
columnVariable	(An object of type Variable 488) An observed variable.

4.5.7.1.1.26 GetSampleCorrelationsMatrix Method

Gets the sample correlation matrix. As an alternative, you can use GetSampleCorrelationsElement 464 to get the sample correlation between two observed variables.

Syntax

object.**GetSampleCorrelationsMatrix** (*theSymmetricMatrix, variables*)

The **GetSampleCorrelationsMatrix** method syntax has the following parts:

Part	Description
object	An object of type CValue 451.
theSymmetricMatrix	The sample correlation matrix.
variables	A list of the variables 488 that correspond to the rows (and also the columns) of the sample correlation matrix.

4.5.7.1.1.27 GetSampleCovariancesElement Method

Gets one element of the sample covariance matrix, giving the sample covariance between two observed variables. As an alternative, you can use GetSampleCovariancesMatrix 466 to get the entire matrix of sample covariances.

Syntax

result = *object*.**GetSampleCovariancesElement** (*rowVariableName, columnVariableName*)
result = *object*.**GetSampleCovariancesElement** (*rowVariable, columnVariable*)

The **GetSampleCovariancesElement** method syntax has the following parts:

Part	Description
result	The sample covariance.
object	An object of type CValue 451.
rowVariableName	(String) The name of an observed variable.
columnVariableName	(String) The name of an observed variable.
rowVariable	(An object of type Variable 488) An observed variable.
columnVariable	(An object of type Variable 488) An observed variable.

4.5.7.1.1.28 GetSampleCovariancesMatrix Method

Gets the sample covariance matrix. As an alternative, you can use GetSampleCovariancesElement[465] to get the sample covariance between two variables or the sample variance of a single variable.

Syntax

object.**GetSampleCovariancesMatrix** (*theSymmetricMatrix, variables*)

The **GetSampleCovariancesMatrix** method syntax has the following parts:

Part	Description
object	An object of type CValue[451].
theSymmetricMatrix	The sample correlation matrix. The returned matrix contains only the non-redundant diagonal and subdiagonal elements.
variables	A list of the variables[488] that correspond to the rows (and also the columns) of the sample covariance matrix.

4.5.7.1.1.29 GetSampleMeansElement Method

Gets one element of the sample means vector, giving the sample mean for a single observed variable. As an alternative, you can use GetSampleMeansVector[467] to get the entire vector of sample means.

Syntax

result = *object*.**GetSampleMeansElement** (*variableName*)
result = *object*.**GetSampleMeansElement** (*theVariable*)

The **GetSampleMeansElement** method syntax has the following parts:

Part	Description
result	The sample mean.
object	An object of type CValue[451].
variableName	(String) The name of an observed variable whose sample mean you want.
theVariable	(An object of type Variable[488]) An observed variable whose sample mean you want.

4.5.7.1.1.30 GetSampleMeansVector Method

Gets the sample means vector. As an alternative, you can use GetSampleMeansElement [466] to get the sample mean of a single observed variable.

Syntax

object.**GetSampleMeansVector** (*theVector, variables*)

The **GetSampleMeansVector** method syntax has the following parts:

Part	Description
object	An object of type CValue [451].
theVector	A vector of sample means.
variables	A list of the variables [488] that correspond to the elements of the sample means vector.

4.5.7.1.1.31 GetSmc Method

Gets the squared multiple correlation for an endogenous variable.

Syntax

result = *object*.**GetIndirectEffectsElement** (*variableName*)
result = *object*.**GetIndirectEffectsElement** (*theVariable*)

The **GetIndirectEffectsElement** method syntax has the following parts:

Part	Description
result	(Double) The squared multiple correlation.
object	An object of type CValue [451].
variableName	(String) The name of an endogenous variable.
theVariable	(An object of type Variable [488]) An endogenous variable.

4.5.7.1.1.32 GetSmcList Method

Gets a list of squared multiple correlations for the endogenous variables.

Syntax

result = *object*.**GetSmcList** ()

The **GetSmcList** method syntax has the following parts:

Part	Description
result	A list of VariableAndValue 489 objects. Each VariableAndValue 489 object consists of an endogenous Variable 488 and a squared multiple correlation.
object	An object of type CValue 451.

4.5.7.1.1.33 GetStandardizedDirectEffectsElement Method

Gets one element of the matrix of standardized direct effects, giving the standardized direct effect of one variable on another. As an alternative, you can use GetStandardizedDirectEffectsMatrix 469 to get the entire matrix of standardized direct effects.

Syntax

result = *object*.**GetStandardizedDirectEffectsElement** (*rowVariableName, columnVariableName*)
result = *object*.**GetStandardizedDirectEffectsElement** (*rowVariable, columnVariable*)

The **GetStandardizedDirectEffectsElement** method syntax has the following parts:

Part	Description
result	One element of the matrix of standardized direct effects, giving the standardized direct effect of one variable on another.
object	An object of type CValue 451.
rowVariableName	(String) The name of the "dependent" variable for which you want to estimate the standardized direct effect of some other variable.
columnVariableName	(String) The name of the "independent" variable whose standardized direct effect on some other variable you want to estimate.
rowVariable	(An object of type Variable 488) The "dependent" variable for which you want to estimate the standardized direct effect of some other variable.
columnVariable	(An objet of type Variable 488) The "independent" variable whose standardized direct effect on some other variable you want to

	estimate.

4.5.7.1.1.34 GetStandardizedDirectEffectsMatrix Method

Gets the matrix of standardized direct effects. As an alternative, you can use GetStandardizedDirectEffectsElement 468 to get the standardized direct effect of one variable on another.

Syntax

object.**GetStandardizedDirectEffectsMatrix** (*theStandardizedDirectEffects, rowVariables, columnVariables*)

The **GetStandardizedDirectEffectsMatrix** method syntax has the following parts:

Part	Description
object	An object of type CValue 451.
theStandardizedDirectEffects	The matrix of standardized direct effects. Each row and each column corresponds to a single model variable. The elements of the matrix give the effects of the column variables on the row variables. In other words, the column variables affect the row variables.
rowVariables	A list of the variables 488 that correspond to the rows of the matrix of standardized direct effects.
columnVariables	A list of the variables 488 that correspond to the columns of the matrix of standardized direct effects.

4.5.7.1.1.35 GetStandardizedIndirectEffectsElement Method

Gets one element of the matrix of standardized indirect effects, giving the standardized indirect effect of one variable on another. As an alternative, you can use GetStandardizedIndirectEffectsMatrix 470 to get the entire matrix of standardized indirect effects.

Syntax

result = *object*.**GetStandardizedIndirectEffectsElement** (*rowVariableName, columnVariableName*)
result = *object*.**GetStandardizedIndirectEffectsElement** (*rowVariable, columnVariable*)

The **GetStandardizedIndirectEffectsElement** method syntax has the following parts:

Part	Description

result	One element of the matrix of standardized indirect effects, giving the standardized indirect effect of one variable on another.
object	An object of type CValue 451.
rowVariableName	(String) The name of the "dependent" variable for which you want to estimate the standardized indirect effect of some other variable.
columnVariableName	(String) The name of the "independent" variable whose standardized indirect effect on some other variable you want to estimate.
rowVariable	(An object of type Variable 488) The "dependent" variable for which you want to estimate the standardized indirect effect of some other variable.
columnVariable	(An object of type Variable 488) The "independent" variable whose standardized indirect effect on some other variable you want to estimate.

4.5.7.1.1.36 GetStandardizedIndirectEffectsMatrix Method

Gets the matrix of standardized indirect effects. As an alternative, you can use GetStandardizedIndirectEffectsElement 469 to get the standardized indirect effect of one variable on another.

Syntax

object.**GetStandardizedIndirectEffectsMatrix** (*theStandardizedIndirectEffects*, *rowVariables*, *columnVariables*)

The **GetStandardizedIndirectEffectsMatrix** method syntax has the following parts:

Part	Description
object	An object of type CValue 451.
theStandardizedIndirectEffects	The matrix of standardized indirect effects. Each row and each column corresponds to a single model variable. The elements of the matrix give the effects of the column variables on the row variables. In other words, the column variables affect the row variables.

rowVariables	A list of the variables[488] that correspond to the rows of the matrix of standardized indirect effects.
columnVariables	A list of the variables[488] that correspond to the columns of the matrix of standardized indirect effects.

4.5.7.1.1.37 GetStandardizedRegressionWeightList Method

Gets the standardized regression weights.

Syntax

result = *object*.**GetStandardizedRegressionWeightList** ()

The **GetStandardizedRegressionWeightList** method syntax has the following parts:

Part	Description
result	A list of OrderedPairAndValue[485] objects. Each OrderedPairAndValue[485] object consists of a dependent Variable[488] (the 'to' variable), an independent Variable[488] (the 'from' variable), and a standardized regression weight.
object	An object of type CValue[451].

4.5.7.1.1.38 GetStandardizedTotalEffectsElement Method

Gets one element of the matrix of standardized total effects, giving the standardized total effect of one variable on another. As an alternative, you can use GetStandardizedTotalEffectsMatrix[472] to get the entire matrix of standardized total effects.

Syntax

result = *object*.**GetStandardizedTotalEffectsElement** (*rowVariableName, columnVariableName*)
result = *object*.**GetStandardizedTotalEffectsElement** (*rowVariable, columnVariable*)

The **GetStandardizedTotalEffectsElement** method syntax has the following parts:

Part	Description
result	One element of the matrix of standardized total effects, giving the standardized total effect of one variable on another.
object	An object of type CValue[451].

rowVariableName	(String) The name of the "dependent" variable for which you want to estimate the standardized total effect of some other variable.
columnVariableName	(String) The name of the "independent" variable whose standardized total effect on some other variable you want to estimate.
rowVariable	(An object of type Variable 488) The "dependent" variable for which you want to estimate the standardized total effect of some other variable.
columnVariable	(An object of type Variable 488) The "independent" variable whose standardized total effect on some other variable you want to estimate.

4.5.7.1.1.39 GetStandardizedTotalEffectsMatrix Method

Gets the matrix of standardized total effects. As an alternative, you can use GetStandardizedTotalEffectsElement 471 to get the standardized total effect of one variable on another.

Syntax

object.**GetStandardizedTotalEffectsMatrix** (*theStandardizedTotalEffects, rowVariables, columnVariables*)

The **GetStandardizedTotalEffectsMatrix** method syntax has the following parts:

Part	Description
object	An object of type CValue 451.
theStandardizedTotalEffects	The matrix of standardized total effects. Each row and each column corresponds to a single model variable. The elements of the matrix give the effects of the column variables on the row variables. In other words, the column variables affect the row variables.
rowVariables	A list of the variables 488 that correspond to the rows of the matrix of standardized total effects.
columnVariables	A list of the variables 488 that correspond to the columns of the matrix of standardized total effects.

4.5.7.1.1.40 GetTotalEffectsElement Method

Gets one element of the matrix of total effects, giving the total effect of one variable on another. As an alternative, you can use GetTotalEffectsMatrix [473] to get the entire matrix of total effects.

Syntax

result = *object*.**GetTotalEffectsElement** (*rowVariableName, columnVariableName*)
result = *object*.**GetTotalEffectsElement** (*rowVariable, columnVariable*)

The **GetTotalEffectsElement** method syntax has the following parts:

Part	Description
result	One element of the matrix of total effects, giving the total effect of one variable on another.
object	An object of type CValue [451].
rowVariableName	(String) The name of the "dependent" variable for which you want to estimate the total effect of some other variable.
columnVariableName	(String) The name of the "independent" variable whose total effect on some other variable you want to estimate.
rowVariable	(An object of type Variable [488]) The "dependent" variable for which you want to estimate the total effect of some other variable.
columnVariable	(An object of type Variable [488]) The "independent" variable whose total effect on some other variable you want to estimate.

4.5.7.1.1.41 GetTotalEffectsMatrix Method

Gets the matrix of total effects. As an alternative, you can use GetTotalEffectsElement [473] to get the total effect of one variable on another.

Syntax

object.**GetTotalEffectsMatrix** (*theTotalEffects, rowVariables, columnVariables*)

The **GetTotalEffectsMatrix** method syntax has the following parts:

Part	Description
object	An object of type CValue [451].

theTotalEffects	The matrix of total effects. Each row and each column corresponds to a single model variable. The elements of the matrix give the effects of the column variables on the row variables. In other words, the column variables affect the row variables.
rowVariables	A list of the variables[488] that correspond to the rows of the matrix of total effects.
columnVariables	A list of the variables[488] that correspond to the columns of the matrix of total effects.

4.5.7.1.1.42 GetVarianceList Method

Gets the variances of exogenous variables.

Syntax

result = *object*.**GetVarianceList** ()

The **GetVarianceList** method syntax has the following parts:

Part	Description
result	A list of VariableAndValue[489] objects. Each VariableAndValue[489] object consists of an exogenous Variable[488] and a variance.
object	An object of type CValue[451].

4.5.7.1.1.43 IsModelingMeansAndIntercepts Method

Gets a boolean value that indicates whether means and intercepts are explicit model parameters.

Syntax

result = *object*.**IsModelingMeansAndIntercepts** ()

The **IsModelingMeansAndIntercepts** method syntax has the following parts:

Part	Description
result	True if means and intercepts are explicit model parameters, false otherwise.

object	An object of type CValue [451].

4.5.7.1.1.44 ListOfEndogenousVariables Method

Gets a list of the endogenous variables in the model.

Syntax

result = *object*.**ListOfEndogenousVariables** ()

The **ListOfEndogenousVariables** method syntax has the following parts:

Part	Description
result	The list of endogenous variables [488] in the model.
object	An object of type CValue [451].

4.5.7.1.1.45 ListOfObservedVariables Method

Gets a list of the observed variables in the model.

Syntax

result = *object*.**ListOfObservedVariables** ()

The **ListOfObservedVariables** method syntax has the following parts:

Part	Description
result	The list of observed variables [488] in the model.
object	An object of type CValue [451].

4.5.7.1.1.46 ListOfUnobservedVariables Method

Gets a list of the unobserved variables in the model.

Syntax

result = *object*.**ListOfUnobservedVariables** ()

The **ListOfUnobservedVariables** method syntax has the following parts:

Part	Description
result	The list of unobserved variables [488] in the model.

object	An object of type CValue [451].

4.5.7.1.1.47 NumberOfGroups Method

Gets the number of groups.

Syntax

result = *object*.**NumberOfGroups** ()

The **NumberOfGroups** method syntax has the following parts:

Part	Description
result	The number of groups.
object	An object of type CValue [451].

4.5.7.1.1.48 NumberOfParameters Method

Gets the number of parameters.

Syntax

result = *object*.**NumberOfParameters** ()

The **NumberOfParameters** method syntax has the following parts:

Part	Description
result	The number of parameters.
object	An object of type CValue [451].

4.5.7.1.1.49 ParameterName Method

Gets the name of a parameter.

Syntax

result = *object*.**ParameterName** (*parameterIndex*)

The **ParameterName** method syntax has the following parts:

Part	Description

result	A parameter name, which will be an empty string for parameters that are not named.
object	An object of type CValue [451].
parameterIndex	An integer index that identifies a parameter uniquely. parameterIndex ranges from 1 to n, where n is the number of parameters.

4.5.7.1.1.50 ParameterNumber Method

Gets an integer index that identifies a parameter uniquely.

Syntax

result = *object*.**ParameterNumber** (*parameterName*)

The **ParameterNumber** method syntax has the following parts:

Part	Description
result	An integer index that identifies a parameter uniquely. The index ranges from 1 to n, where n is the number of parameters.
object	An object of type CValue [451].
parameterName	The parameter name. (Note that some parameters may be unnamed.)

4.5.7.1.1.51 ParameterValue Method

Gets a parameter value.

Syntax

result = *object*.**ParameterValue** (*parameterName*)
result = *object*.**ParameterValue** (*rowVariable, columnVariable*)

The **ParameterValue** method syntax has the following parts:

Part	Description
result	The parameter value.
object	An object of type CValue [451].

parameterName	The parameter name. (Note that some parameters may be unnamed.)
parameterIndex	An integer index that identifies a parameter uniquely. parameterIndex ranges from 1 to n, where n is the number of parameters.

4.5.7.1.1.52 ParameterVector Method

Gets a vector of all parameter values.

Syntax

object.**ParameterVector** (*theVector*)

The **ParameterVector** method syntax has the following parts:

Part	Description
object	An object of type CValue 451.
theVector	The vector of parameter values.

4.5.7.1.1.53 VariableFromName Method

Gets a Variable 488 object . A **Variable** object contains information about a variable, such as its name, whether it is exogenous, etc.

Syntax

result = *object*.**VariableFromName** (*variableName*)

The **VariableFromName** method syntax has the following parts:

Part	Description
result	An object of type Variable 488.
object	An object of type CValue 451.
variableName	(String) A variable name.

4.5.8 CValueSimple Class Reference

CValueSimple Class Members 479

4.5.8.1 CValueSimple Class Members

The CValueSimple [478] class members consist only of methods [479]. The class has no public properties.

4.5.8.1.1 Methods

This section documents the methods of the CValueSimple [478] class.

4.5.8.1.1.1 DirectEffect Method

Gets the direct effect of one variable on another.

Syntax

result = *object*.**DirectEffect** (*dependentVariable, independentVariable*)

The **DirectEffect** method syntax has the following parts:

Part	Description
result	The direct effect.
object	An object of type CValueSimple [478].
dependentVariable	(An object of type Variable [488]) The "dependent" variable.
independentVariable	(An object of type Variable [488]) The "independent" variable.

4.5.8.1.1.2 FactorScoreWeight Method

Gets the regression weight applied to one of the observed variables when all observed variables are used to predict one of the latent variables.

Syntax

result = *object*.**FactorScoreWeight** (*latentVariable, observedVariable*)

The **FactorScoreWeight** method syntax has the following parts:

Part	Description
result	The standardized total effect.
object	An object of type CValueSimple [478].
latentVariable	(An object of type Variable [488]) A (predicted) latent variable.

observedVariable	(An object of type Variable 488) An observed predictor variable.

4.5.8.1.1.3 ImpliedCorrelation Method

Gets the implied correlation between two observed or latent variables. This method cannot be used to get the implied correlation between a unique variable (such as an error variable) and another variable.

Syntax

result = *object*.**ImpliedCorrelation** (*variable1, variable2*)

The **ImpliedCorrelation** method syntax has the following parts:

Part	Description
result	The implied correlation.
object	An object of type CValueSimple 478.
variable1	(An object of type Variable 488) An observed or latent variable.
variable2	(An object of type Variable 488) An observed or latent variable.

4.5.8.1.1.4 ImpliedCovariance Method

Gets the implied covariance between two observed or latent variables. This method cannot be used to get the implied covariance between a unique variable (such as an error variable) and another variable.

Syntax

result = *object*.**ImpliedCovariance** (*variable1, variable2*)

The **ImpliedCovariance** method syntax has the following parts:

Part	Description
result	The implied covariance.
object	An object of type CValueSimple 478.
variable1	(An object of type Variable 488) An observed or latent variable.

variable2	(An object of type Variable 488) An observed or latent variable.

4.5.8.1.1.5 ImpliedMean Method

Gets the implied mean for a single observed or latent variable. This method cannot be used to get the implied mean of a unique variable (such as an error variable).

Syntax

result = *object*.**ImpliedMean** (*theVariable*)

The **ImpliedMean** method syntax has the following parts:

Part	Description
result	The implied mean.
object	An object of type CValueSimple 478.
theVariable	(An object of type Variable 488) An observed or latent variable.

4.5.8.1.1.6 IndirectEffect Method

Gets the indirect effect of one variable on another.

Syntax

result = *object*.**IndirectEffect** (*dependentVariable, independentVariable*)

The **IndirectEffect** method syntax has the following parts:

Part	Description
result	The indirect effect.
object	An object of type CValueSimple 478.
dependentVariable	(An object of type Variable 488) The "dependent" variable.
independentVariable	(An object of type Variable 488) The "independent" variable.

4.5.8.1.1.7 Intercept Method

Gets the intercept in the regression equation for predicting an endogenous variable.

Syntax

result = *object*.**Intercept** (*theVariable*)

The **Intercept** method syntax has the following parts:

Part	Description
result	The intercept.
object	An object of type CValueSimple 478.
theVariable	(An object of type Variable 488) An endogenous variable.

4.5.8.1.1.8 SampleCorrelation Method

Gets the sample correlation between two observed variables.

Syntax

result = *object*.**SampleCorrelation** (*variable1, variable2*)

The **SampleCorrelation** method syntax has the following parts:

Part	Description
result	The sample correlation.
object	An object of type CValueSimple 478.
variable1	(An object of type Variable 488) An observed variable.
variable2	(An object of type Variable 488) An observed variable.

4.5.8.1.1.9 SampleCovariance Method

Gets the sample covariance between two observed variables.

Syntax

result = *object*.**SampleCovariance** (*variable1, variable2*)

The **SampleCovariance** method syntax has the following parts:

Part	Description
result	The sample covariance.
object	An object of type CValueSimple 478.
variable1	(An object of type Variable 488) An observed variable.
variable2	(An object of type Variable 488) An observed variable.

4.5.8.1.1.10 SampleMean Method

Gets the sample mean for a single observed variable.

Syntax

result = *object*.**SampleMean** (*theVariable*)

The **SampleMean** method syntax has the following parts:

Part	Description
result	The sample mean.
object	An object of type CValueSimple 478.
theVariable	(An object of type Variable 488) An observed variable.

4.5.8.1.1.11 Smc Method

Gets the squared multiple correlation between an endogenous variable and its predictors.

Syntax

result = *object*.**Smc** (*theVariable*)

The **Smc** method syntax has the following parts:

Part	Description
result	The squared multiple correlation.
object	An object of type CValueSimple 478.
theVariable	(An object of type Variable 488) An endogenous variable.

4.5.8.1.1.12 StandardizedDirectEffect Method

Gets the standardized direct effect of one variable on another.

Syntax

result = *object*.**StandardizedDirectEffect** (*dependentVariable, independentVariable*)

The **StandardizeDirectEffect** method syntax has the following parts:

Part	Description
result	The standardized direct effect.
object	An object of type CValueSimple [478].
dependentVariable	(An object of type Variable [488]) The "dependent" variable.
independentVariable	(An object of type Variable [488]) The "independent" variable.

4.5.8.1.1.13 StandardizedIndirectEffect Method

Gets the standardized indirect effect of one variable on another.

Syntax

result = *object*.**StandardizedIndirectEffect** (*dependentVariable, independentVariable*)

The **StandardizedIndirectEffect** method syntax has the following parts:

Part	Description
result	The standardized indirect effect.
object	An object of type CValueSimple [478].
dependentVariable	(An object of type Variable [488]) The "dependent" variable.
independentVariable	(An object of type Variable [488]) The "independent" variable.

4.5.8.1.1.14 StandardizedTotalEffect Method

Gets the standardized total effect of one variable on another.

Syntax

result = *object*.**StandardizedTotalEffect** (*dependentVariable, independentVariable*)

The **StandardizedTotalEffect** method syntax has the following parts:

Part	Description
result	The standardized total effect.
object	An object of type CValueSimple [478].
dependentVariable	(An object of type Variable [488]) The "dependent" variable.
independentVariable	(An object of type Variable [488]) The "independent" variable.

4.5.8.1.1.15 TotalEffect Method

Gets the total effect of one variable on another.

Syntax

result = *object*.**TotalEffect** (*dependentVariable, independentVariable*)

The **TotalEffect** method syntax has the following parts:

Part	Description
result	The total effect.
object	An object of type CValueSimple [478].
dependentVariable	(An object of type Variable [488]) The "dependent" variable.
independentVariable	(An object of type Variable [488]) The "independent" variable.

4.5.9 The t Namespace

The t namespace contains several classes that you are likely to make use of in defining your own estimands. When writing a program to specify a user-defined estimand you can get an intelliprompt list of classes in the t namespace by typing "t." (in other words, by typing the letter 't' and then a period ('.')

4.5.9.1 OrderedPairAndValue Class Reference

An **OrderedPairAndValue** object consists of an ordered pair of variables and a numeric value associated with the pair. For example, it might consist of an independent variable, a dependent variable, and a regression weight.

4.5.9.1.1 OrderedPairAndValue Class Members

4.5.9.1.1.1 Properties

Gets a Variable [488] object that refers to an independent variable. A **Variable** object contains information about a variable, such as its name, whether it is exogenous, etc.

Syntax

result = *object*.**fromVariable**

The **fromVariable** property syntax has the following parts:

Part	Description
result	The independent Variable [488] object.
object	An object of type CValue [451].

Gets a Variable [488] object that refers to a dependent variable. A **Variable** object contains information about a variable, such as its name, whether it is exogenous, etc.

Syntax

result = *object*.**toVariable**

The **toVariable** property syntax has the following parts:

Part	Description
result	The dependent Variable [488] object.
object	An object of type CValue [451].

Gets a numeric value, either a regression weight or a standardized regression weight, that is associated with an ordered pair of variables.

Syntax

result = *object*.**value**

The **value** property syntax has the following parts:

Part	Description
result	The regression weight or standardized regression weight.
object	An object of type CValue [451].

4.5.9.2 UnorderedPairAndValue Class Reference

An **UnorderedPairAndValue** object consists of an unordered pair of variables and a numeric value associated with the pair. For example, it might consist of two variables and their correlation.

4.5.9.2.1 UnorderedPairAndValue Class Members

4.5.9.2.1.1 Properties

Gets a Variable 488 object. A **Variable** object contains information about a variable, such as its name, whether it is exogenous, etc.

Syntax

result = *object*.**variable1**

The **variable1** property syntax has the following parts:

Part	Description
result	A Variable 488 object.
object	An object of type CValue 451.

Gets a Variable 488 object. A **Variable** object contains information about a variable, such as its name, whether it is exogenous, etc.

Syntax

result = *object*.**variable2**

The **variable2** property syntax has the following parts:

Part	Description
result	A Variable 488 object.
object	An object of type CValue 451.

Gets a numeric value, such as a correlation or covariance, that is associated with an unordered pair of variables.

Syntax

result = *object*.**value**

The **value** property syntax has the following parts:

Part	Description
result	The numeric value.
object	An object of type CValue [451].

4.5.9.3 Variable Class Reference

A **Variable** object contains information about a single model variable, such as the variable's name, whether it is observed, whether it is exogenous, and so on.

4.5.9.3.1 Variable Class Members

4.5.9.3.1.1 Properties

Gets the variable's name.

Syntax

result = *object*.**Name**

The **Name** property syntax has the following parts:

Part	Description
result	Gets the variable's name.
object	An object of type CValue [451].

Gets a boolean value that indicates whether the variable is a unique variable (i.e., is unobserved and exogenous, and affects only one other variable).

Syntax

result = *object*.**IsUnique**

The **IsUnique** property syntax has the following parts:

Part	Description
result	True if the variable is a unique variable.
object	An object of type CValue [451].

Gets a boolean value that indicates whether the variable is endogenous.

Syntax

result = *object*.**IsEndogenous**

The **IsEndogenous** property syntax has the following parts:

Part	Description
result	True if the variable is endogenous.
object	An object of type CValue 451.

Gets a boolean value that indicates whether the variable is observed.

Syntax

result = *object*.**IsObserved**

The **IsObserved** property syntax has the following parts:

Part	Description
result	True if the variable is observed.
object	An object of type CValue 451.

4.5.9.4 VariableAndValue Class Reference

A **VariableAndValue** object consists of a model variable together with a numeric value that is associated with the variable. For example, a VariableAndValue object might contain a variable and its variance, or a variable and its mean.

4.5.9.4.1 VariableAndValue Class Members

4.5.9.4.1.1 Properties

4.5.9.4.1.2 variable Property

Gets a Variable 488 object. A **Variable** object contains information about a variable, such as its name, whether it is exogenous, etc.

Syntax

result = *object*.**variable**

The **variable** property syntax has the following parts:

Part	Description
result	The Variable 488 object.
object	An object of type CValue 451.

4.5.9.4.1.3 value Property

Gets a numeric value that is associated with a model variable (for example, the variable's mean or variance).

Syntax

result = *object*.**value**

The **value** property syntax has the following parts:

Part	Description
result	The numeric value.
object	An object of type CValue 451.

4.5.9.5 VariableList Class Reference

The **VariableList** class inherits from Generic.List(Of Variable). VariableList does not have any members other than the members of Generic.List(Of Variable). The VariableList class is provided only as shorthand for Generic.List(Of Variable).

4.6 Additional Programming Examples

4.6.1 Examples using the Amos Graphics classes

4.6.1.1 Use the Amos Graphics classes to calculate a new fit measure

The following plugin adds to Amos Graphics the ability to compute an additional fit measure -- the standardized root mean square residual (RMR). After you run this plugin, Amos Graphics will display the standardized root mean square residual for each analysis that it performs.

The **PreFitOptions** subroutine tells the Amos Engine that sample correlations and implied correlations are needed for calculation of the standardized RMR.

The **PostFitResults** subroutine calculates and displays the standardized RMR.

```
Imports Microsoft.VisualBasic
Imports Amos
Imports AmosEngineLib.AmosEngine.TMatrixID
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
    Implements IPlugin

    Public Function Mainsub() As Integer Implements IPlugin.Mainsub
        AddHandler Pd.PreFitOptions, AddressOf PreFitOptions
        AddHandler Pd.PostFitResults, AddressOf PostFitResults
        MsgBox("The plugin for calculating standardized RMR is now installed.",, "Standardized RMR")
    End Function

    Private Sub PreFitOptions(ByVal Sem As AmosEngineLib.AmosEngine)
        Sem.NeedEstimates(SampleCorrelations)
        Sem.NeedEstimates(ImpliedCorrelations)
    End Sub

    Private Sub PostFitResults(ByVal Sem As AmosEngineLib.AmosEngine, ByVal ModelName As String, ByVal status As Integer)
        Dim N As Integer
        Dim i As Integer
        Dim j As Integer
        Dim DTemp As Double
        Dim Sample(,) As Double
        Dim Implied(,) As Double
        Sem.GetEstimates(SampleCorrelations, Sample)
        Sem.GetEstimates(ImpliedCorrelations, Implied)
        N = UBound(Sample, 1) + 1

        DTemp = 0
        For i = 1 To N - 1
            For j = 0 To i - 1
                DTemp = DTemp + (Sample(i, j) - Implied(i, j)) ^ 2
            Next
        Next

        DTemp = System.Math.Sqrt(DTemp / (N * (N - 1) / 2))

        'Dtemp is the standardized RMR
        'Display it
        Dim message As String
        message = "Model: " & ModelName & vbCrLf
        If status = 0 Then
            message &= "Standardized RMR = " & DTemp.ToString("#.0000")
        Else
            message &= ("The model w as not successfully fitted.")
        End If
        MsgBox(message,,"Standardized RMR")
    End Sub

    Public Function Name() As String Implements IPlugin.Name
    End Function

    Public Function Description() As String Implements IPlugin.Description
    End Function
End Class
```

4.6.1.2 Use the Amos Graphics classes to change the appearance of latent variables

Running the following Amos Graphics plugin modifies the appearance of latent variables. The Undraw 155 and Draw 144 methods make the changes immediately visible. The UndoToHere 106 and UndoResume 105 methods allow you to undo all the changes carried out by the program with one press of  . The **On Error GoTo** statement is a safety measure to make sure that undo capability is restored before the program exits.

```
Imports Amos
Imports System.drawing
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
  Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    Dim x As PDElement
    Pd.UndoToHere()
    On Error Goto ErrExit

    For Each x In Pd.PDElements
      If x.IsLatentVariable Then
        x.Undraw()
        x.BorderColor = Color.Black.ToArgb
        x.FillColor = Color.White.ToArgb
        x.NameColor = Color.Red.ToArgb
        x.ParameterColor = Color.Green.ToArgb
        x.Fillstyle = 0
        x.Penwidth = 2
        x.Draw()
      End If
    Next
  ErrExit:
    Pd.UndoResume()
  End Function

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.6.1.3 Use the Amos Graphics classes to create user-defined properties

When the path diagram of Example 4 (which includes a variable called **Performance**) is in the Amos Graphics window, the following plugin uses the PropertySave 100 method to create two properties for Performance. The two properties are called **Reliability** and **Variance**. Reliability is assigned the value ".8230" and Variance is assigned the value ".0209". These numbers are in fact the reliability estimate and the sample variance for Performance reported by Warren, White and Fuller (1974) 527. When the path diagram is subsequently saved, these two properties of Performance will be saved along with it.

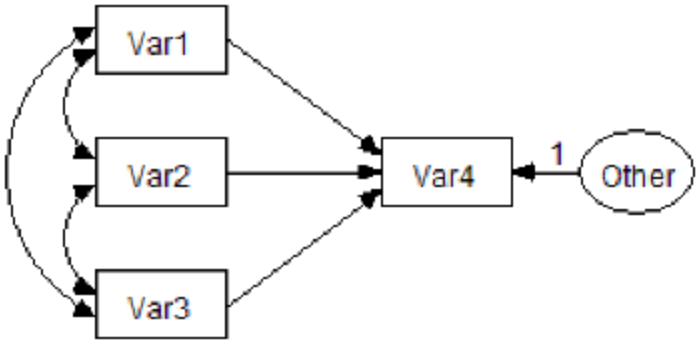
```
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim E As PDElement
      E = Pd.PDE("Performance")
      E.PropertySave("Reliability", ".8230")
      E.PropertySave("Variance", ".0209")
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

The following plugin uses the properties created by the previous plugin to compute an estimate of Performance's error variance. If Reliability or Variance is undefined, PropertyGet[99] returns the non-numeric string, "x". Attempting to perform arithmetic with the non-numeric string generates the error message "Could not compute error variance."

```
Imports Microsoft.VisualBasic
Imports Amos
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
   Implements IPlugin

   Public Function Mainsub() As Integer Implements IPlugin.Mainsub
      Dim ErrorVariance As Double
      Try
         Dim E As PDElement = Pd.PDE("Performance")
         ErrorVariance = (1 - E.PropertyGet("Reliability", "x")) _
               * E.PropertyGet("Variance", "x")
         MsgBox("Error variance = " & ErrorVariance,, "Error variance")
      Catch ex As System.Exception
         MsgBox("Could not compute error variance.",, "Error variance")
      End Try
   End Function

   Public Function Name() As String Implements IPlugin.Name
   End Function

   Public Function Description() As String Implements IPlugin.Description
   End Function
End Class
```

4.6.1.4 Use the Amos Graphics classes to draw a path diagram

The Amos Graphics plugin below draws the following path diagram.



```
Imports Microsoft.VisualBasic
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
    Implements IPlugin

    Public Function Mainsub() As Integer Implements IPlugin.Mainsub
        If FileNew() Then
            Return 0
        End If

        Dim E As PDElement

        'Coordinates of variables
        Dim Y1 As Single, Y2 As Single, Y3 As Single, Y4 As Single
        Dim X1 As Single, X2 As Single, X3 As Single
        Y1 = 1
        Y2 = 2
        Y3 = 3
        Y4 = 5
        X1 = 1
        X2 = 3
        X3 = 4.5

        'Height and width of variables
        Dim XHeight As Single, XWidth As Single
        XHeight = 0.5
        XWidth = 0.7

        'Draw the variables
        E = DiagramDraw.Observed(X1, Y1, XWidth, XHeight)
        E.NameOrCaption = "Var1"
        E.NameFontSize = 16
        E = DiagramDraw.Observed(X1, Y2, XWidth, XHeight)
        E.NameOrCaption = "Var2"
        E.NameFontSize = 16
        E = DiagramDraw.Observed(X1, Y3, XWidth, XHeight)
        E.NameOrCaption = "Var3"
        E.NameFontSize = 16
        E = DiagramDraw.Observed(X2, Y2, XWidth, XHeight)
        E.NameOrCaption = "Var4"
        E.NameFontSize = 16
        E = DiagramDraw.Unobserved(X3, Y2, XWidth, XHeight)
        E.NameOrCaption = "Other"
        E.NameFontSize = 12

        'Draw the paths
        DiagramDraw.Path("Var1", "Var4")
        DiagramDraw.Path("Var2", "Var4")
        DiagramDraw.Path("Var3", "Var4")
        E = DiagramDraw.Path("Other", "Var4")
        E.Value1 = 1
        E.ParameterFontSize = 14

        'Draw the covariances
        DiagramDraw.Covariance("var3", "var2")
        DiagramDraw.Covariance("var2", "var1")
        DiagramDraw.Covariance("var3", "var1")
```

```
    'Improve the path diagram's appearance
    EditSelectAll()
    EditTouchUp("Var1")
    Pd.Refresh()
    EditTouchUp("Var1")
    EditDeselectAll()
    EditFitToPage()
  End Function

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.6.1.5 Use the Amos Graphics classes to draw double-headed arrows

The following Amos Graphics plugin draws double-headed arrows among the selected exogenous variables. (Use  or  beforehand to select the exogenous variables that you want to be correlated.)

```
Imports Microsoft.VisualBasic
Imports Amos
Imports AmosEngineLib.AmosEngine.TMatrixID
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
    Implements IPlugin

    Public Function Mainsub() As Integer Implements IPlugin.Mainsub
        Dim SelectedEx As New Collection
        Dim x As PDElement
        Dim i As Integer
        Dim j As Integer

        'Construct the collection of selected, exogenous variables
        For Each x In Pd.PDElements
            If x.IsSelected And x.IsExogenousVariable Then
                SelectedEx.Add(x)
            End If
        Next

        'Draw the double-headed arrows
        For i = 1 To SelectedEx.Count
            For j = i + 1 To SelectedEx.Count
                x = Pd.DiagramDrawCovariance(SelectedEx(i), SelectedEx(j))
                x.IsSelected = True
            Next
        Next

        'Try to improve the appearance of the path diagram
        If SelectedEx.Count > 0 Then
            Pd.EditTouchUp(SelectedEx(1))
            Pd.EditTouchUp(SelectedEx(1))
        End If

        'The user will probably want to modify the curvature of the new
        'double-headed arrows.
        Pd.EditShapeOfObject()

    End Function

    Public Function Name() As String Implements IPlugin.Name
    End Function

    Public Function Description() As String Implements IPlugin.Description
    End Function
End Class
```

4.6.1.6 Use the Amos Graphics classes to name unobserved variables

The following Amos Graphics plugin assigns names to any unnamed, unobserved variables. Latent variables are given names like "F1", "F2", and so forth. Unique variables are given names like "e1", "e2", and so forth.

```
Imports Microsoft.VisualBasic
Imports Amos
Imports AmosEngineLib.AmosEngine.TMatrixID
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
    Implements IPlugin

    Public Function Mainsub() As Integer Implements IPlugin.Mainsub
        Const UniquePrefix = "e"
        Const LatentPrefix = "F"
        Dim UniqueCounter As Integer, LatentCounter As Integer
        UniqueCounter = LargestVariableNumber(UniquePrefix)
        LatentCounter = LargestVariableNumber(LatentPrefix)
        Dim E As PDElement
        For Each E In Pd.PDElements
            If E.NameOrCaption = "" Then
                If E.IsUnobservedVariable Then
                    'E.Undraw ()
                    If E.IsUniqueVariable Then
                        UniqueCounter += 1
                        E.NameOrCaption = UniquePrefix & UniqueCounter
                    Else
                        LatentCounter += 1
                        E.NameOrCaption = LatentPrefix & LatentCounter
                    End If
                    'E.Draw ()
                End If
            End If
        Next
        Pd.Refresh()
    End Function

    'Examine all variable names of the form, Prefix<Number>.
    'For example, if Prefix is "X", then examine variable
    'names like "X1", "X2", "X001", "X25", etc.
    'Return the largest value of <Number>, rounded off to an integer.
    Private Function LargestVariableNumber(ByVal Prefix As String) As Integer
        Dim E As PDElement, S1 As String, S2 As String
        For Each E In Pd.PDElements

            S1 = Left$(E.NameOrCaption, Len(Prefix))
            S2 = Mid$(E.NameOrCaption, Len(Prefix) + 1)

            If UCase$(S1) = UCase$(Prefix) Then
                If IsNumeric(S2) Then
                    If S2 > LargestVariableNumber Then
                        LargestVariableNumber = S2
                    End If
                End If
            End If

        Next
    End Function

    Public Function Name() As String Implements IPlugin.Name
    End Function

    Public Function Description() As String Implements IPlugin.Description
    End Function
End Class
```

4.6.1.7 Use the Amos Graphics classes to resize all rectangles in Amos Graphics

The following Amos Graphics plugin resizes each rectangle in the path diagram so that it is 1.4 times larger than the largest observed variable name. The Undraw 155 method hides each rectangle before the size changes, and the Draw 144 method draws it after the change. The UndoToHere 106 / UndoResume 105 method pair allows you to undo the changes by pressing  .

```
Imports Amos
Imports Amos.Pd
<System.ComponentModel.Composition.Export(GetType(IPlugin))>
Public Class CustomCode
  Implements IPlugin

  Public Function Mainsub() As Integer Implements IPlugin.Mainsub
    Dim x As PDElement
    Dim LargestWidth As Single, LargestHeight As Single

    LargestWidth = 0
    LargestHeight = 0
    For Each x In PDElements
      If x.IsObservedVariable Then
        If x.NameWidth > LargestWidth Then LargestWidth = x.NameWidth
        If x.NameHeight > LargestHeight Then LargestHeight = x.NameHeight
      End If
    Next
    LargestWidth = LargestWidth * 1.4
    LargestHeight = LargestHeight * 1.4

    UndoToHere()
    If LargestWidth > 0.2 And LargestHeight > 0.1 Then
      For Each x In PDElements
        If x.IsObservedVariable Then
          x.Width = LargestWidth
          x.Height = LargestHeight
        End If
      Next
      Pd.Refresh()
    End If
    DiagramRedraw Diagram()
    UndoResume()
  End Function

  Public Function Name() As String Implements IPlugin.Name
  End Function

  Public Function Description() As String Implements IPlugin.Description
  End Function
End Class
```

4.6.2 Examples using the AmosEngine class

4.6.2.1 Use the AmosEngine class to evaluate derivatives numerically and display the results with the Amos Debug class

The following program fits the model of Example 8 and displays derivatives evaluated at the discrepancy function minimum. For purposes of comparison, approximate derivatives based on finite differences are also displayed.

```
Imports AmosEngineLib
Module MainModule
    Dim Sem As New AmosEngine
    Dim ad As New AmosDebug.AmosDebug

    Sub Main()
        Dim Originalparameters() As Double

        Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
        Sem.AStructure("visperc = (1) spatial + (1) err_v")
        Sem.AStructure("cubes = (a) spatial + (1) err_c")
        Sem.AStructure("lozenges = (b) spatial + (1) err_l")
        Sem.AStructure("paragraph = (1) verbal + (1) err_p")
        Sem.AStructure("sentence = (c) verbal + (1) err_s")
        Sem.AStructure("w ordmean = (d) verbal + (1) err_w ")

        If (Sem.FitModel() = 0) Then
            'Save parameter values so they can be restored
            Sem.ParameterVector(Originalparameters)
            TestDerivatives()
            'Restore the parameter values and display them
            Sem.PutParameterVector(Originalparameters)
            ad.PrintX(Originalparameters, "Parameter values")
        End If
        Sem.Dispose()
    End Sub

    Sub TestDerivatives()
        Dim Ind As Integer, F As Double
        Dim G() As Double, GNumeric() As Double
        Dim H() As Double, HNumeric() As Double
        Sem.Evaluate2e(Ind, F, G, H)
        If Ind <> 0 Then
            Throw (New System.exception("Could not evaluate function and derivatives."))
        End If
        NumericDerivatives(Sem, GNumeric, HNumeric)
        ad.DecimalPlaces = 8
        ad.PrintX(G, "1st Derivatives")
        ad.PrintX(GNumeric, "Numerical 1st Derivatives")
        ad.PrintTriangle(H, "2nd Derivatives")
        ad.PrintTriangle(HNumeric, "Numerical 2nd Derivatives")
    End Sub

    Sub NumericDerivatives(ByVal Sem As AmosEngine, ByRef GNumeric() As Double, ByRef HNumeric() As Double)
        Const Delta As Double = 0.0001
        Dim NParams As Integer
        Dim FPlus As Double, FMinus As Double
        Dim FPP As Double, FMM As Double
        Dim FPM As Double, FMP As Double
        Dim i As Integer, j As Integer, k As Integer
        Dim DTempi As Double, DTempj As Double
        NParams = Sem.NumberOfParameters
        ReDim GNumeric(NParams - 1)
        ReDim HNumeric(NParams * (NParams + 1) \ 2 - 1)

        '1st Derivatives
        For i = 1 To NParams
            DTempi = Sem.ParameterValue(i)
            Sem.PutParameterValue(i, DTempi + Delta)
            FPlus = Evaluate()
```

```
        Sem.PutParameterValue(i, DTempi - Delta)
        FMinus = Evaluate()
        Sem.PutParameterValue(i, DTempi)
        GNumeric(i - 1) = (FPlus - FMinus) / (2 * Delta)
    Next

    '2nd Derivatives
    k = 0
    For i = 1 To NParams
        For j = 1 To i
            DTempi = Sem.ParameterValue(i)
            DTempj = Sem.ParameterValue(j)
            If i = j Then
                FPM = Evaluate()
                FMP = FPM
                Sem.PutParameterValue(i, DTempi + 2 * Delta)
                FPP = Evaluate()
                Sem.PutParameterValue(i, DTempi - 2 * Delta)
                FMM = Evaluate()
            Else
                Sem.PutParameterValue(i, DTempi + Delta)
                Sem.PutParameterValue(j, DTempj + Delta)
                FPP = Evaluate()
                Sem.PutParameterValue(j, DTempj - Delta)
                FPM = Evaluate()
                Sem.PutParameterValue(i, DTempi - Delta)
                FMM = Evaluate()
                Sem.PutParameterValue(j, DTempj + Delta)
                FMP = Evaluate()
            End If
            Sem.PutParameterValue(i, DTempi)
            Sem.PutParameterValue(j, DTempj)

            HNumeric(k) = (FPP + FMM - FPM - FMP) / (4 * Delta ^ 2)
            k = k + 1
        Next
    Next
End Sub

Function Evaluate() As Double
    Dim Ind As Integer
    Dim F As Double
    Sem.Evaluate0(Ind, F)
    If Ind <> 0 Then
        Throw (New System.Exception("Could not evaluate function."))
    End If
    Evaluate = F
End Function
End Module
```

4.6.2.2 Use the AmosEngine class to test for scale- and location-invariance

The following program performs a crude test of scale-invariance (Browne, 1982 [509], page 75) by changing the scale of each observed variable, one variable at a time, and re-fitting the model after each change of scale. Changing the scale of a variable consists of pre- and post-multiplying the sample covariance matrix by a diagonal matrix. The model is reported to be "probably scale-invariant" if every change of scale has a small effect on the discrepancy function. If means/intercepts are explicit model parameters, an additional crude test of location-invariance is performed by changing the location of each observed variable (i.e., adding a constant to its sample mean) and re-fitting the model. The model is

reported to be "probably location- invariant" if every change of location has a small effect on the discrepancy function.

A more thorough test of scale-invariance and location-invariance could be achieved by examining the implied moments after each change of scale and location.

```
Imports System
Imports System.Diagnostics
Imports AmosEngineLib
Imports AmosEngineLib.AmosEngine.TMatrixID
Imports Microsoft.VisualBasic
Module MainModule

  Sub Main()
    Dim Sem As New AmosEngine
    Dim i As Integer, j As Integer
    Sem.NeedEstimates(SampleCovariances)
    Sem.NeedEstimates(SampleMeans)
    SpecifyModel(Sem)
    TestModelForInvariance(Sem)
    Sem.Dispose()
  End Sub

  Sub TestModelForInvariance(ByVal Sem As AmosEngine)
    'Discrepancy function values within Eps of each other are considered to be equal
    Const Eps As Double = 0.0000000001
    Const ScaleFactor As Double = 10
    Const LocationShift As Double = 1
    Dim BaselineF As Double
    Dim Covariances(,) As Double, Means(,) As Double
    Dim VariableNames() As String
    Dim NVariables As Integer
    Dim i As Integer, j As Integer
    Dim IsScaleSensitive As Boolean, IsLocationSensitive As Boolean

    If Sem.NumberOfGroups <> 1 Then
      Throw (New exception("This routine works only for 1-group models."))
    End If

    BaselineF = MinimumDiscrepancy(Sem)
    Sem.GetEstimates(SampleCovariances, Covariances)
    Sem.RowNames(SampleCovariances, VariableNames)
    NVariables = UBound(Covariances)
    Dim VectorMeans() As Double
    If Sem.IsModelingMeansAndIntercepts() Then
      Sem.GetEstimates(SampleMeans, Means)
      'Means() is now a 2-dimensional array with 1 row,
      'but a 1-dimensional array of means is required.
      ReDim VectorMeans(NVariables - 1)
      For i = 0 To NVariables - 1
        VectorMeans(i) = Means(0, i)
      Next
    End If

    'Test for scale-invariance
    For i = 0 To NVariables - 1
      ScaleOneVariable(Covariances, i, ScaleFactor)
      If Sem.IsModelingMeansAndIntercepts() Then
        Sem.PutSampleMoments(Covariances, VectorMeans)
      Else
        Sem.PutSampleCovariances(Covariances)
      End If
      If Math.Abs(MinimumDiscrepancy(Sem) - BaselineF) > Eps Then
        IsScaleSensitive = True
        Debug.WriteLine( _
        "The model is probably sensitive to the scale of " & VariableNames(i))
```

```
        End If
        ScaleOneVariable(Covariances, i, 1 / ScaleFactor)
      Next
      If Not IsScaleSensitive Then
        Debug.WriteLine("The model is probably scale-invariant.")
      End If

      'Test for location-invariance
      If Sem.IsModelingMeansAndIntercepts() Then
        Dim DTemp As Double
        For i = 0 To NVariables - 1
          DTemp = VectorMeans(i)
          VectorMeans(i) = VectorMeans(i) + LocationShift
          Sem.PutSampleMoments(Covariances, VectorMeans)
          If Math.Abs(MinimumDiscrepancy(Sem) - BaselineF) > Eps Then
          IsLocationSensitive = True
          Debug.WriteLine( _
          "Model is probably sensitive to the location of " & VariableNames(i))
          End If
          VectorMeans(i) = DTemp
        Next
        If Not IsLocationSensitive Then
          Debug.WriteLine("Model is probably location-invariant.")
        End If
      End If
    End Sub

    Sub ScaleOneVariable(ByVal ScaledCovariances(,) As Double, ByVal k As Integer, ByVal ScaleFactor As Double)
      Dim i As Integer
      Dim NVariables As Integer
      NVariables = UBound(ScaledCovariances, 1) + 1
      For i = 0 To NVariables - 1
        ScaledCovariances(i, k) = ScaledCovariances(i, k) * ScaleFactor
        ScaledCovariances(k, i) = ScaledCovariances(k, i) * ScaleFactor
      Next
    End Sub

    Function MinimumDiscrepancy(ByVal Sem As AmosEngine) As Double
      If Sem.FitModel() <> 0 Then
        Throw (New exception("Could not fit model."))
      End If
      MinimumDiscrepancy = Sem.Cmin()
    End Function

    Sub SpecifyModel(ByVal Sem As AmosEngine)
      'Example 8
      Sem.BeginGroup(AmosEngine.AmosDir & "Examples\English\UserGuide.xls", "Grnt_fem")
      Sem.AStructure("visperc = (1) spatial + (1) err_v")
      Sem.AStructure("cubes = (a) spatial + (1) err_c")
      Sem.AStructure("lozenges = (b) spatial + (1) err_l")
      Sem.AStructure("paragraph = (1) verbal + (1) err_p")
      Sem.AStructure("sentence = (c) verbal + (1) err_s")
      Sem.AStructure("w ordmean = (d) verbal + (1) err_w ")
    End Sub
End Module
```

5 References

5.1 Akaike (1973)

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In Petrov, B.N. & Csaki, F. [Eds.], *Proceedings of the 2nd International Symposium on Information Theory*. Budapest: Akademiai Kiado, 267–281.

5.2 Akaike (1978)

Akaike, H. (1978). A Bayesian analysis of the minimum AIC procedure. *Annals of the Institute of Statistical Mathematics*, *30*, 9-14.

5.3 Akaike (1987)

Akaike, H. (1987). Factor analysis and AIC. *Psychometrika*, *52*, 317–332.

5.4 Allison (2002)

Allison, P. D. (2002). *Missing data* (Vol. 136). Thousand Oaks, CA: Sage.

5.5 Anderson (1935)

Anderson, E. (1935). The irises of the Gaspe Peninsula. *Bulletin of the American Iris Society*, *59*, 2-5.

5.6 Anderson (1957)

Anderson, T.W. (1957). Maximum likelihood estimates for a multivariate normal distribution when some observations are missing. *Journal of the American Statistical Association*, *52*, 200–203.

5.7 Anderson (1984)

Anderson, T.W. (1984). *An introduction to multivariate statistical analysis*. New York: Wiley.

5.8 Arbuckle (unpublished)

Arbuckle, J.L. (unpublished). Bootstrapping and model selection for analysis of moment structures.

5.9 Arbuckle (1994a)

Arbuckle, J.L. (1994a). Advantages of model-based analysis of missing data over pairwise deletion. Presented at the RMD Conference on Causal Modeling, West Lafayette, IN.

5.10 Arbuckle (1994b)

Arbuckle, J.L. (1994b). A permutation test for analysis of covariance structures. Presented at the annual meeting of the Psychometric Society, University of Illinois, Champaign, IL.

5.11 Arbuckle (1996)

Arbuckle, J.L. (1996). Full information estimation in the presence of incomplete data. In G.A. Marcoulides & R.E. Schumacker [Eds.] *Advanced structural equation modeling.* Mahwah, New Jersey: Lawrence Erlbaum Associates.

5.12 Arminger, Stein, & Wittenberg (1999)

Arminger, G., Stein, P., & Wittenberg, J. (1999). Mixtures of conditional mean- and covariance-structure models. *Psychometrika*, *64*(4), 475-494.

5.13 Attig (1983)

Attig, M.S. (1983). The processing of spatial information by adults. Presented at the annual meeting of The Gerontological Society, San Francisco.

5.14 Beale & Little (1975)

Beale, E.M.L. & Little, R.J.A. (1975). Missing values in multivariate analysis. *Journal of the Royal Statistical Society Series B*, *37*, 129–145.

5.15 Beck (1967)

Beck, A. T. (1967). Depression: Causes and Treatment. Philadelphia, PA: University of Pennsylvania Press.

5.16 Bentler (1980)

Bentler, P.M. (1980). Multivariate analysis with latent variables: Causal modeling. *Annual Review of Psychology*, *31*, 419–456.

5.17 Bentler (1985)

Bentler, P.M. (1985). *Theory and Implementation of EQS: A Structural Equations Program*. Los Angeles: BMDP Statistical Software.

5.18 Bentler (1989)

Bentler, P. (1989). *EQS structural equations program* manual. Los Angeles, CA: BMDP Statistical Software.

5.19 Bentler (1990)

Bentler, P.M. (1990). Comparative fit indexes in structural models. *Psychological Bulletin*, *107*, 238–246

5.20 Bentler & Bonett (1980)

Bentler, P.M. & Bonett, D.G. (1980). Significance tests and goodness of fit in the analysis of covariance structures. *Psychological Bulletin*, *88*, 588–606.

5.21 Bentler & Chou (1987)

Bentler, P.M. & Chou, C. (1987). Practical issues in structural modeling. *Sociological Methods and Research, 16*, 78–117.

5.22 Bentler & Freeman (1983)

Bentler, P.M. & Freeman, E.H. (1983). Tests for stability in linear structural equation systems. *Psychometrika*, *48*, 143–145.

5.23 Bentler & Weeks (1980)

Bentler, P.M. & Weeks, D.G. (1980). Linear structural equations with latent variables. Psychometrika, *45, 289–308.*

5.24 Bentler & Woodward (1979)

Bentler, P.M. & Woodward, J.A. (1979). Nonexperimental evaluation research: Contributions of causal modeling. In Datta, L. & Perloff, R. [Eds.], *Improving Evaluations*. Beverly Hills: Sage.

5.25 Bollen (1986)

Bollen, K.A. (1986). Sample size and Bentler and Bonett's nonnormed fit index. *Psychometrika*, *51*, 375–377.

5.26 Bollen (1987)

Bollen, K.A. (1987). Outliers and improper solutions: A confirmatory factor analysis example. *Sociological Methods and Research*, *15*, 375–384.

5.27 Bollen (1989a)

Bollen, K.A. (1989a). *Structural equations with latent variables*. New York: Wiley.

5.28 Bollen (1989b)

Bollen, K.A. (1989b). A new incremental fit index for general structural equation models. *Sociological Methods and Research*, *17*, 303–316.

5.29 Bollen & Curran (2006)

Bollen, K. A., & Curran, P. J. (2006). *Latent curve models: A structural equation modeling perspective*. Hoboken, NJ: Wiley.

5.30 Bollen & Jöreskog (1985)

Bollen, K.A. & Jöreskog, K.G. (1985). Uniqueness does not imply identification: A note on confirmatory factor analysis. *Sociological Methods and Research*, 14, 155–163.

5.31 Bollen & Liang (1988)

Bollen, K.A. & Liang, J. (1988). Some properties of Hoelter's CN. *Sociological Methods and Research*, *16*, 492–503.

5.32 Bollen & Long (1993)

Bollen, K.A. & Long, J.S. [Eds.] (1993). *Testing structural equation models*. Newbury Park, CA: Sage.

5.33 Bollen & Stine (1992)

Bollen, K.A. & Stine, R.A. (1992). Bootstrapping goodness-of-fit measures in structural equation models. *Sociological Methods and Research*, *21*, 205–229.

5.34 Bolstad & Curran (2017)

Bolstad, W. M. (2004). Introduction to Bayesian Statistics. Hoboken, NJ: John Wiley and Sons.

5.35 Boomsma (1987)

Boomsma, A. (1987). The robustness of maximum likelihood estimation in structural equation models. In Cuttance, P. & Ecob, R. [Eds.] *Structural Modeling by Example: Applications in Educational, Sociological, and Behavioral Research*. Cambridge University Press, 160–188.

5.36 Botha, Shapiro & Steiger (1988)

Botha, J.D., Shapiro, A. & Steiger, J.H. (1988). Uniform indices-of-fit for factor analysis models. *Multivariate Behavioral Research*, *23*, 443–450.

5.37 Bozdogan (1987)

Bozdogan, H. (1987). Model selection and Akaike's information criterion (AIC): The general theory and its analytical extensions. *Psychometrika*, *52*, 345–370.

5.38 Brown (1983)

Brown, C.H. (1983). Asymptotic comparison of missing data procedures for estimating factor loadings. *Psychometrika*, *48(2)*, 269- 291.

5.39 Brown (1994)

Brown, R.L. (1994). Efficacy of the indirect approach for estimating structural equation models with missing data: A comparison of five methods. *Structural Equation Modeling: A Multidisciplinary Journal*, *1*, 287-316.

5.40 Browne (1982)

Browne, M.W. (1982). Covariance structures. In Hawkins, D.M. [Ed.] *Topics in applied multivariate analysis*. Cambridge: Cambridge University Press, 72–141.

5.41 Browne (1984)

Browne, M.W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology*, *37*, 62–83.

5.42 Browne & Cudeck (1989)

Browne, M.W. & Cudeck, R. (1989). Single sample cross-validation indices for covariance structures. *Multivariate Behavioral Research*, *24*, 445–455.

5.43 Browne & Cudeck (1993)

Browne, M.W. & Cudeck, R. (1993). Alternative ways of assessing model fit. In Bollen, K.A. & Long, J.S. [Eds.] *Testing structural equation models*. Newbury Park, CA: Sage, 136–162.

5.44 Browne & Mels (1992)

Browne, M.W. & Mels, G. (1992). RAMONA User's Guide. The Ohio State University, Columbus, OH.

5.45 Burnham & Anderson (2002)

Burnham, K. P., & Anderson, D. R. (2002). Model selection and multimodel inference: A practical information-theoretic approach (2nd ed.). New York: Springer-Verlag.

5.46 Burns (1999)

Burns, D. D. 1999. *Feeling good: the new mood therapy*. New York: Avon Books.

5.47 Burns (2020)

Burns, D. D. 2020. *Feeling great: the revolutionary new treatment for depression and anxiety* . Eau Claire, WI: PESI.

5.48 Byrne (1989)

Byrne, B. M. (1989). *A primer of LISREL: Basic applications and programming for confirmatory factor analytic models*. New York: Springer-Verlag.

5.49 Byrne (2016)

Byrne, B. M. (2016). *Structural equation modeling with AMOS: Basic concepts, applications and programming*, 3rd edition. Abingdon: Routledge.

5.50 Carmines & McIver (1981)

Carmines, E.G. & McIver, J.P. (1981). Analyzing models with unobserved variables. In Bohrnstedt, G.W. & Borgatta, E.F. [Eds.] *Social measurement: Current issues*. Beverly Hills: Sage.

5.51 Cattell (1966)

Cattell, R.B. (1966). The scree test for the number of factors. *Multivariate Behavioral* Research, 1, 245-276.

5.52 Celeux, Hurn & Robert (2000)

Celeux, G., Hurn, M., & Robert, C. P. (2000). Computational and inferential difficulties with mixture posterior distributions. *Journal of the American Statistical Association*, *95*(451), 957-970.

5.53 Chen, Bollen, Paxton, Curran & Kirby (2001)

Chen, F., Bollen, K. A., Paxton, P., Curran, P. J., & Kirby, J. B. (2001). Improper solutions in structural equation models: Causes, consequences, and strategies. Sociological Methods & Research, 29(4), 468-

508.

5.54 Chung, Loken & Schafer (2004)

Chung, H., Loken, E., & Schafer, J. L. (2004). Difficulties in drawing inferences with finite-mixture models: A simple example with a simple solution. *American Statistician*, *58*(2), 152-158.

5.55 Cliff (1973)

Cliff, N. (1973). Scaling. *Annual Review of Psychology*, *24*, 473–506.

5.56 Cliff (1983)

Cliff, N. (1983). Some cautions concerning the application of causal modeling methods. *Multivariate Behavioral Research*, *18*, 115–126.

5.57 Cochran (1952)

Cochran, W.G. (1952). The χ^2 test of goodness of fit. *Annals of Mathematical Statistics*, *23*, 315–345.

5.58 Collins, Schafer, & Kam (2001)

Collins, L. M., Schafer, J. L., & Kam, C.-M. (2001). A comparison of inclusive and restrictive strategies in modern missing data procedures. Psychological Methods, 6(4), 330-351.

5.59 Cook & Campbell (1979)

Cook, T.D. & Campbell, D.T. (1979). *Quasi-experimentation: Design and analysis issues for field settings*. Chicago: Rand McNally.

5.60 Croon (2002)

Croon, M. (2002). Ordering the classes. In J. A. Hagenaars & A. L. McCutcheon (Eds.), *Applied latent class analysis* (pp. 137-162). Cambridge, UK: Cambridge University Press.

5.61 Crowley & Hu (1977)

Crowley, J., & Hu, M. (1977). Covariance analysis of heart transplant data. *Journal of the American Statistical Association*, *72*, 27-36.

5.62 Cudeck & Browne (1983)

Cudeck, R. & Browne, M.W. (1983). Cross-validation of covariance structures. *Multivariate Behavioral Research*, *18*, 147–167.

5.63 Davis (1993)

Davis, W.R. (1993). The FC1 rule of identification for confirmatory factor analysis: A general sufficient condition. *Sociological Methods and Research*, *21*, 403–437.

5.64 Diaconis & Efron (1983)

Diaconis, P. & Efron, B. (1983). Computer-intensive methods in statistics. *Scientific American*, *248(5)*, 116–130.

5.65 Ding (2006)

Ding, C. (2006). Using Regression Mixture Analysis in Educational Research. *Practical Assessment Research & Evaluation*, *11*(11). Available online: http://pareonline.net/getvn.asp?v=11&n=11.

5.66 Dolker, Halperin & Divgi (1982)

Dolker, M., Halperin, S. & Divgi, D.R. (1982). Problems with bootstrapping Pearson correlations in very small samples. *Psychometrika*, *47*, 529–530.

5.67 Draper & Smith (1981)

Draper, N.R. & Smith, H. (1981). *Applied regression analysis. (2nd Ed.)* New York: Wiley.

5.68 Duncan, Duncan & Strycker (2006)

Duncan, T. E., Duncan, S. C., & Strycker, L. A. (2006). *An introduction to latent variable growth curve modeling: Concepts, issues and applications*. Mahwah, NJ: Erlbaum.

5.69 Edgington (1987)

Edgington, E.S. (1987). *Randomization Tests* (Second edition). New York: Marcel Dekker.

5.70 Efron (1979)

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *Annals of Statistics*, *7*, 1–26.

5.71 Efron (1982)

Efron, B. (1982). *The jackknife, the bootstrap and other resampling plans*. (SIAM Monograph #38) Philadelphia: Society for Industrial and Applied Mathematics.

5.72 Efron (1987)

Efron, B. (1987). Better bootstrap confidence intervals. *Journal of the American Statistical Association*, *82*, 171–185.

5.73 Efron & Gong (1983)

Efron, B. & Gong, G. (1983). A leisurely look at the bootstrap, the jackknife, and cross-validation. *American Statistician*, *37*, 36–48.

5.74 Efron & Hinkley (1978)

Efron, B. & Hinkley, D.V. (1978). Assessing the accuracy of the maximum likelihood estimator: Observed versus expected Fisher information. *Biometrika*, *65*, 457-87.

5.75 Efron & Tibshirani (1993)

Efron, B. & Tibshirani, R.J. (1993). *An introduction to the bootstrap*. New York: Chapman and Hall.

5.76 European Values Study

European Values Study Group and World Values Survey Association. EUROPEAN AND WORLD VALUES SURVEYS FOUR-WAVE INTEGRATED DATA FILE, 1981-2004, v.20060423, 2006. Aggregate File Producers: Análisis Sociológicos Económicos y Políticos (ASEP) and JD Systems (JDS), Madrid, Spain/Tilburg University, Tilburg, The Netherlands. Data Files Suppliers: Analisis Sociologicos Economicos y Politicos (ASEP) and JD Systems (JDS), Madrid, Spain/Tillburg University, Tillburg, The Netherlands/ Zentralarchiv fur Empirische Sozialforschung (ZA), Cologne, Germany:) Aggregate File Distributors: Análisis Sociológicos Económicos y Políticos (ASEP) and JD Systems (JDS), Madrid, Spain/Tillburg University, Tilburg, The Netherlands/Zentralarchiv fur Empirische Sozialforschung (ZA) Cologne, Germany.

5.77 Felson & Bohrnstedt (1979)

Felson, R.B. & Bohrnstedt, G.W. (1979). "Are the good beautiful or the beautiful good?" The relationship between children's perceptions of ability and perceptions of physical attractiveness. *Social Psychology Quarterly*, *42*, 386–392.

5.78 Fienberg (1977)

Fienberg, S. E. (1977). *The analysis of cross-classified categorical data*. Cambridge, MA: MIT Press.

5.79 Fisher (1936)

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, *7*, 179-188.

5.80 Fox (1980)

Fox, J. (1980). Effect analysis in structural equation models. *Sociological Methods and Research*, *9*, 3–28.

5.81 Fraley & Raftery (2002)

Fraley, C., & Raftery, A. E. (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, *97*(458), 611-631.

5.82 Frühwirth-Schnatter (2004)

Frühwirth-Schnatter, S. (2004). Estimating marginal likelihoods for mixture and Markov switching models using bridge sampling techniques. *The Econometrics Journal*, *7*, 143-167.

5.83 Furnival & Wilson (1974)

Furnival, G.M. & Wilson, R.W. (1974). Regression by leaps and bounds. *Technometrics*, *16*, 499-511.

5.84 Gill (2004)

Gill, J. (2004). Introduction to the Special Issue. Political Analysis, 12(4), 323-337.

5.85 Graham (2003)

Graham, J. W. (2003). Adding missing-data relevant variables to FIML-based structural equation models. Structural Equation Modeling, 10, 80-100.

5.86 Graham, et al. (1996)

Graham, J.W., Hofer, S.M. & MacKinnon, D.P. (1996). Maximizing the usefulness of data obtained with planned missing value patterns: An application of maximum likelihood procedures. *Multivariate Behavorial Research*, *31*, 197-218.

5.87 Graham, et al. (1997)

Graham, J.W., Hofer, S.M., Donaldson, S.I., Mackinnon, D.P. & Schafer, J.L. (1997). Analysis with missing data in prevention research. In K. Bryant, M. Windle & S. West [Eds.] *The science of prevention: Methodological advances from alcohol and substance abuse research.* Washington, DC: American Psychological Association.

5.88 Gelman, et al. (2004)

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A. & Rubin, D. B. (2013). *Bayesian data analysis* (3rd ed.). Boca Raton: Chapman and Hall/CRC.

5.89 Gulliksen & Tukey (1958)

Gulliksen, H. & Tukey, J.W. (1958). Reliability for the law of comparative judgment. *Psychometrika*, *23*, 95–110.

5.90 Hagenaars & McCutcheon (2002)

Hagenaars, J. A., & McCutcheon, A. L. (2002). *Applied latent class analysis*. Cambridge: Cambridge University Press.

5.91 Hamilton (1960)

Hamilton, M. (1960). A Rating Scale for Depression. Journal of Neurology Neurosurgery and Psychiatry, 23, 56-62.

5.92 Hamilton (1990)

Hamilton, L.C. (1990). *Statistics with Stata*. Pacific Grove, CA: Brooks/Cole.

5.93 Hayduk (1987)

Hayduk, L.A. (1987). *Structural equation modeling with LISREL*. Baltimore: Johns Hopkins University Press.

5.94 Hoelter (1983)

Hoelter, J.W. (1983). The analysis of covariance structures: Goodness-of-fit indices. *Sociological Methods and Research*, *11*, 325–344.

5.95 Hoeting, et al. (1999)

Hoeting, J.A., Madigan, D., Raftery, A.E. & Volinsky, C.T. (1999). Bayesian model averaging: A tutorial. *Statistical Science*, *14*, 382- 417.

5.96 Holzinger & Swineford (1939)

Holzinger, K.J. & Swineford, F.A. (1939). A study in factor analysis: The stability of a bi-factor solution. *Supplementary Educational Monographs*, No. 48. Chicago: University of Chicago, Dept. of Education.

5.97 Hoshino (2001)

Hoshino, T. (2001). Bayesian inference for finite mixtures in confirmatory factor analysis. *Behaviormetrika*, *28*(1), 37-63.

5.98 Hu & Bentler (1999)

Hu, L. & Bentler, P. (1999). Cutoff criteria for fit indices in covariance structure analysis: conventional criteria versus new alternatives. *Structural Equation Modeling, 6*, 1-55.

5.99 Hubert & Golledge (1981)

Hubert, L.J. & Golledge, R.G. (1981). A heuristic method for the comparison of related structures. *Journal of Mathematical Psychology*, *23*, 214–226.

5.100 Huitema (1980)

Huitema, B.E. (1980). *The analysis of covariance and alternatives*. New York: Wiley.

5.101 Jackman (2000)

Jackman, S. (2000). Estimation and Inference Via Bayesian Simulation: An Introduction to Markov Chain Monte Carlo. American Journal of Political Science, 44(2), 375-404.

5.102 James, Mulaik & Brett (1982)

James, L.R., Mulaik, S.A. & Brett, J.M. (1982). *Causal analysis: Assumptions, models and data*. Beverly Hills: Sage.

5.103 Jasra, Holmes & Stephens (2005)

Jasra, A., Holmes, C. C., & Stephens, D. A. (2005). Markov chain Monte Carlo methods and the label switching problem in Bayesian mixture modeling. *Statistical Science*, *20*(1), 50-67.

5.104 Jöreskog (1967)

Jöreskog, K.G. (1967). Some contributions to maximum likelihood factor analysis. *Psychometrika*, *32*, 443–482.

5.105 Jöreskog (1969)

Jöreskog, K.G. (1969). A general approach to confirmatory maximum likelihood factor analysis. *Psychometrika*, *34*, 183–202.

5.106 Jöreskog (1971)

Jöreskog, K.G. (1971). Simultaneous factor analysis in several populations. *Psychometrika*, *36*, 409–426.

5.107 Jöreskog (1979)

Jöreskog, K.G. (1979). A general approach to confirmatory maximum likelihood factor analysis with addendum.. In Jöreskog, K.G. & Sörbom, D. [Eds.] *Advances in factor analysis and structural equation models*. Cambridge, MA: Abt Books, 21–43.

5.108 Jöreskog & Sörbom (1984)

Jöreskog, K.G. & Sörbom, D. (1984). *LISREL-VI user's guide* (3rd ed.). Mooresville, IN: Scientific Software.

5.109 Jöreskog & Sörbom (1989)

Jöreskog, K.G. & Sörbom, D. (1989). *LISREL-7 user's reference guide*. Mooresville, IN: Scientific Software.

5.110 Jöreskog & Sörbom (1996)

Jöreskog, K.G. & Sörbom, D. (1996). *LISREL 8 User's reference* guide. Chicago: Scientific Software.

5.111 Kalbfleisch & Prentice (2002)

Kalbfleisch, J. D., & Prentice, R. L. (2002). *The statistical analysis of failure time data*. Hoboken, NJ: Wiley.

5.112 Kaplan (1989)

Kaplan, D. (1989). Model modification in covariance structure analysis: Application of the expected parameter change statistic. *Multivariate Behavioral Research*, *24*, 285–305.

5.113 Kendall & Stuart (1973)

Kendall, M.G. & Stuart, A. (1973). *The advanced theory of statistics* (vol. 2, 3rd edition). New York: Hafner.

5.114 Kline (2016)

Kline, R. B. (2016). Principles and practice of structural equation modeling (4th ed.). New York: The Guilford Press.

5.115 Kullback & Leibler (1951)

Kullback, S. & Leibler, R.A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, *22*, 79–86.

5.116 Lazarsfeld & Henry (1968)

Lazarsfeld, P. F., & Henry, N. W. (1968). *Latent structure analysis*. Boston: Houghton Mifflin.

5.117 Lee (2007)

Lee, S. Y. (2007). *Structural equation modeling: A Bayesian approach*. Chichester, UK: John Wiley and Sons.

5.118 Lee & Hershberger (1990)

Lee, S. & Hershberger, S. (1990). A simple rule for generating equivalent models in covariance structure modeling. *Multivariate Behavioral Research*, *25*, 313–334.

5.119 Lee & Song (2003)

Lee, S. Y. & Song, X. Y. (2003). Bayesian analysis of structural equation models with dichotomous variables. *Statistics in Medicine*, *22*, 3073-3088.

5.120 Lee & Song (2004)

Lee, S.Y. & Song, X.Y. (2004). Evaluation of the Bayesian and maximum likelihood approaches in analyzing structural equation models with small sample sizes. *Multivariate Behavioral Research*, *39*(4), 653-686.

5.121 Linhart & Zucchini (1986)

Linhart, H. & Zucchini, W. (1986). *Model selection*. New York: Wiley.

5.122 Little & Rubin (1989)

Little, R.J.A. & Rubin, D.B. (1989). The analysis of social science data with missing values. *Sociological Methods and Research*, *18*, 292–326.

5.123 Little & Rubin (2020)

Little, R.J.A. & Rubin, D.B. (2020). *Statistical analysis with missing data* (3rd ed.). Hoboken: Wiley.

5.124 Little & Schenker (1995)

Little, R.J.A. & Schenker, N. (1995). Missing data. In G. Arminger, C.C. Clogg & M.E. Sobel [Eds.] *Handbook of statistical modeling for the social and behavioral sciences.* New York: Plenum.

5.125 Loehlin & Beaujean (2017)

Loehlin, J.C. (1992). *Latent variable models: An introduction to factor, path, and structural analysis* (2nd edition). Mahwah, New Jersey: Lawrence Erlbaum Associates.

5.126 Loken (2004)

Loken, E. (2004). Using latent class analysis to model temperament types. *Multivariate Behavioral Research*, *39*(4), 625-652.

5.127 Lord (1955)

Lord, F.M. (1955). Estimation of parameters from incomplete data. *Journal of the American Statistical Association*, *50*, 870–876.

5.128 Lubke & Muthén (2005)

Lubke, G. H., & Muthén, B. (2005). Investigating population heterogeneity with factor mixture models. *Psychological Methods*, *10*(1), 21-39.

5.129 MacCallum (1986)

MacCallum, R.C. (1986). Specification searches in covariance structure modeling. *Psychological Bulletin*, *100*, 107–120.

5.130 MacCallum (1990)

MacCallum, R.C. (1990). The need for alternative measures of fit in covariance structure modeling. Multivariate *Behavioral Research, 25, 157–162*.

5.131 MacCallum, Roznowski & Necowitz (1992)

MacCallum, R.C., Roznowski, M. & Necowitz, L.B. (1992). Model modifications in covariance structure analysis: The problem of capitalization on chance. *Psychological Bulletin*, *111*, 490–504.

5.132 MacCallum, et al. (1993)

MacCallum, R.C., Wegener, D.T., Uchino, B.N. & Fabrigar, L.R. (1993). The problem of equivalent models in applications of covariance structure analysis. *Psychological Bulletin*, *114*, 185–199.

5.133 MacKay (2003)

MacKay, D. J. C. (2003). *Information theory, inference & learning algorithms*. Cambridge, UK: Cambridge University Press.

5.134 MacKinnon, Lockwood & Williams (2004)

MacKinnon, D. P., Lockwood, C. M., & Williams, J. (2004). Confidence Limits for the Indirect Effect: Distribution of the Product and Resampling Methods. Multivariate Behavioral Research, 39(1), 99-128.

5.135 Madigan & Raftery (1994)

Madigan, D. & Raftery, A.E. (1994). Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, *89*, 1535-1546.

5.136 Manly (1991)

Manly, B.F.J. (1991). *Randomization and Monte Carlo Methods in Biology*. London: Chapman and Hall.

5.137 Mantel (1967)

Mantel, N. (1967). The detection of disease clustering and a generalized regression approach. *Cancer Research*, *27*, 209–220.

5.138 Mantel & Valand (1970)

Mantel, N. & Valand, R.S. (1970). A technique of nonparametric multivariate analysis. *Biometrics*, *26*, 47–558.

5.139 Mardia (1970)

Mardia, K.V. (1970). Measures of multivariate skewness and kurtosis with applications. *Biometrika*, *57*, 519–530.

5.140 Mardia (1974)

Mardia, K.V. (1974). Applications of some measures of multivariate skewness and kurtosis in testing normality and robustness studies. *Sankhya*, Series B, *36*, 115–128.

5.141 Marsh & Hocevar (1985)

Marsh, H.W. & Hocevar, D. (1985). Application of confirmatory factor analysis to the study of self-concept: First- and higher-order factor models and their invariance across groups. *Psychological Bulletin*, *97*, 562–582.

5.142 Martin & McDonald (1975)

Martin, J. K., & McDonald, R. P. (1975). Bayesian estimation in unrestricted factor analysis: A treatment for Heywood cases. *Psychometrika*, *40*, 505-517.

5.143 Matsumoto & Nishimura (1998)

Matsumoto, M. & Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation, 8*, 3-30.

5.144 Matthai (1951)

Matthai, A. (1951). Estimation of parameters from incomplete data with application to design of sample surveys. *Sankhya*, *11*, 145–152.

5.145 McArdle & Aber (1990)

McArdle, J.J. & Aber, M.S. (1990). Patterns of change within latent variable structural equation models. In A. von Eye [Ed.] *Statistical methods in longitudinal research, Volume I: Principles and structuring change*. New York: Academic Press, 151-224.

5.146 McDonald (1978)

McDonald, R.P. (1978). A simple comprehensive model for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology*, *31*, 59–72.

5.147 McDonald (1982)

McDonald, R.P. (1982). A note on the investigation of local and global identifiability. *Psychometrika*, *47*, 101–103.

5.148 McDonald (1989)

McDonald, R.P. (1989). An index of goodness-of-fit based on noncentrality. *Journal of Classification*, *6*, 97–103.

5.149 McDonald & Krane (1977)

McDonald, R.P. & Krane, W.R. (1977). A note on local identifiability and degrees of freedom in the asymptotic likelihood ratio test. *British Journal of Mathematical and Statistical Psychology*, *30*, 198–203.

5.150 McDonald & Krane (1979)

McDonald, R.P. & Krane, W.R. (1979). A Monte-Carlo study of local identifiability and degrees of freedom in the asymptotic likelihood ratio test. *British Journal of Mathematical and Statistical Psychology*, *32*, 121–132.

5.151 McDonald & Marsh (1990)

McDonald, R.P. & Marsh, H.W. (1990). Choosing a multivariate model: Noncentrality and goodness of fit. *Psychological Bulletin*, *107*, 247-255.

5.152 Mulaik (1990)

Mulaik, S.A. (1990). An analysis of the conditions under which the estimation of parameters inflates goodness of fit indices as measures of model validity. Paper presented at the Annual Meeting, Psychometric Society, Princeton, New Jersey, June 28–30, 1990.

5.153 Mulaik, et al. (1989)

Mulaik, S.A., James, L.R., Van Alstine, J., Bennett, N., Lind, S. & Stilwell, C.D. (1989). Evaluation of goodness-of-fit indices for structural equation models. *Psychological Bulletin*, *105*, 430–445.

5.154 Muthén, Kaplan & Hollis (1987)

Muthén, B., Kaplan, D. & Hollis, M. (1987). On structural equation modeling with data that are not missing completely at random. *Psychometrika*, *52*, 431–462

5.155 Olinsky, Chen & Harlow (2003)

Olinsky, A., Chen, S., & Harlow, L. (2003). The comparitive efficacy of imputation methods for missing data in structural equation modeling. European Journal of Operational Research, 151(53-79).

5.156 Olsson (1973)

Olsson, S. (1973). *An experimental study of the effects of training on test scores and factor structure*. Uppsala, Sweden: University of Uppsala, Department of Education.

5.157 Pearl, J. (2009)

Pearl, J. (2009). *Causality: Models, Reasoning, and Inference* , 2nd Edition. Cambridge, UK: Cambridge University Press.

5.158 Pearl, J., Glymour, M. and Jewell, N.P. (2016)

Pearl, J., Glymour, M. and Jewell, N.P. (2016). *Causal Inference in Statistics* . Wiley.

5.159 Raftery (1993)

Raftery, A.E. (1993). Bayesian model selection in structural equation models. In Bollen, K.A. & Long, J.S. [Eds.] *Testing structural equation models*. Newbury Park, CA: Sage, 163–180.

5.160 Raftery (1995)

Raftery, A. (1995). Bayesian model selection in social research. In P. Marsden (Ed.), *Sociological Methodology 1995* (pp. 111-163): San Francisco.

5.161 Rigdon (1994a)

Rigdon, E.E. (1994a). Calculating degrees of freedom for a structural equation model. *Structural Equation Modeling*, *1*, 274–278.

5.162 Rigdon (1994b)

Rigdon, E.E. (1994b). Demonstrating the effects of unmodeled random measurement error. *Structural Equation Modeling*, *1*, 375–380.

5.163 Rock, Werts, Linn & Jöreskog (1977)

Rock, D.A., Werts, C.E., Linn, R.L. & Jöreskog, K.G. (1977). A maximum likelihood solution to the errors in variables and errors in equations model. *Journal of Multivariate Behavioral Research*, *12*, 187–197.

5.164 Rubin (1976)

Rubin, D.E. (1976). Inference and missing data. *Biometrika*, *63*, 581–592.

5.165 Rubin (1987)

Rubin, D.E. (1987). *Multiple imputation for nonresponse in surveys*. New York: Wiley.

5.166 Runyon & Haber (1980)

Runyon, R.P. & Haber, A. (1980). *Fundamentals of behavioral statistics*, 4th ed. Reading, Mass.: Addison-Wesley.

5.167 Salhi (1998)

Salhi, S. (1998). Heuristic search methods. In G.A. Marcoulides (Ed.) *Modern methods for business research*. Mahwah, NJ: Erlbaum, 147-175.

5.168 Saris, Satorra & Sörbom (1987)

Saris, W.E., Satorra, A. & Sörbom, D. (1987). The detection and correction of specification errors in structural equation models. In Clogg, C.C. [Ed.]. *Sociological methodology 1987*. San Francisco: Jossey-Bass.

5.169 Schafer (1997)

Schafer, J.L. (1997). *Analysis of incomplete multivariate data.* London, UK: Chapman and Hall.

5.170 Schafer & Graham (2002)

Schafer, J. L., & Graham, J. W. (2002). Missing data: Our view of the state of the art. *Psychological Methods*, *7*(2), 147-177.

5.171 Schafer & Olsen (1998)

Schafer, J. L., & Olsen, M. K. (1998). Multiple imputation for multivariate missing-data problems: A data analyst's perspective. Multivariate Behavioral Research, 33(4), 545-571.

5.172 Scheines, Hoijtink & Boomsma (1999)

Scheines, R., Hoijtink, H., & Boomsma, A. (1999). Bayesian estimation and testing of structural equation models. *Psychometrika*, *64*, 37-52.

5.173 Schwarz (1978)

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6, 461–464.

5.174 Shrout & Bolger (2002)

Shrout, P. E., & Bolger, N. (2002). Mediation in Experimental and Nonexperimental Studies: New Procedures and Recommendations. Psychological Methods, 7(4), 422-445.

5.175 Sobel (1982)

Sobel, M. E. (1982). Asymptotic confidence intervals for indirect effects in structural equation models. In S. Leinhart (Ed.), Sociological Methodology (pp. 290-312). San Francisco: Jossey-Bass.

5.176 Sobel (1986)

Sobel, M. E. (1986). Some new results on indirect effects and their standard errors in covariance structure models. In S. Leinhart (Ed.), Sociological Methodology (pp. 159-186). San Francisco: Jossey-Bass.

5.177 Sobel & Bohrnstedt (1985)

Sobel, M.E. & Bohrnstedt, G.W. (1985). Use of null models in evaluating the fit of covariance structure models. In Tuma, N.B [Ed.] *Sociological methodology 1985*. San Francisco: Jossey-Bass, 152–178.

5.178 Sörbom (1974)

Sörbom, D. (1974). A general method for studying differences in factor means and factor structure between groups. *British Journal of Mathematical and Statistical Psychology*, *27*, 229–239.

5.179 Sörbom (1978)

Sörbom, D. (1978). An alternative to the methodology for analysis of covariance. *Psychometrika*, *43*, 381–396.

5.180 Spiegelhalter, et al (2002)

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & van der Linde, A. (2002). Bayesian measures of model complexity and fit. Journal of the Royal Statistical Society Series B, 64(4), 583-639.

5.181 Spirtes, Scheines & Glymour (1990)

Spirtes, P., Scheines, R. & Glymour, C. (1990). Simulation studies of the reliability of computer-aided model specification using the TETRAD II, EQS, and LISREL programs. *Sociological Methods and Research*, *19*, 3-66.

5.182 Steiger (1989)

Steiger, J.H. (1989). *EzPATH: Causal modeling*. Evanston, IL: Systat.

5.183 Steiger (1990)

Steiger, J.H. (1990). Structural model evaluation and modification: An interval estimation approach. *Multivariate Behavioral Research*, *25*, 173–180.

5.184 Steiger & Lind (1980)

Steiger, J.H. & Lind, J.C. (1980, May 30, 1980). *Statistically-based tests for the number of common factors.* Paper presented at the Annual Spring Meeting of the Psychometric Society, Iowa City.

5.185 Steiger, Shapiro & Browne (1985)

Steiger, J.H., Shapiro, A. & Browne, M.W. (1985). On the multivariate asymptotic distribution of sequential chi-square statistics. *Psychometrika*, *50*, 253–263.

5.186 Stelzl (1986)

Stelzl, I. (1986). Changing a causal hypothesis without changing the fit: Some rules for generating equivalent path models. *Multivariate Behavioral Research*, *21*, 309–331.

5.187 Stephens (2000)

Stephens, M. (2000). Dealing with label switching in mixture models. *Journal of the Royal Statistical Society Series B*, *62*(4), 795-809.

5.188 Stine (1989)

Stine, R.A. (1989). An introduction to bootstrap methods: Examples and ideas. *Sociological Methods and Research*, *18*, 243–291.

5.189 Swain (1975)

Swain, A.J. (1975). Analysis of parametric structures for variance matrices. Unpublished Ph.D. thesis, University of Adelaide.

5.190 Tanaka & Huba (1985)

Tanaka, J.S. & Huba, G.J. (1985). A fit index for covariance structure models under arbitrary GLS estimation. *British Journal of Mathematical and Statistical Psychology*, *38*, 197–201.

5.191 Tanaka & Huba (1989)

Tanaka, J.S. & Huba, G.J. (1989). A general coefficient of determination for covariance structure models under arbitrary GLS estimation. *British Journal of Mathematical and Statistical Psychology*, *42*, 233–239.

5.192 Tucker & Lewis (1973)

Tucker, L.R & Lewis, C. (1973). A reliability coefficient for maximum likelihood factor analysis. *Psychometrika*, *38*, 1–10.

5.193 Verleye (1996)

Verleye, G. (1996). *Missing at random data problems in attitude measurements using maximum likelihood structural equation modeling*. Unpublished dissertation. Frije Universiteit Brussels, Department of Psychology.

5.194 Vermunt & Magidson (2005)

Vermunt, J. K., & Magidson, J. (2005). Structural equation models: Mixture models. In B. Everitt & D. Howell (Eds.), *Encyclopedia of statistics in behavioral science* (pp. 1922-1927). Chichester, UK: Wiley.

5.195 Warren, White & Fuller (1974)

Warren, R.D., White, J.K. & Fuller, W.A. (1974). An errors-in-variables analysis of managerial role performance. *Journal of the American Statistical Association*, *69*, 886–893.

5.196 Weatherburn (1968)

Weatherburn, C.E. (1968) *A First Course in Mathematical Statistics*. Cambridge, UK: Cambridge University Press.

5.197 Wheaton (1987)

Wheaton, B. (1987). Assessment of fit in overidentified models with latent variables. *Sociological Methods and Research*, *16*, 118–154.

5.198 Wheaton, Muthén, Alwin & Summers (1977)

Wheaton, B., Muthén, B., Alwin, D.F. & Summers, G.F. (1977). Assessing reliability and stability in panel models. In Heise, D.R. [Ed.] *Sociological methodology 1977*. San Francisco: Jossey-Bass, 84–136.

5.199 Wichman & Hill (1982)

Wichman, B.A. & Hill, I.D. (1982). An efficient and portable pseudo-random number generator. Algorithm AS 183. *Applied Statistics*, *31*, 188–190.

5.200 Winer (1971)

Winer, B.J. (1971). *Statistical principles in experimental design*. New York: McGraw-Hill.

5.201 Wing (1962)

Wing, J. K. (1962). Institutionalism in mental hospitals. *British Journal of Social and Clinical Psychology*, *1*, 38-51.

5.202 Wothke (1993)

Wothke, W. (1993). Nonpositive definite matrices in structural modeling. In Bollen, K.A. & Long, J.S. [Eds.], *Testing structural equation models* (pp. 256–293). Newbury Park, CA: Sage.

5.203 Zhu & Lee (2001)

Zhu, H. T., & Lee, S. Y. (2001). A Bayesian analysis of finite mixtures in the LISREL model. *Psychometrika*, *66*(1), 133-152.

Index

- A -

- B -

- C -

- D -

- E -

- F -

- G -

- H -

- I -

- J -

- K -

- L -

- M -

- N -

- O -

- P -

- Q -

- R -

- S -

- T -

- U -

- V -

- W -

- X -