34

An AHP Application for Choosing a Job

34.1 Introduction

This application implements the analytical hierarchy process (AHP) in the context of choosing a job. AHP is useful in many multiobjective decision problems. You list a number of criteria and a number of possible decisions that meet the criteria to various degrees. In this case, the criteria are salary, nearness to family, benefits, and possibly others, and the decisions are your available job offers. The first step in AHP is to compare the criteria—which are the most important to you? This is discovered through a series of pairwise comparisons. The jobs are then compared with each other on each criterion, again by making a series of pairwise comparisons. The final result is a score for each job, and the job with the highest score is identified as your preferred job.

New Learning Objectives: VBA

- To learn how online help can be provided on a worksheet by taking advantage of a worksheet's `BeforeDoubleClick` event.
- To learn several new controls for user forms: scrollbars, combo boxes, and command buttons other than the usual OK/Cancel combination.

New Learning Objectives: Non-VBA

- To learn the basic elements of AHP.

34.2 Functionality of the Application

The application first asks the user to specify the criteria that are relevant for making the job decision. Several criteria, such as salary, location, and benefits, are already in the list of possibilities, but the user can add other criteria to the list if desired. Next, the user is asked to list the available job offers. The user is asked to make a series of pairwise comparisons, first between pairs of criteria and then between pairs of jobs on each criterion. After all pairwise comparisons have been made, the application performs the necessary calculations for AHP and reports the results on a Report worksheet, highlighting the job with the highest score. The scores for the various jobs can also be viewed graphically. Finally, to check whether the user was internally consistent when making the pairwise comparisons (because it is easy to be inconsistent), consistency indexes are reported.

After a given AHP analysis, the user can run another analysis with the *same* criteria and jobs (by making new pairwise comparisons). Alternatively, the user can run another analysis with entirely new inputs.

34.3 Running the Application

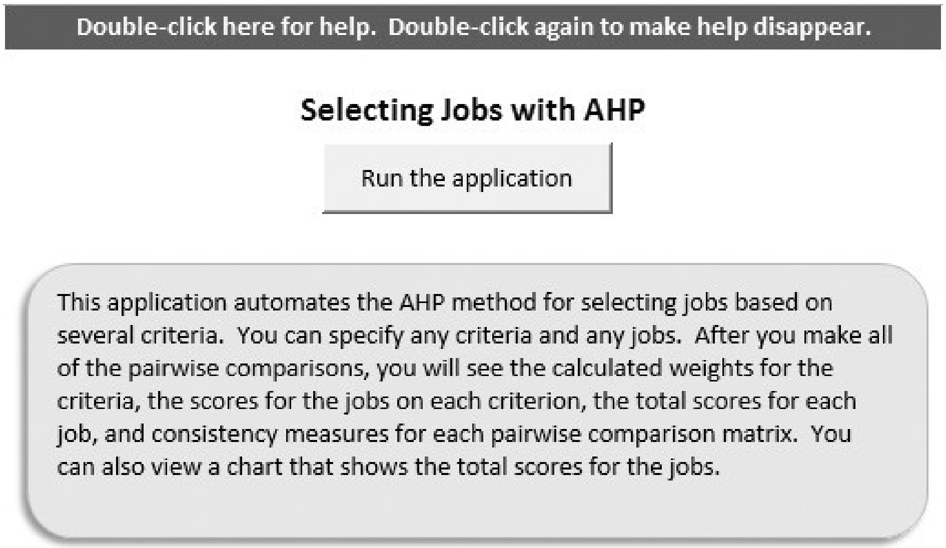
The application is stored in the file **AHP.xlsm**. When this file is opened, the Explanation worksheet in Figure 34.1 appears. Because AHP is probably not well known to most users, the application provides some help in a text box. This text box is currently hidden, but it can be displayed by double-clicking anywhere in row 1 of the Explanation worksheet. The help text box then appears, as shown in Figure 34.2. It can be hidden by again double-clicking in row 1. The way this online help is accomplished with VBA is explained later in the chapter.

Clicking the button in Figure 34.1 produces the dialog box in Figure 34.3. It has a combo box with a dropdown list of criteria the user can choose from. Alternatively, the user can type a *new* criterion in the box. After a criterion is entered in the box, the user should click the Add button to add the criterion to the list that will be used in making the decision.

When all desired criteria have been added, the user should click the No More button. The dialog box shown in Figure 34.4 then appears. It has the same functionality as the first dialog box, except that there is no dropdown list; the user must enter all available jobs, one at a time, in the text box.

After all criteria and jobs have been entered, several dialog boxes similar to the one shown in Figure 34.5 appear. Each asks the user to make a pairwise comparison between two of the criteria. This can be done by clicking the button for

Figure 34.1 Explanation Worksheet

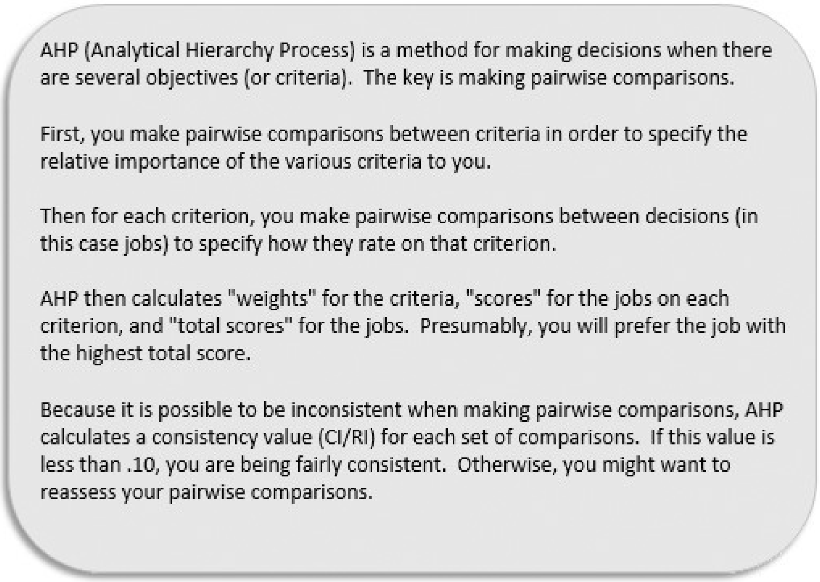


Double-click here for help. Double-click again to make help disappear.

Selecting Jobs with AHP

This application automates the AHP method for selecting jobs based on several criteria. You can specify any criteria and any jobs. After you make all of the pairwise comparisons, you will see the calculated weights for the criteria, the scores for the jobs on each criterion, the total scores for each job, and consistency measures for each pairwise comparison matrix. You can also view a chart that shows the total scores for the jobs.

Figure 34.2 Help for AHP



AHP (Analytical Hierarchy Process) is a method for making decisions when there are several objectives (or criteria). The key is making pairwise comparisons.

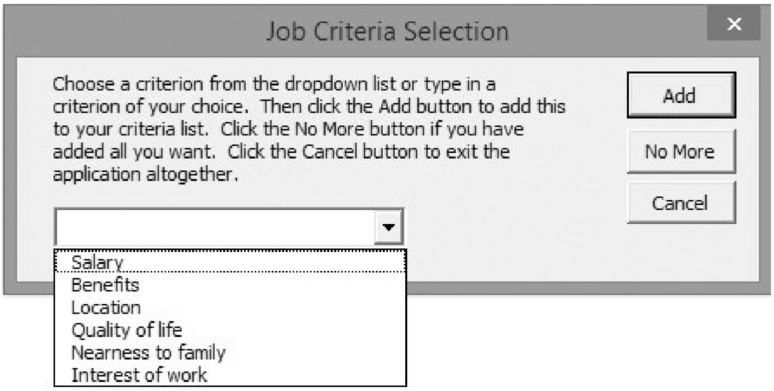
First, you make pairwise comparisons between criteria in order to specify the relative importance of the various criteria to you.

Then for each criterion, you make pairwise comparisons between decisions (in this case jobs) to specify how they rate on that criterion.

AHP then calculates "weights" for the criteria, "scores" for the jobs on each criterion, and "total scores" for the jobs. Presumably, you will prefer the job with the highest total score.

Because it is possible to be inconsistent when making pairwise comparisons, AHP calculates a consistency value (CI/RI) for each set of comparisons. If this value is less than .10, you are being fairly consistent. Otherwise, you might want to reassess your pairwise comparisons.

Figure 34.3 Dialog for Choosing Criteria

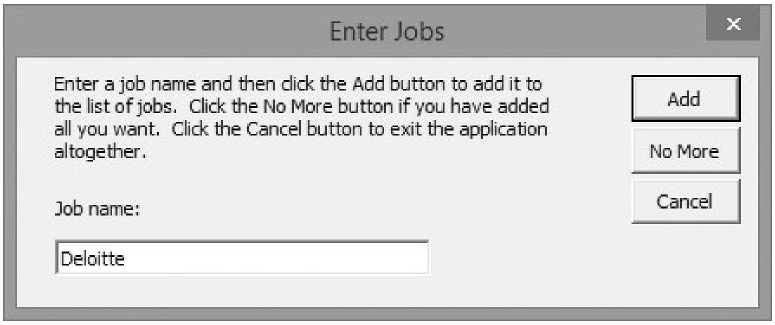


Job Criteria Selection

Choose a criterion from the dropdown list or type in a criterion of your choice. Then click the Add button to add this to your criteria list. Click the No More button if you have added all you want. Click the Cancel button to exit the application altogether.

the criterion that is considered more important and then using the scrollbar to indicate how much more important it is. The scrollbar goes in discrete one-unit steps, from 1 to 9. The labels below the scrollbar attach suggestive meanings to the numbers. Note that there can be quite a few pairwise comparisons to make. For example, if there are four criteria, there are six such pairwise comparisons (the number of ways two things can be chosen from four things). The counter on the dialog box reminds the user how many more pairwise comparisons remain.

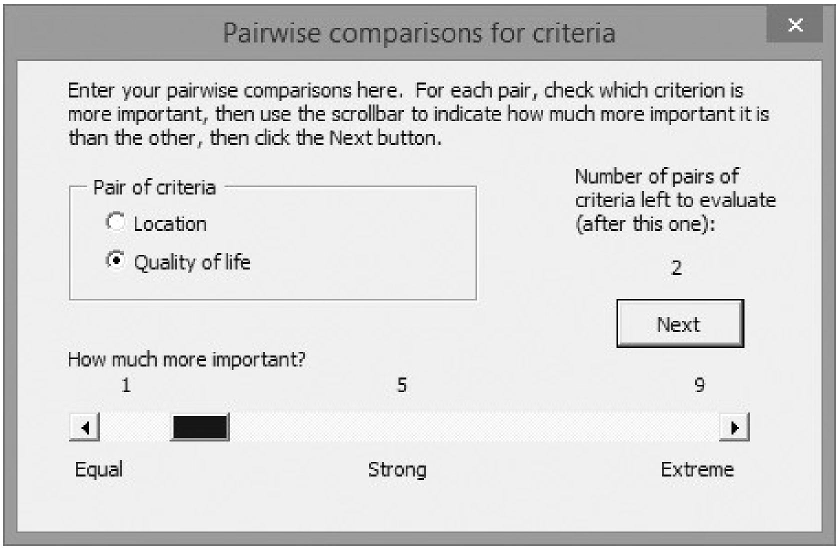
Figure 34.4 Dialog Box for Choosing Jobs



Enter a job name and then click the Add button to add it to the list of jobs. Click the No More button if you have added all you want. Click the Cancel button to exit the application altogether.

Job name: Deloitte

Figure 34.5 Pairwise Comparison Dialog Box for Criteria



Enter your pairwise comparisons here. For each pair, check which criterion is more important, then use the scrollbar to indicate how much more important it is than the other, then click the Next button.

Pair of criteria

○ Location

● Quality of life

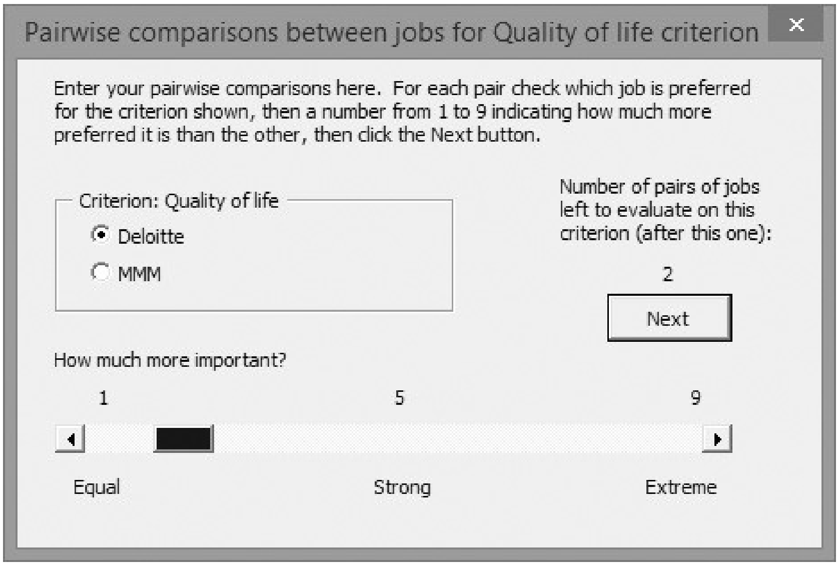
Number of pairs of criteria left to evaluate (after this one): 2

How much more important?

The application then presents a series of dialog boxes similar to those in Figures 34.6 and 34.7, where the user must make pairwise comparisons between pairs of jobs on the various criteria. Again, if there are quite a few criteria and jobs, the number of required pairwise comparisons will be large.

When all pairwise comparisons have been made, the application does the AHP calculations and reports the results in a Report worksheet, as shown in Figure 34.8. This report lists the weights for the criteria, the scores for the jobs on each criterion, and the total scores for the jobs. The job with the highest total score is boldfaced. (In this example, Microsoft is the winner.) The bottom of the report lists consistency indexes. If the user has to make many pairwise comparisons, there is a good chance of being inconsistent. This bottom section alerts the user to this possibility. Specifically, if it reports inadequate consistency, the user should probably go through the process again and attempt to make more consistent pairwise comparisons.

Figure 34.6 Pairwise Comparison Between Jobs on Salary



Pairwise comparisons between jobs for Quality of life criterion

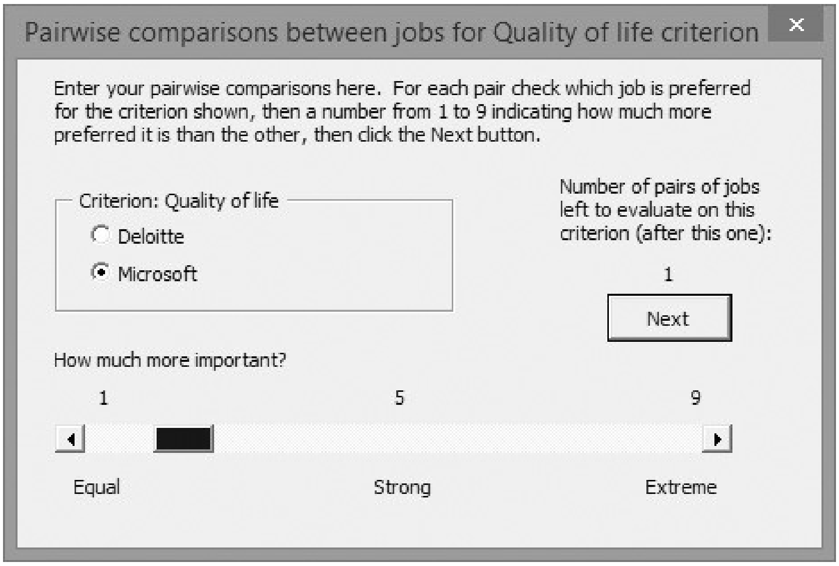
Enter your pairwise comparisons here. For each pair check which job is preferred for the criterion shown, then a number from 1 to 9 indicating how much more preferred it is than the other, then click the Next button.

Criterion: Quality of life
● Deloitte
○ MMM

Number of pairs of jobs left to evaluate on this criterion (after this one): 2

How much more important?

Figure 34.7 Pairwise Comparison Between Jobs on Location



Pairwise comparisons between jobs for Quality of life criterion

Enter your pairwise comparisons here. For each pair check which job is preferred for the criterion shown, then a number from 1 to 9 indicating how much more preferred it is than the other, then click the Next button.

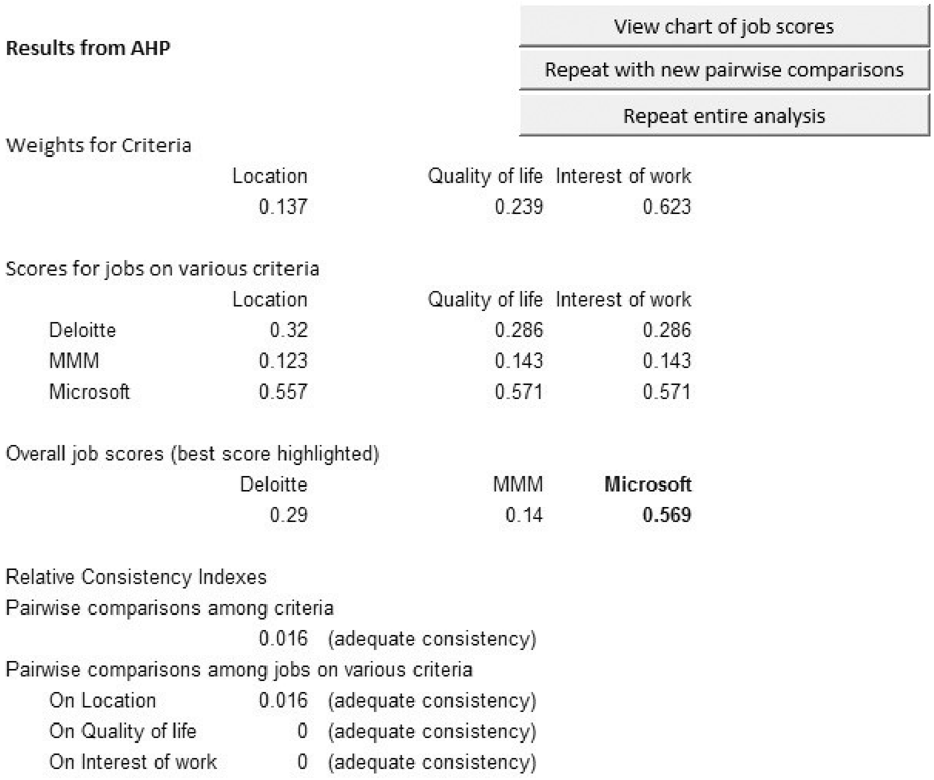
Criterion: Quality of life
○ Deloitte
● Microsoft

Number of pairs of jobs left to evaluate on this criterion (after this one): 1

How much more important?

By clicking the top button on the Report worksheet, the user can view the chart in Figure 34.9, which indicates the total scores for the jobs. The other two buttons on the Report worksheet allow the user to repeat the analysis with the same criteria and jobs (by making new pairwise comparisons) or with entirely new inputs.

Figure 34.8 Report Worksheet



Results from AHP

Weights for Criteria

Location	Quality of life	Interest of work
0.137	0.239	0.623

Scores for jobs on various criteria

	Location	Quality of life	Interest of work
Deloitte	0.32	0.286	0.286
MMM	0.123	0.143	0.143
Microsoft	0.557	0.571	0.571

Overall job scores (best score highlighted)

Deloitte	MMM	**Microsoft**
0.29	0.14	**0.569**

Relative Consistency Indexes

Pairwise comparisons among criteria

0.016 (adequate consistency)

Pairwise comparisons among jobs on various criteria

On Location	0.016	(adequate consistency)
On Quality of life	0	(adequate consistency)
On Interest of work	0	(adequate consistency)

Figure 34.9 Chart of Total Job Scores

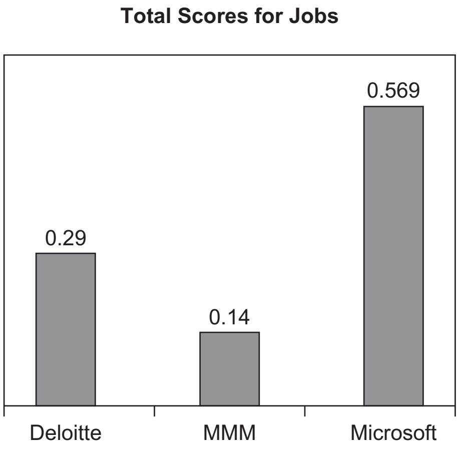
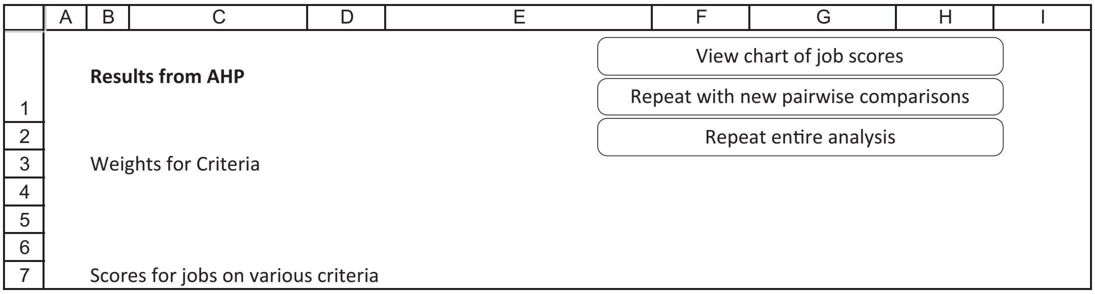


Figure 34.10 Report Worksheet Template



	A	B	C	D	E	F	G	H	I
1		**Results from AHP**							
2									
3		Weights for Criteria							
4									
5									
6									
7		Scores for jobs on various criteria							

34.4 Setting Up the Excel Sheets

The **AHP.xlsm** file contains Explanation and Report worksheets and a ScoresChart sheet. (Unlike most of the other applications in the book, there is no Model worksheet where most of the calculations take place. All calculations in this application are done directly in memory with VBA—that is, they are *not* performed through spreadsheet formulas.) The Report worksheet, shown earlier in Figure 34.8, must be completed almost entirely at run time. The only template that can be developed at run time appears in Figure 34.10. However, the chart can be developed with the Excel's chart tools at design time, using any set of trial inputs, and then it can be tied to the actual job scores at run time.

34.5 Getting Started with the VBA

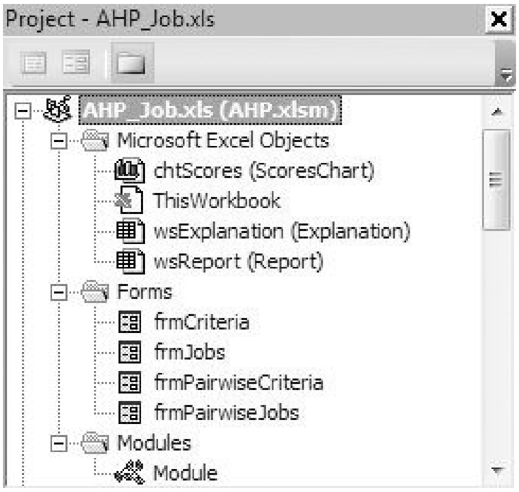
The application contains four user forms named `frmCriteria`, `frmJobs`, `frmPairwiseCriteria`, and `frmPairwiseJobs`, and a single module. Once these are inserted, the Project Explorer window will appear as in Figure 34.11.

Workbook_Open Code

To guarantee that the Explanation worksheet appears when the file is opened, the following code is placed in the `ThisWorkbook` code window. This code hides the Report and ScoresChart sheets.

```
Private Sub Workbook_Open()
    With wsExplanation
        .Activate
        .Range("F5").Select
    End With
    wsReport.Visible = False
    chtScores.Visible = False
End Sub
```

Figure 34.11 Project Explorer Window

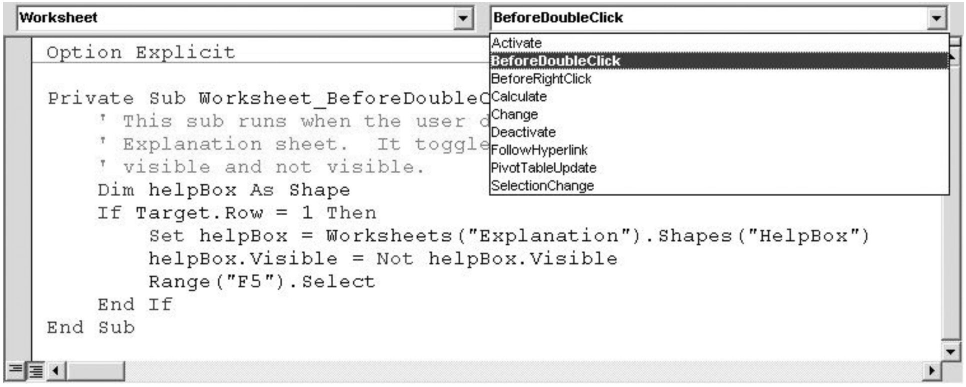


Worksheet_BeforeDoubleClick Event Handler

The Workbook_Open sub has been used repeatedly in previous applications. It responds to the Open event of the Workbook object. When the workbook opens, this code runs. There are many other events that objects can respond to. In each case, it is possible to write an event handler for the event. This can come in very handy. In this application, the user can double-click in row 1 of the Explanation worksheet to display or hide a help text box. This is accomplished by the following event handler for the worksheet's BeforeDoubleClick event. The built-in sub for this event comes with an argument called Target. This argument is the cell that is double-clicked. Therefore, an If statement checks whether Target.Row equals 1. If it does, this means that the user double-clicked somewhere in row 1. In this case, the Visible property of the HelpBox (the name of the text box that contains the help) is toggled from True to False or vice versa. Note that the text box is named HelpBox at *design* time. A text box can be named exactly like a range—you select it and then type a name in the name box area in Excel.

```
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)
    ' This sub runs when the user double-clicks anywhere in row 1 of the
    ' Explanation sheet. It toggles a pre-formed text box between
    ' visible and not visible.
    Dim helpBox As Shape
    If Target.Row = 1 Then
        Set helpBox = Worksheets("Explanation").Shapes("HelpBox")
        helpBox.Visible = Not helpBox.Visible
        Range("F5").Select
    End If
End Sub
```

Figure 34.12 Inserting Sub for BeforeDoubleClick Event



```
Option Explicit

Private Sub Worksheet_BeforeDoubleC
    ' This sub runs when the user d
    ' Explanation sheet.  It toggle
    ' visible and not visible.
    Dim helpBox As Shape
    If Target.Row = 1 Then
        Set helpBox = Worksheets("Explanation").Shapes("HelpBox")
        helpBox.Visible = Not helpBox.Visible
        Range("F5").Select
    End If
End Sub
```

This code should be stored in the code window for the Explanation worksheet. To get to it, double-click the Explanation worksheet item in the Project Explorer window of the VBE. (See Figure 34.11.) Then in the code window, select Worksheet in the left dropdown list and double-click the BeforeDoubleClick item in the right dropdown list. (See Figure 34.12.) This inserts a "stub" for the event handler, as in the following two lines. You can then enter the code you need in the middle. Note that the Target and Cancel arguments are built in—you have no choice whether to include them in the first line. However, only the Target argument is used in the code in this example; the Cancel argument is ignored.

```
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)
End Sub
```

How can you learn about events like this? Probably the best way is to use the Object Browser in the VBE. Figure 34.13 shows where I discovered that a Worksheet object has a BeforeDoubleClick event. The online help then describes the details, such as what the Target and Cancel arguments mean.

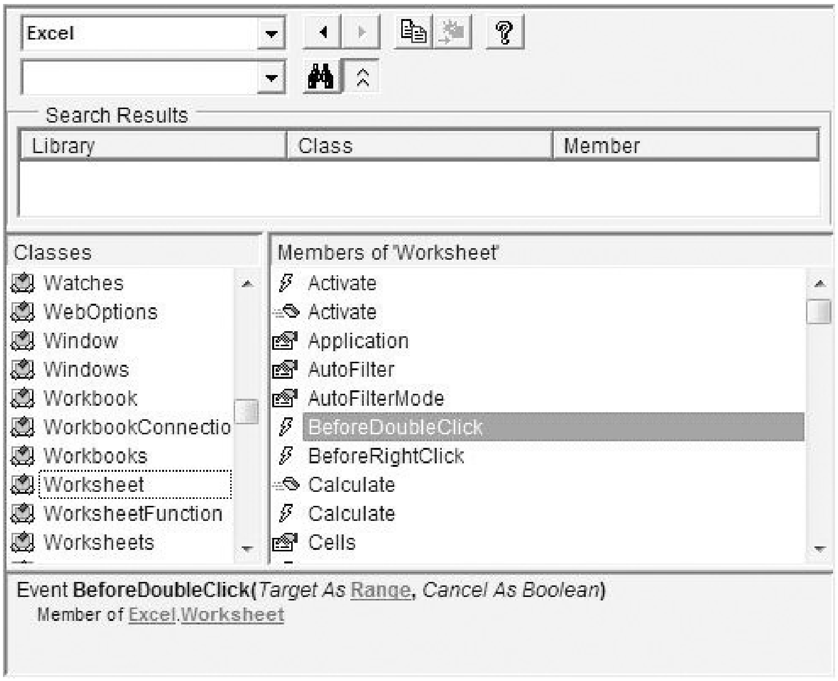
34.6 The User Forms

The user forms include some features not seen in previous chapters: frmCriteria has a combo box control, frmPairwiseCriteria and frmPairwiseJobs each have a scrollbar control, and the buttons on frmCriteria and frmJobs are not the standard OK/Cancel pair. However, this just illustrates the flexibility of the controls available in the Control Toolbox. You can choose the ones that are most appropriate for your application.

frmCriteria

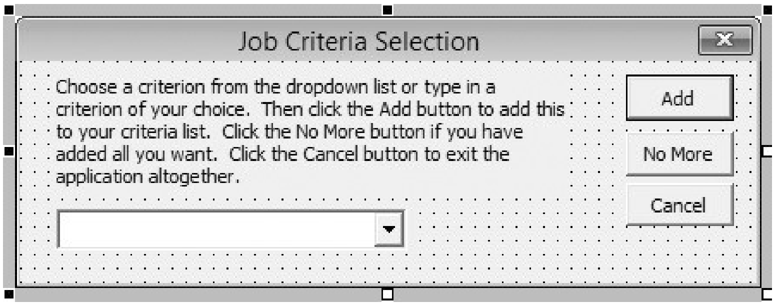
This form has three buttons named btnAdd, btnNoMore, and btnCancel, an explanation label, and a combo box named cboCriteria. Its design appears in Figure 34.14.

Figure 34.13 Object Browser Help for BeforeDoubleClick Worksheet Event



Event **BeforeDoubleClick(***Target As* Range, *Cancel As Boolean***)**
Member of Excel.Worksheet

Figure 34.14 frmCriteria Design



Choose a criterion from the dropdown list or type in a criterion of your choice. Then click the Add button to add this to your criteria list. Click the No More button if you have added all you want. Click the Cancel button to exit the application altogether.

The UserForm_Initialize sub creates an array of criteria that is used to populate the combo box.[1] The btnAdd_Click sub does some error checking and then adds the newly chosen criterion to the list of criteria in the publicly declared criterion array. The btnNoMore_Click sub simply unloads the form. By this time, the user

---

[1] Unlike the applications in the previous chapters, the code involving forms is the "old" version used in the previous edition of the book. This is partly for variety and partly because too much work would have been required to change everything.

has entered all desired criteria, so she just wants the dialog box to disappear. The btnCancel_Click sub unloads the form and terminates the program.

Note that a ComboBox control is essentially a blend of a list box and a text box. Specifically, its List property can be set equal to an array to populate the list, and its Value property returns the item in the box.

```
Private Sub btnAdd_Click()
    Dim newItem As String, isNew As Boolean

    Dim i As Integer
    ' Check that a criterion has been entered and that it is not a criterion
    ' that was already entered. (If it is, set isNew to False.)
    With cboCriteria
        If .Value = "" Then
            MsgBox "Please make a selection", vbExclamation, "No selection"
            .SetFocus
            Exit Sub
        Else
            newItem = .Value
            isNew = True
            If nCriteria > 0 Then

                ' This loop goes through criteria already entered to check
                ' whether the current criterion is new.
                For i = 1 To nCriteria
                    If newItem = criterion(i) Then
                        MsgBox "You already chose this item.", _
                            vbExclamation, "Duplicate"
                        isNew = False
                        Exit For
                    End If
                Next
            End If

            ' Update the number of criteria only if isNew is True.
            If isNew Then
                nCriteria = nCriteria + 1
                If nCriteria = 1 Then
                    ReDim criterion(nCriteria)
                Else
                    ReDim Preserve criterion(nCriteria)
                End If
                criterion(nCriteria) = newItem
            End If

            ' Get ready for the next criterion.
            .Text = ""
            .SetFocus
        End If
    End With
End Sub

Private Sub btnCancel_Click()
    Unload Me
    End
End Sub
```

```
Private Sub btnNoMore_Click()
    Unload Me
End Sub

Private Sub UserForm_Initialize()
    ' Define an array of items that make up the "default" list in the combo box.
    ' The user can add a different item at run time if desired.
    Dim criteriaArray As Variant
    criteriaArray = Array("Salary", "Benefits", "Location", "Quality of life", _
        "Nearness to family", "Interest of work")

    ' Fill the combo box, but don't select any items by default.
    With cboCriteria
        .List = criteriaArray
        .Value = ""
    End With
End Sub
```

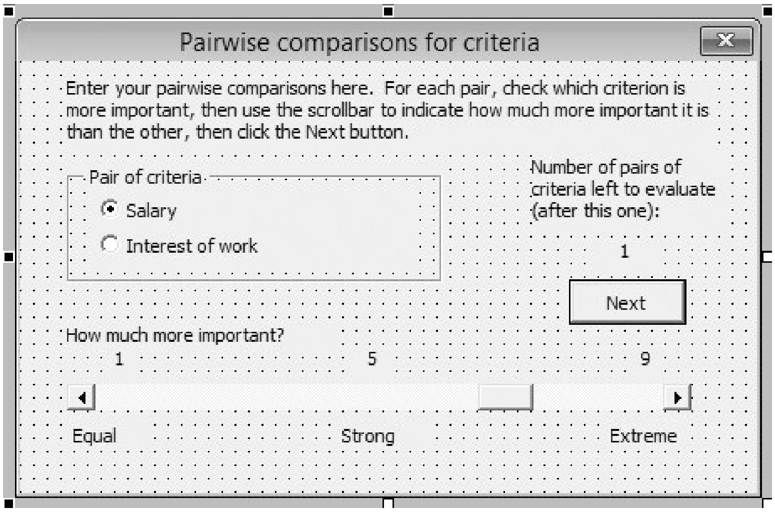
frmJobs

The frmJobs form, shown earlier in Figure 34.4, is analogous to frmCriteria, so its design and event handlers are not repeated here. The only difference is that it contains a text box for capturing the job name, not a combo box.

frmPairwiseCriteria

The design for the frmPairwiseCriteria form appears in Figure 34.15. It has 10 labels, a command button named btnNext, a frame that contains two option buttons named optChoice1 and optChoice2, and a scrollbar named scbCompareValue. Note that the numbers and descriptions above and below the scrollbar are all *labels,* as is the highlighted number in the figure. This latter label is named lblNLeft. A ScrollBar control has several properties, Min, Max, LargeChange, and SmallChange, that can

Figure 34.15 Design of Pairwise Criteria Form



Pairwise comparisons for criteria

Enter your pairwise comparisons here. For each pair, check which criterion is more important, then use the scrollbar to indicate how much more important it is than the other, then click the Next button.

Pair of criteria
● Salary
○ Interest of work

Number of pairs of criteria left to evaluate (after this one):
1

be set at design time. For this application, the SmallChange and LargeChange properties can be left at their default values of 1, but the Min and Max properties should be changed to 1 and 9. (You can probably guess what these properties are all about. See online help on the ScrollBar control for more details.)

The UserForm_Initialize sub uses three public variables, criterion1, criterion2, and nPairsLeft, which have been declared publicly in the module, to initialize the user form. The first two of these are used as captions for the option buttons, and the third (an integer) is used as the caption for the lblNLeft label. By default, the scrollbar is put at its leftmost position, and the first option button is checked. The btnNext_Click sub then captures the option that has been checked and the value of the scrollbar in the public variables choseFirst and pairwiseValue. Note that the Value property of a ScrollBar control is its default property. It is an integer between the scrollbar's Min and Max, determined by the position of the "slider" on the scrollbar.

```
Private Sub btnNext_Click()
    ' Capture which of the two options is favored (choseFirst) and by how
    ' much (pairwiseValue).
    choseFirst = optChoice1.Value
    pairwiseValue = scbCompareValue.Value
    Unload Me
End Sub

Private Sub UserForm_Initialize()
    ' Set up the dialog box. It uses the public variables criterion1,
    ' criterion2, and nPairsLeft, defined in the module.
    With optChoice1
        .Value = True
        .Caption = criterion1
    End With
    optChoice2.Caption = criterion2
    scbCompareValue.Value = 1
    lblNLeft.Caption = nPairsLeft
End Sub
```

frmPairwiseJobs

The frmPairwiseJobs form, shown earlier in ~~Figure~~ Figures 34.6 and 34.7, is very similar to frmPairwiseCriteria, so its design and event handlers are not repeated here.

34.7 The Module

The bulk of the work is performed in the module. When the user clicks the button in the Explanation worksheet, the Main sub runs. It "shows" frmCriteria and frmjobs to get the lists of criteria and jobs, redimensions a number of arrays, and then calls the DoCalculations sub to perform the AHP. For a change, I will not list all of the VBA code here. Unless you understand the calculations that go into the AHP methodology, this code won't make much sense. Besides, from a VBA viewpoint, there is nothing new. If you do happen to be familiar with the AHP methodology and want to see how it is implemented with VBA, the code is available in the **AHP.xlsm** file.

34.8 Summary

This chapter has presented an application that should be useful for many readers of this book—students who are looking for a job. It is easy to use, and it is realistic. The VBA details are somewhat complex, and they will be mysterious to readers who are not familiar with the inner workings of AHP. However, this is part of the beauty of VBA applications. They can be used by people who are not familiar with what is happening "under the hood."

EXERCISES

1. Open a new workbook and draw an oval on it, positioned and captioned approximately as in Figure 34.16. (This was on the Drawing toolbar in Excel 2003 and earlier versions; it is on the Shapes dropdown on the Insert ribbon in Excel 2007 and later versions.) Then insert a text box, positioned approximately as in the figure, and type some text into it. (Make up anything.) Now write code so that when the user clicks the oval, the text box appears (if it was invisible) or disappears (if it was visible). (*Hint:* Once you create the oval, right-click it. You will see that you can attach a macro to it. This macro, placed in a *module*, is where you will store your code. The resulting macro isn't really an event handler, because you can name it anything. However, it does illustrate how you can attach a macro to a shape object in Excel.)
2. The scrollbars used for the pairwise comparison are just one possibility. Another possibility is to use a spinner and an accompanying text box, as illustrated in Figure 34.17. Change the application so that it uses this approach instead of scrollbars. Make sure that the resulting `frmPairwiseCriteria` and `frmPairwiseJobs` are laid out nicely and are meaningful for the user. (Look up help for spinner controls in the MSForms library in the Object Browser.)

Figure 34.16 Getting Online Help

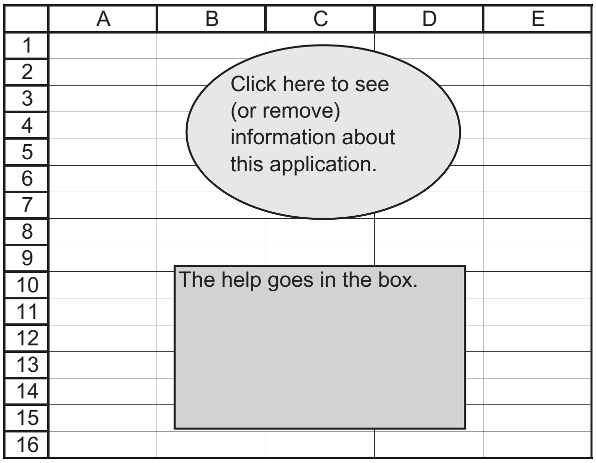
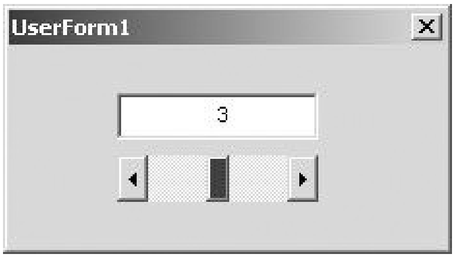

A
B
C
D
E
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Click here to see (or remove) information about this application.
The help goes in the box.

Figure 34.17 Spinner and Text Box Combination



3. Most people who use the AHP method suggest a 1 to 9 scale for making the pairwise comparisons, and this was implemented here. Change the application so that the scale is from 1 to 5. Now the index 5 means what the old index 9 meant. There are simply fewer choices for the user. From a user's standpoint, which of these two scales would you rather use?
4. Change the application so that it pertains to deciding where to go on vacation. Change the automatic entries in frmCriteria's combo box to ones that might be used in this type of decision. Also, replace the text box in frmJobs by a combo box. Place two automatic entries in this combo box: Wife's parents and Husband's parents. (You can assume that these are *always* possible vacation spots, even if they aren't necessarily the preferred ones.)
5. If you open the **AHP.xlsm** file and look at the code in the module, you will see that the CreateReport sub is too long for the taste of many programmers. Rewrite it so that it calls several smaller subs that perform the individual tasks. You can choose the number of smaller subs, but they should make logical sense.