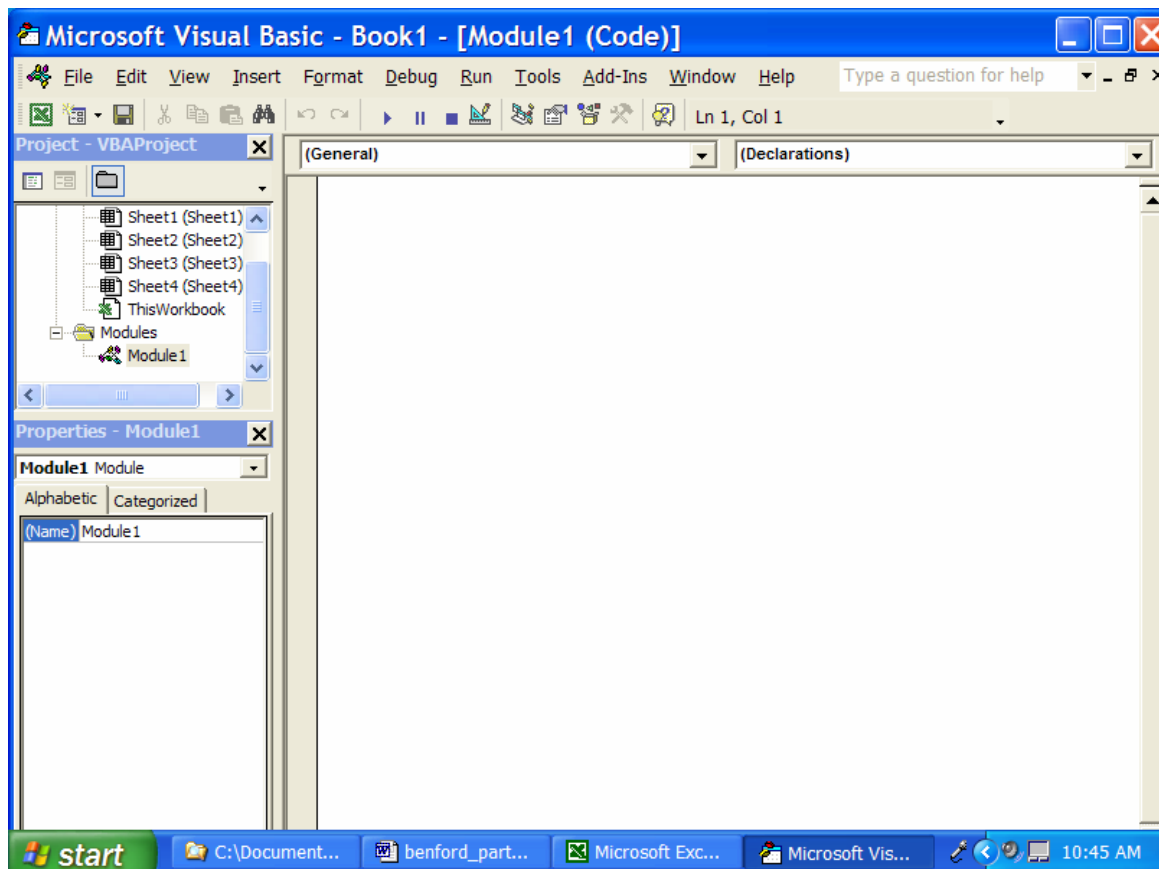


How to Create the Fraud Buster Application

Start by opening a new Excel workbook. First, you will need to create a module where your code will be stored. Placing the code in a module instead of a worksheet will allow you to run the code from multiple sheets. To create a module, click on **Tools, Macro and Visual Basic Editor**. Next, click on **Insert** and then **Module**. You should see a blank frame on the right side of the screen, which is where you will write your code (see exhibit 1, below).

Exhibit 1: Book 1



The Code

Instructions for creating the code follow. The code is presented in blue text.

1. Name the subroutine with the *SUB* command. We called ours Benford.

```
Sub Benford()
```

2. Create a series of arrays for each of the 9 digits. The arrays will keep track of the digit places and counts of digit occurrences.

```

Dim Arrayone(0 To 9) As Integer
Dim Arraytwo(0 To 9) As Integer
Dim Arraythree(0 To 9) As Integer
Dim Arrayfour(0 To 9) As Integer
Dim Arrayfive(0 To 9) As Integer
Dim Arraysix(0 To 9) As Integer
Dim Arrayseven(0 To 9) As Integer
Dim Arrayeight(0 To 9) As Integer
Dim Arraynine(0 To 9) As Integer
Dim Arrayzero(0 To 9) As Integer
Dim Arraytwotest(10 To 99) As Integer

```

3. Create variables for keeping track of the number of columns (Col), number of rows (Row), number of rows in each column (Colcells), number of digits in each number (Digits), current column (Step), first two digits in a number (X) and a counter variable called I.

```

Dim Row As Long, Col As Long, Step As Long, Colcells
Dim Digits As Long
Dim X, I

```

4. Count the total number of columns that have data.

```
Col = Application.CountA(ActiveSheet.Range("1:1"))
```

5. Begin at the first column, go to the first cell in the first column, then move down to the last row that contains data. Use the location of the last row to count the number of rows in the column. The *FOR* Loop will repeat these procedures for every column that contains data.

```

For Step = 1 To Col
    Cells(1, Step).Select
    Selection.End(xlDown).Select
    Row = ActiveCell.Row

```

6. Create another *FOR* Loop to analyze each row within the current column. If you have titles at the top of your data, you can change the “*For Colcells = 1 to Row*” to ignore the rows with titles. For example, if the first row

contains descriptions of each column, simply change this statement to “*For Colcells = 2 to Row.*”

Now use the *LEFT* function to truncate the first two digits from each number in the column. If there are any numbers with only one digit, the *IF* statement will prevent the program from returning an error. The counts for each possible two-digit pair (for example, 10-99) are stored in the array called *Arraytwotest*.

For Colcells = 1 To Row

```
x = Left(Cells(Colcells, Step), 2)
If x > 9 Then
Arraytwotest(x) = Arraytwotest(x) + 1
End If
```

7. Now select each individual digit from a cell. The *FOR* Loop will run the next segment of code for every digit in each cell.

For Digits = 1 To Len(Cells(Colcells, Step))

8. The *MID* function determines the value of each digit in every number. The *SELECT CASE* statement allows you to run alternative segments of code for each potential value of the case. Because the *MID* function returns each digit in a number, there are 10 potential cases, one for each digit 0 through 9. For each digit value, you will maintain a total count in each possible location (for example, the first digit in a number, the second, and so on). The counts are stored in the array variables that you defined for each digit. These counts later will be used to perform the first- and second-digit tests, following Benford’s Law.

Select Case Mid(Cells(Colcells, Step), Digits, 1)

Case 1

Arrayone(Digits) = Arrayone(Digits) + 1

Case 2

Arraytwo(Digits) = Arraytwo(Digits) + 1

Case 3

Arraythree(Digits) = Arraythree(Digits) + 1

Case 4

Arrayfour(Digits) = Arrayfour(Digits) + 1

Case 5

Arrayfive(Digits) = Arrayfive(Digits) + 1

Case 6

Arraysix(Digits) = Arraysix(Digits) + 1

Case 7

Arrayseven(Digits) = Arrayseven(Digits) + 1

Case 8

Arrayeight(Digits) = Arrayeight(Digits) + 1

Case 9

Arraynine(Digits) = Arraynine(Digits) + 1

Case 0

Arrayzero(Digits) = Arrayzero(Digits) + 1

End Select

9. End the *FOR* statement for digits with a *NEXT* statement. This moves you to the next digit in a cell.

Next Digits

10. End the *FOR* statement for rows within the column with a *NEXT* statement. This moves you to the next row in a column after you have analyzed all of the digits in the current cell.

Next Colcells

11. End the *FOR* statement for the column with a *NEXT* statement. This moves you to the next column after you have analyzed all of the rows in the current column.

Next Step

12. Place the counts for the first-digit test into a spreadsheet. We have decided to put the counts on Worksheet 2, beginning in cell C5. For each array variable, place a 1 in parentheses. This indicates you want the count for the first-digit places. There is an array for each of the possible nine digits that can appear as the first digit in a number (note that zero cannot be the first digit). These digit counts can now be compared with the predicted counts from Benford's formula. This will be performed on the spreadsheet.

```
Worksheets(2).Range("C5").Value = Arrayone(1)
Worksheets(2).Range("C6").Value = Arraytwo(1)
Worksheets(2).Range("C7").Value = Arraythree(1)
Worksheets(2).Range("C8").Value = Arrayfour(1)
Worksheets(2).Range("C9").Value = Arrayfive(1)
Worksheets(2).Range("C10").Value = Arraysix(1)
Worksheets(2).Range("C11").Value = Arrayseven(1)
Worksheets(2).Range("C12").Value = Arrayeight(1)
Worksheets(2).Range("C13").Value = Arraynine(1)
```

13. The next segment of code places the counts from the second-digit test on worksheet 3.

```
Worksheets(3).Range("C5").Value = Arrayzero(2)
Worksheets(3).Range("C6").Value = Arrayone(2)
Worksheets(3).Range("C7").Value = Arraytwo(2)
Worksheets(3).Range("C8").Value = Arraythree(2)
Worksheets(3).Range("C9").Value = Arrayfour(2)
Worksheets(3).Range("C10").Value = Arrayfive(2)
Worksheets(3).Range("C11").Value = Arraysix(2)
Worksheets(3).Range("C12").Value = Arrayseven(2)
Worksheets(3).Range("C13").Value = Arrayeight(2)
Worksheets(3).Range("C14").Value = Arraynine(2)
```

14. Place the counts for the first-two-digits test on the fourth worksheet. You can do this with less code than was necessary for the first- and second-digit tests, because there is only one array variable necessary for the second-digit

test. First, select the fourth worksheet, and then select the cell where you wish to begin inserting the counts for the two-digit combinations (10-99). Create a *FOR* Loop that will repeat the code for each of the digit pairs 10 through 99. The *ActiveCell.Offset* moves the cursor down one cell, so that each count will be placed in the next cell. *ActiveCell.Value* sets the value of the currently selected cell equal to the count of the current digit pair.

```
Worksheets(4).Select  
Range("c3").Select
```

```
For I = 10 To 99  
ActiveCell.Offset(1, 0).Select  
ActiveCell.Value = Arraytwotest(I)  
Next I
```

15. Go to the second sheet, where the results from the first-digit test are stored. When the program finishes making the calculations, you will be taken directly to Worksheet 2, where the results of the first-digit test are stored.

```
Worksheets(2).Select
```

16. End the subroutine.

```
End Sub
```

That completes the code necessary to conduct the first-digit, second-digit, and first-two-digits tests. Once you have captured the digits and their places, you also can perform many other tests using slight variations in the code. For example, you can quickly alter the code to test the third digit, last digit and last two digits, for example.

The Worksheets

Once you have created the code, you will need to format four worksheets to use the calculations from your code. Exhibits 2 through 4 display the formulas necessary to prepare the worksheets that use the results from the VBA code.

Exhibit 2: First-Digit Test Sheet

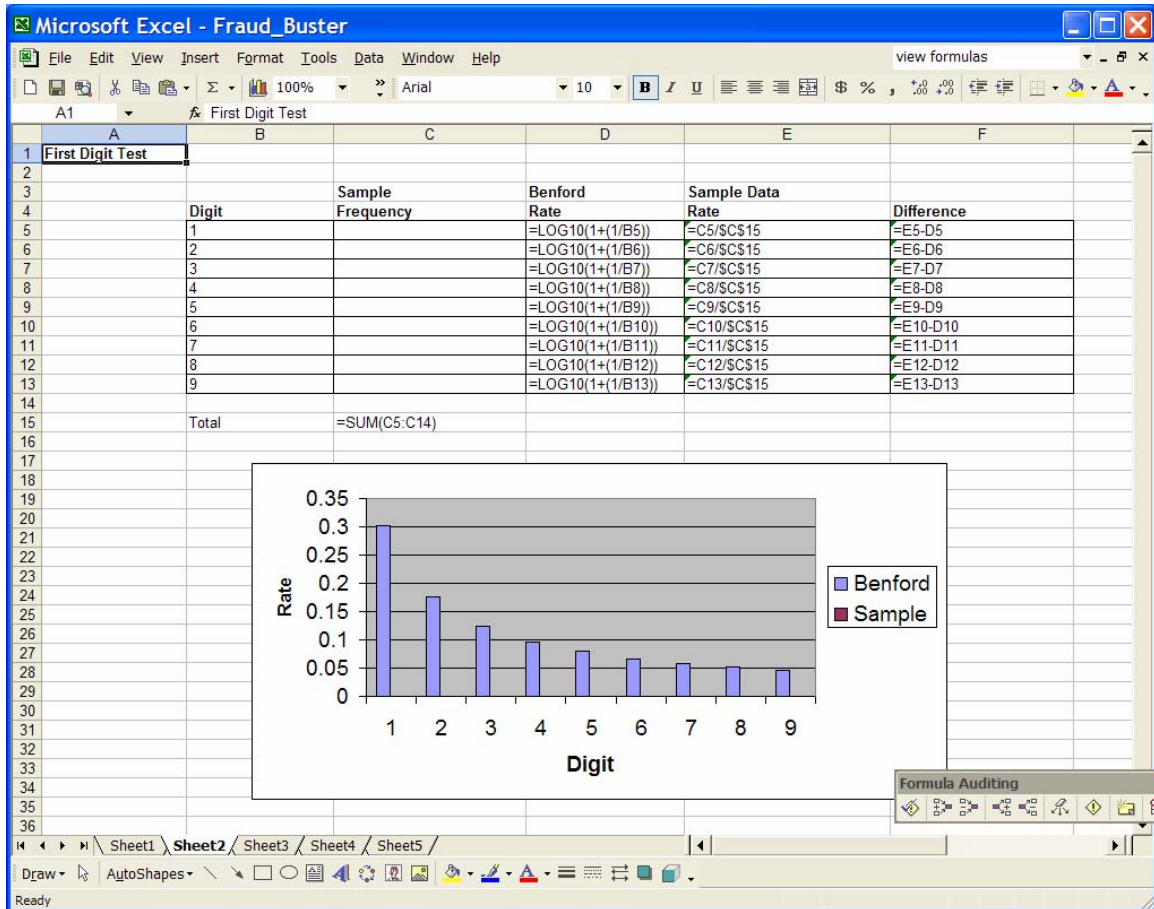


Exhibit 3: Second-Digit Test Sheet

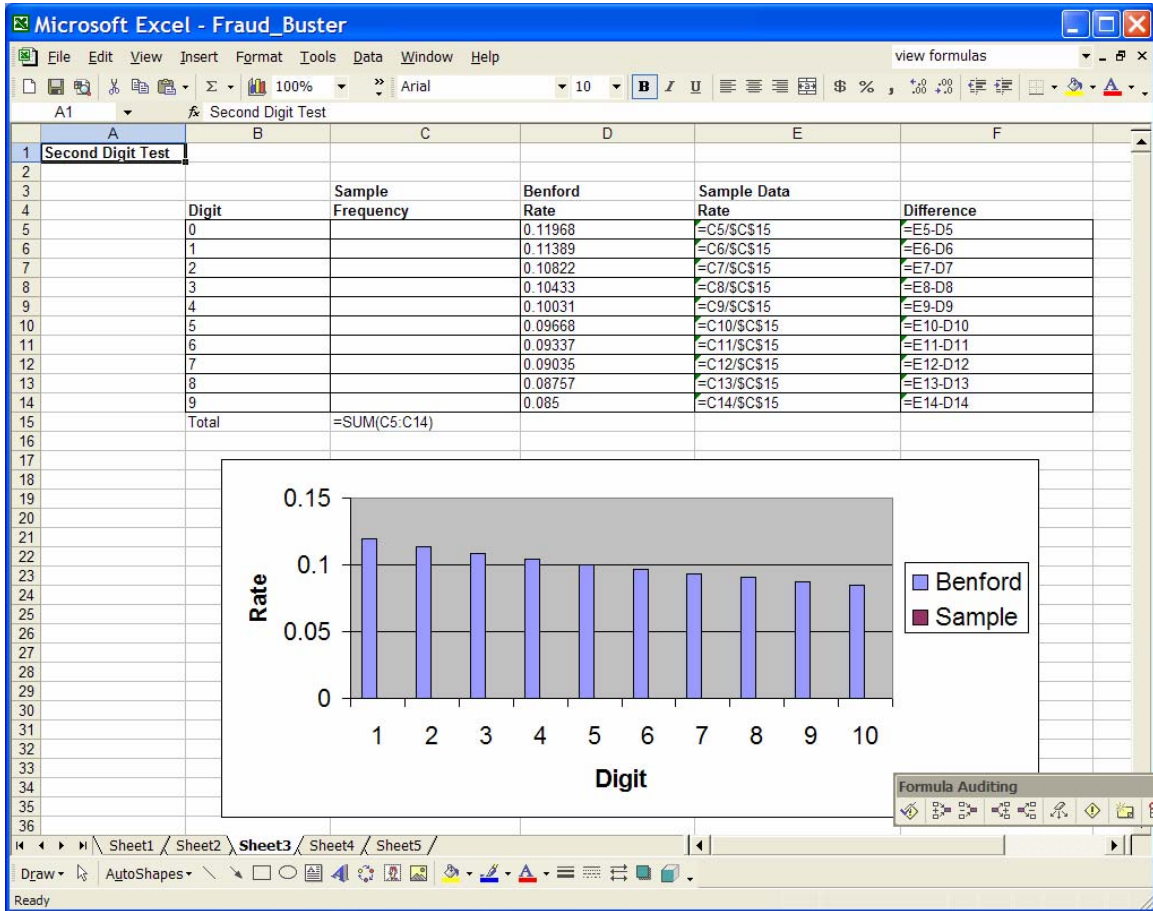
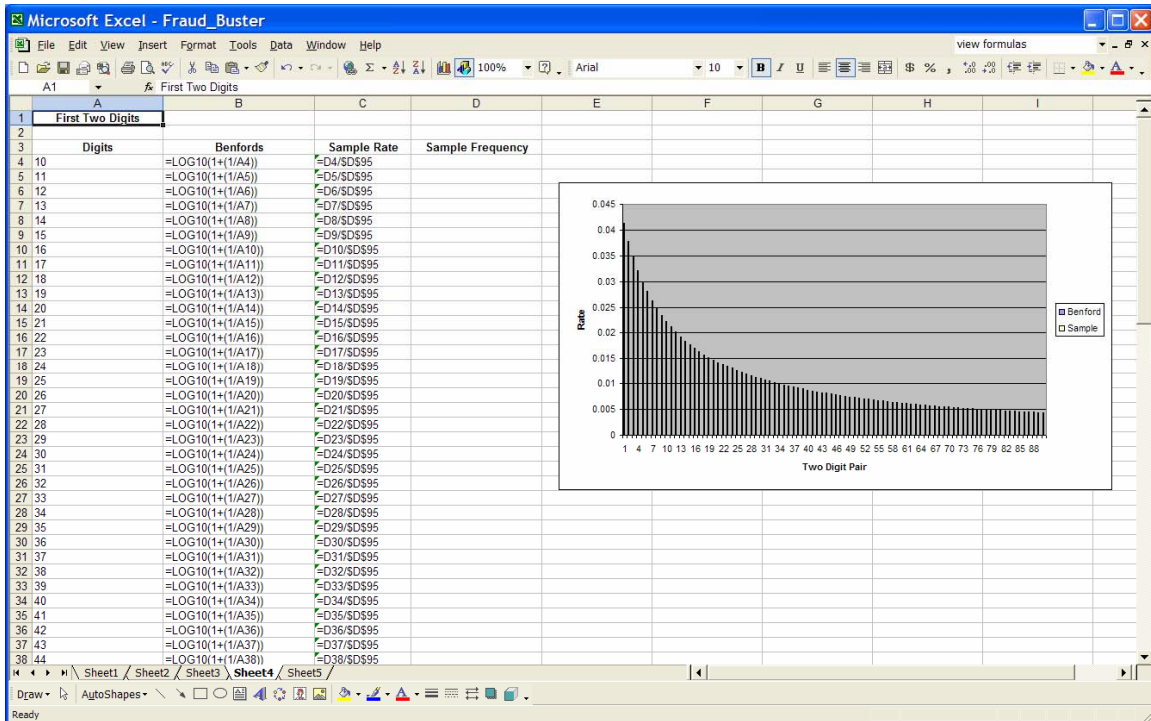


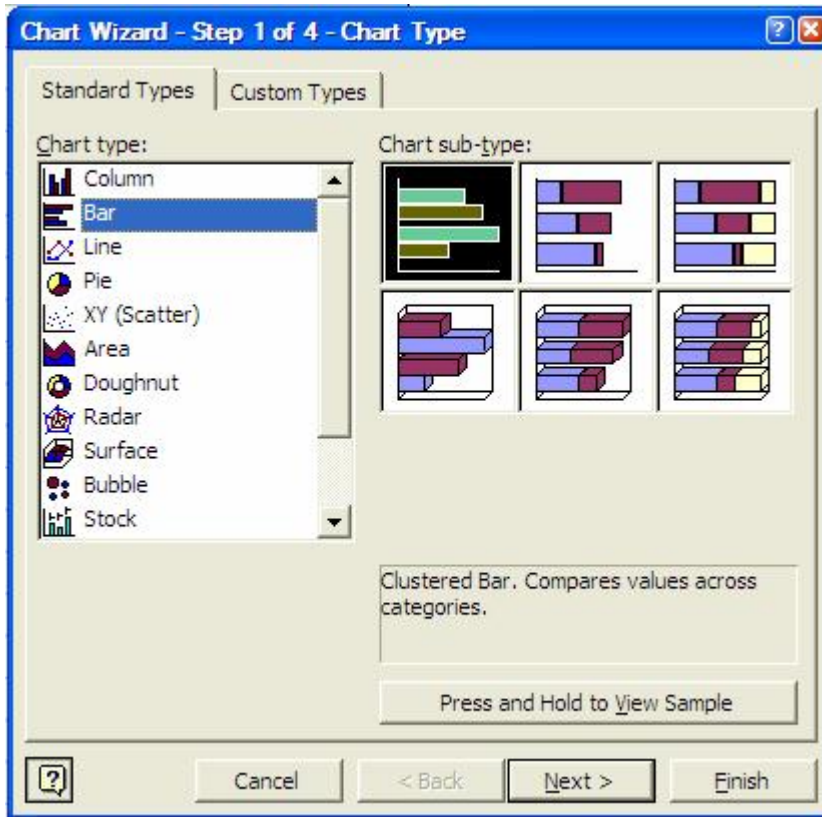
Exhibit 4: First-Two-Digits Test Sheet



The Bar Charts

To create the charts, go to the toolbar, click on **Insert, Chart** and choose the bar chart (see exhibit 5). The chart wizard will ask for the chart's data range. Select the range that includes the Benford predicted rates and the sample rates. Format the chart as you desire.

Exhibit 5: Chart Wizard



The Button to Run the Code

To place a button on a sheet, click on **Insert** from the toolbar, then on **Picture, AutoShape**. Now choose the auto shape you like by clicking on the shape with the left mouse button. Draw the shape on the worksheet by holding down the left mouse button. Right-click on the shape and choose **Format AutoShape** to set its properties. For example, you can change the color. Right-click again and choose **Assign Macro**. This allows you to assign your code to the button. We chose the **Benford** macro that we created for this program.