Audit Sampling Project Write-up

A VBA project completed by David Burgess

Executive Summary

Sampling can be a difficult and tedious task for auditors performing an independent audit of financial statements. Oftentimes the process of choosing a random sample and checking the values can become biased as auditors have to make the selections themselves. Using an excel spreadsheet and visual basic for applications, I have automated the process of choosing a random sample, comparing the amounts with audited values, and ultimately recommending an opinion on the account being audited.

As an auditor for the Utah state auditors and for Deloitte, I could've used a program that automated this process in a completely unbiased way. While the solution I have provided is a simpler example, the same process could be used for more complicated sampling procedures. In this write-up, I will attempt to explain the basics of a sampling procedure, how my solution will be implemented to make sampling easier, and what I learned by going through this project. I hope to be able to use what I have learned to help me when I start with Deloitte full-time in September.

Using Sampling to Perform an Audit

To understand how my solution works, a basic understanding of how sampling is used in the auditing process is necessary. First, an auditor will plan the audit and set certain parameters for each account. The auditor sets a limit on how much the account can be off, called tolerable misstatement, and estimates how much misstatement and standard deviation they will find as well. A confidence level is chosen, and then the sample size is calculated based on these parameters and the total population size. Second, the auditor generates a random sample and compares the sample items to an audited value that is verified by the auditor. Finally, the auditor can use the amount of misstatement in the accounts and projects it to the rest of the population – producing a confidence interval that must be within tolerable misstatement for the account to be accepted by the auditor.

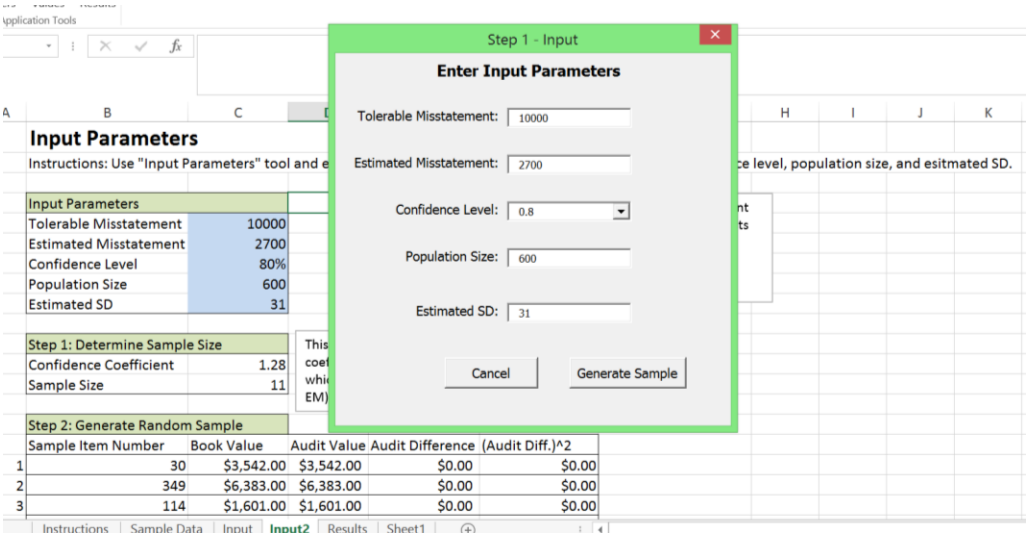
Solution Walkthrough

The solution I have provided starts with the assumption that the auditor has access to some financial data, and the "Sample Data" tab has 600 account balances (randomly generated) to simulate this process. This tab contains a dynamically named region with data that can be changed as the auditor changes to a different account. The three main steps in the solution include the following:

1. Input Parameters
2. Enter Audit Values
3. Export Results

Step 1 – Input Parameters

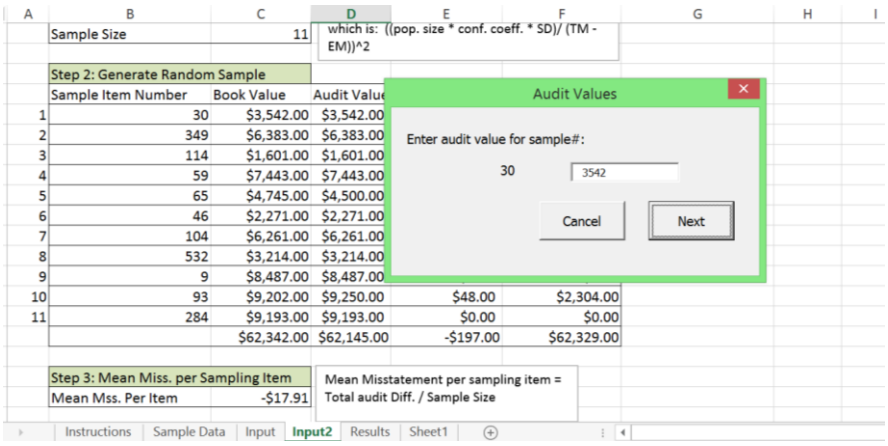
In the custom ribbon named "Application Tools," the auditor will start by clicking on the "Enter Parameters" button which opens the form frmInput. The following screenshot, exhibit 1, shows the form.



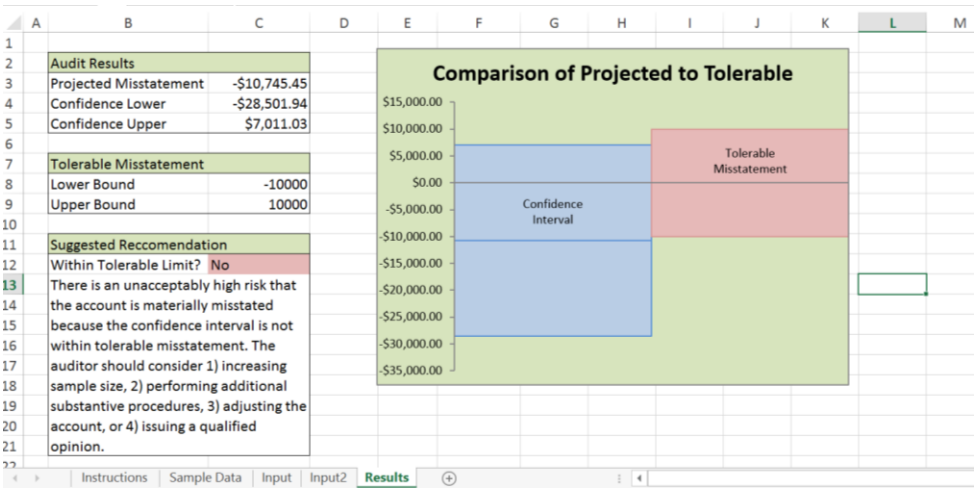
The user enters the parameters of the account in each text box and chooses a confidence level from the dropdown box. If the auditor put the information in correctly, clicking on “Generate Sample” will begin the next step in the process.

Step 2 – Enter Audit Values

The Generate Sample button should have inserted the correct number of rows, generated a random sample, and then pulled the account balances from the Sample Data tab based on the sample item numbers. The next step for the auditor is to compare the audited values with that of the book values for each sample item. Again in the Application Tools custom ribbon, the auditor will select “Enter Audit Values” to pull up the following form, called frmValues in the code.

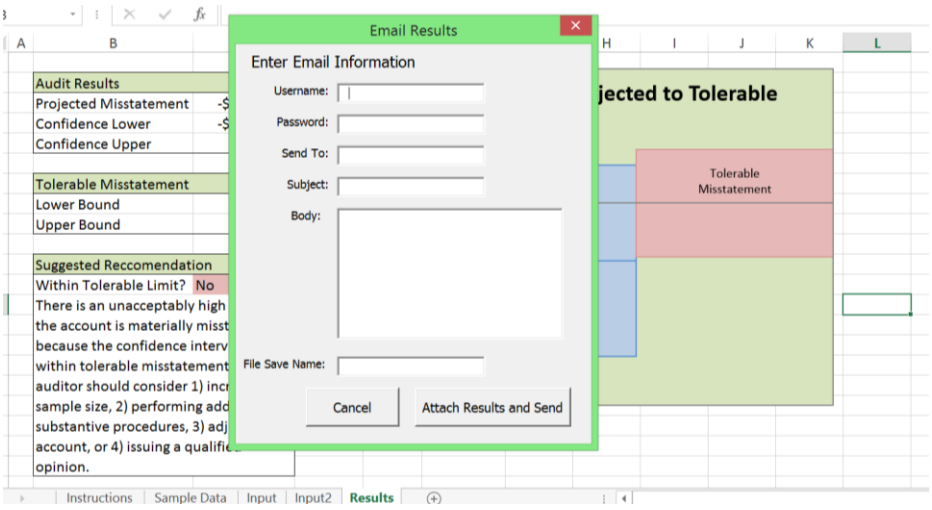


For each sample item, the auditor will enter the verified amount, which will sometimes be slightly different than the book value, and then click next to proceed to the next sample item. The program automatically finds the difference in values and squares the value in the next column. On the final audit value entered, the program will finish the math required to determine the results of the audit. The following screenshot shows the Results tab, which summarizes the output as projected misstatement, confidence interval, tolerable interval, a graphic comparison of the confidence interval compared to tolerable, and suggested recommendation (which uses a text manipulation formula to return an official auditors opinion on the account.



Step 3 – Export Results

The final button in the Application Tools custom ribbon is the Export Results button. This button opens a form, called frmEmail, that takes the necessary information for exporting the results to a pdf and sending that pdf to a specified email. The following screenshot shows the form.



The form takes a username, password (with hidden symbols), send to email, subject, body, and a name to save the file under. Clicking “Attach Results and Send” will use the information in the form to export the file to a pdf and then send that file with a subject, message, and attachment to the specified email address. With the results pdf as documentation, the auditor can then proceed with the audit. For more on how the code works to perform the solution, see either the downloaded spreadsheet or the Appendix section on “Important Code Segments.”

Learning and Conceptual Difficulties

Learning

In creating this solution I learned a great deal in how to implement a solution for an auditing process through VBA. First of all, I was actually surprised by how easy it was to create a form that quickly and professionally automates the process of entering values into the program. Using forms and procedures

that run the behind the forms, I feel like I can automate any process and really impress my superiors. I also noticed how easy it was to create a customized ribbon that runs those forms.

Difficulties

I encountered difficulty when I was thinking how I could import the data to the spreadsheet to begin the sampling process. The problem is that the financial data could come in a wide variety of ways. Some data will be in a pdf, some on an online database, some data may just be printed out, and some data might just be in another spreadsheet. Ultimately, I decided this part of the process was outside the scope of what I wanted this project to do and made the assumption that the auditor can get the data and put it in the Sample Data tab. I checked with Professor Allen and he verified the scope of my project before it was finished.

Most other difficulties encountered during the course of this project involved me just making a silly mistake that was hard to find in the debugging process. Once I found the error, I felt really stupid for making that kind of an error in the first place. However, as I got closer to finishing the project I realized that I was making far fewer errors and could find and correct any errors faster than before.

Assistance

To complete the project and get past certain roadblocks, I did ask for a little help from the TA. Specifically, he helped me get my emailing function working correctly and helped me finish customizing the ribbon. I also used a source from the web to learn how to export a file as a PDF. The sendGMail code is almost directly from the in-class example, but any assistance only amounted to having a few trivial questions answered, and all of the rest of the code in the program is written by me.

Appendix: Important Code Segments

(General) GenerateSampleRows

```
Sub GenerateSampleRows()
Dim s As Worksheet
Dim data As Worksheet
Dim sample As Integer
Dim size As Integer
Dim y As Integer
Dim sum As Long


Set s = Sheets("Input2")
Set data = Sheets("Sample Data")
sample = s.Cells(13, 3).Value
size = s.Cells(8, 3).Value

ClearRows

AddRows

y = 17
Do Until s.Cells(y, 1).Value = ""
s.Cells(y, 2).Value = Randos(1, size)
s.Cells(y, 3).Value = Application.VLookup(s.Cells(y, 2).Value, data.Range("SampleData"), 2)
s.Cells(y, 5).Value = s.Cells(y, 4).Value - s.Cells(y, 3).Value
s.Cells(y, 6).Value = (s.Cells(y, 5).Value) ^ 2
sum = sum + s.Cells(y, 3).Value

y = y + 1

Loop

s.Cells(y, 3).Value = sum

's.Cells(y, 3).Value = Application.sum(Range(Cells(17, 3), Cells(y - 1, 3)))
```

This code generates the required number of rows in for the sample.

(General) Addl

```
Sub ClearRows()
Dim s As Worksheet
Set s = Sheets("Input2")

d = 17
Do Until s.Cells(d, 1).Value = ""
d = d + 1
Loop

Range(s.Cells(17, 1), s.Cells(d - 1, 1)).EntireRow.Delete

End Sub

Sub AddRows()
Dim s As Worksheet
Set s = Sheets("Input2")
Dim b As Integer
Dim samp As Integer
Dim x As Integer

samp = s.Cells(13, 3).Value

b = 17
x = 1
Do Until b = 17 + samp
s.Cells(b, 1).EntireRow.Insert
s.Cells(b, 1).Value = x
x = x + 1
b = b + 1
Loop

End Sub
```

```
End Sub

Private Sub UserForm_Initialize()

 FillConfidence
 'FillInterest

 'ListBoxStuff

 showData
End Sub

Private Sub showData()
   txtTolerable = Sheets("Input2").Cells(5, 3).Value
   txtEstimated = Sheets("Input2").Cells(6, 3).Value
   cboConfidence = Sheets("Input2").Cells(7, 3).Value
   txtPopulation = Sheets("Input2").Cells(8, 3).Value
   txtSD = Sheets("Input2").Cells(9, 3).Value

End Sub

Sub FillConfidence()
  Dim r As Long
  Dim p As Worksheet
  Set p = Sheets("Input2")
  r = 6

  Do Until cboConfidence.ListCount = 0
    cboConfidence.RemoveItem 0
    Loop

  Do Until p.Cells(r, 5).Value = ""
    cboConfidence.AddItem p.Cells(r, 5).Value
    r = r + 1
    Loop
```

```
Private Sub GenerateSample_Click()

If txtTolerable.Text = "" Then
    MsgBox "Must tolerable misstatement.", vbCritical
    txtTolerable.SetFocus
    Exit Sub
End If

If txtEstimated.Text = "" Then
    MsgBox "Must enter estimated misstatement.", vbCritical
    txtEstimated.SetFocus
    Exit Sub
End If

If cboConfidence.Value = "" Then
    MsgBox "Must enter confidence interval.",
    cboConfidence.SetFocus
    Exit Sub
End If

If txtPopulation.Text = "" Then
    MsgBox "Must enter population size.", vbCritical
    txtPopulation.SetFocus
    Exit Sub
End If

If txtSD.Text = "" Then
    MsgBox "Must enter estimated standard deviation.", vbCritical
    txtSD.SetFocus
    Exit Sub
End If

   Sheets("Input2").Cells(5, 3).Value = Format(txtTolerable.Value, "General Number")
   Sheets("Input2").Cells(6, 3).Value = Format(txtEstimated.Value, "General Number")
   Sheets("Input2").Cells(7, 3).Value = Format(cboConfidence.Value, "0.00%")
```

These segments run behind the enter parameters form.

```
Private Sub cmdSend_Click()
    Dim un As String
    Dim pw As String
    Dim sendto As String
    Dim subject As String
    Dim body As String
    Dim attach As String
    Dim SaveName As String

    un = txtUn.Text
    pw = txtPw.Text
    sendto = txtSendto.Text
    subject = txtSubject.Text
    body = txtBody.Text
    SaveName = txtSave.Text

    ChDir ActiveWorkbook.Path & "\"

     Sheets("Results").ExportAsFixedFormat Type:=xlTypePDF, Filename:= _
        SaveName _
        , Quality:=xlQualityStandard, IncludeDocProperties:=True, IgnorePrintAreas
        :=False, OpenAfterPublish:=True

    'Debug.Print un, pw, sendto, subject, body
    attach = ActiveWorkbook.Path & "\" & SaveName & ".pdf"

    If sendGMail(sendto, un, pw, subject, body, attach) Then
        MsgBox "Email sent.", vbOKOnly
        Unload Me
    Else
        MsgBox "Email did not send, check username and password.", vbOKOnly
    End If

End Sub
```

This code will send take the email information and export the file and send it.

```
Option Explicit

Private Sub cmdCancel_Click()
    Unload Me

                                    Name As String)

                                    iveSheet.Name
                                    orkbook.Path & "\"
                                    = ActiveSheet.Name

                                    ExportAsFixedFormat Type:=xlTypePDF, Filename:= _
                                    ame _
        , Quality:=xlQualityStandard, IncludeDocProperties:=True, IgnorePrintA
        :=False, OpenAfterPublish:=True
    MsgBox "File Saved " & " " & fileSaveName
End Sub

Private Sub cmdSend_Click()
    Dim un As String
    Dim pw As String
    Dim sendto As String
    Dim subject As String
    Dim body As String
    Dim attach As String
    Dim SaveName As String

    un = txtUn.Text
    pw = txtPw.Text
    sendto = txtSendto.Text
    subject = txtSubject.Text
    body = txtBody.Text
    SaveName = txtSave.Text

    ChDir ActiveWorkbook.Path & "\"
```

```
    Unload Me
End Sub

Private Sub cmdNext_Click()
Set s = Sheets("Input2")
Dim r As Worksheet

Set r = Sheets("Results")

  s.Cells(row, 4).Value = Format(txtValues, "Currency")
  s.Cells(row, 5).Value = s.Cells(row, 4).Value - s.Cells(row, 3).Value
  s.Cells(row, 6).Value = (s.Cells(row, 5).Value) ^ 2


   If s.Cells(row + 1, 1).Value = "" Then

        s.Cells(row + 1, 4).Value = Application.sum(Range(Cells(17, 4), Cells(row, 4)))
        s.Cells(row + 1, 5).Value = Application.sum(Range(Cells(17, 5), Cells(row, 5)))
        s.Cells(row + 1, 6).Value = Application.sum(Range(Cells(17, 6), Cells(row, 6)))
        r.Cells(3, 3).Value = s.Cells(row + 7, 3).Value
        r.Cells(4, 3).Value = s.Cells(row + 16, 3).Value
        r.Cells(5, 3).Value = s.Cells(row + 17, 3).Value

        MsgBox "Audit Complete, see results tab.", vbOKOnly
      Unload Me
    Else
      row = row + 1
      showSample
    End If
End Sub

Private Sub Label1_Click()

End Sub
```

This program will enter audit values in the Enter Values step.