Corporate Card Holder Receipt Collection Threshold

Ryan Rawlings
MBA 614 Final Project

Executive Summary

My VBA project is the direct result of a consulting project I performed for Adobe as part of Tom Foster's quality management class. While there were other team members assisting with the analysis and the formatting of the project, the VBA code in the project is entirely my own.

The project for Adobe involved their policy for turning in receipts for transactions on company credit cards. In a nutshell, this project was designed to balance the cost and hassle of scanning and processing a receipt with the risk of making errors in sales tax accrual. These errors could take the form of either over or under accruing taxes. Over accruals leads to too much tax being paid by Adobe, while under accruals can lead to penalties and extra interest should a state tax auditor discover it. Adobe does not wish to avoid any taxes, but understanding that errors will inevitably occur, they wanted to know if there was some sort of threshold below which the risk for over-accruals or fines was less than the cost of processing a receipt.

We received 2 years' worth of transaction data from Adobe for their corporate card holders' transactions. Because this is sensitive data for Adobe, I have elected not to submit this data to the blog. My solution required two major operations. The first was to determine how many scans were necessary at any given collection threshold. I was told that employees are required to turn in receipts monthly, so I assumed that they would not be scanning any more frequently than that. The transPerAcct() macro generates a unique list of active card holders from the data, then counts how many transactions each employee had over the threshold. It then parses the date, and categorizes the purchases into months. Finally, it counts the number of months in which an employee had any transaction over the threshold.

The second major operation was to determine the actual tax liability and the estimated risk of mis-accrual. To accomplish this, the code adds a column for the tax charge, multiplying the transaction amount with the state tax rate. Some of the data sheets we were given did not contain any reference to the tax rate—in those cases the code pulls the average US sales tax rate. That total tax liability is the modified by certain values on the "Main" tab—The % of transactions with a trusted vendor, and the % of taxes paid at Point of Sale. Both of these are measures to estimate the risk of over-accrual.

The cost of gathering receipts is combined with several variables on the "Main" tab to generate a total cost of processing at any given collection threshold. The total tax burden is modified by an assumed error rate and the calculated penalty is and added to the calculated over-accrual. The point at which the sum of the tax liability and the receipt processing cost is the lowest is the appropriate receipt collection threshold. Because the receipt holders and the accounting department generally operate with different thresholds, I used 2 data tables to populate 2 graphs showing how the numbers changed with the threshold for any given set of assumptions.

In order to provide some further intuition into the data, the import process also generates a histogram of the transactions by amount. This allows someone analyzing the data another angle into how to manage the balance between high dollar/ low volume transactions and low dollar/high volume transactions.

Implementation Documentation

GetData

There are 3 columns that are necessary to complete the calculation of the threshold limit. Those are the cardholder name, the transaction date and the transaction amount. This procedure also looks for a transaction state column, but because the code has a workaround for a missing transaction state, this column is not required.

Looping through files saved in the \data\ folder, the procedure searches for the required column name (which has been stored in an array, so that I can loop through the column names as well). If the column name is not found the program generates an error telling the user which column name was not found so that they can go and investigate further. Within the loop for each file, there is a separate loop that selects the column name, then uses the xlDown function to select the rest of the column. The data is then copied and pasted into the analysis workbook. At the end of the loop, the program records the row number, then begins pasting data from the next file on the next row.

Outside of this loop, I have the program check for a column containing the state of the transaction. I left this out of the loop because if this field isn't found, it does not need to generate an error. If the field is found, it is imported just like the other three columns.

TransPerAcct

This procedure begins looking for the required columns, and assigning the column numbers of each to different variables. It also inserts a new column next to the transaction dates to hold the data for the parsed month. It then begins checking the data for blank, non-numeric, or less than -$1,000,000 values in the transaction amount column, and deletes the applicable rows. Blank values can throw off the calculations, and non-numeric values can cause errors. Each month of data contains a few summary lines with very large negative numbers. Some negative numbers are allowable, as a reversed transaction does affect sales tax, but these large numbers greatly skew the calculations and need to be removed. Also, this process filters out the column headings that were brought over during the GetData procedure. This loop assumes that there will not be very many of these values, and exits the loop after it has deleted 100 rows.

The procedure then copies the list of cardholders, pastes it over into the "ScanCounts" sheet and removes any duplicates from the list. It then adds a column containing a "COUNTIFS" function counting the transactions that A. belong to that cardholder, and B. were over the collection threshold on the "Main" worksheet. The next column uses a "COUNTIF" function to determine the percentage of that cardholder's transactions that were over the threshold.
The next task was to sort the transactions into months. The procedure labels the columns using the MonthName function in a loop. Because the date formats were not consistent, (ie, January could be 01 or 1), it wasn't possible to use MonthName directly on the transaction date. I used a Case statement to have MonthName correctly identify the month regardless of the date format. The month names are placed into the already-created blank column and are tabulated by "COUNTIFS" statements on the "ScanCounts" sheet.

CalculateFine

This is the program that the ribbon control is pointing to. The first thing this procedure does is check for a field in the "RawData" sheets called "Est Taxes". If it finds one, it returns an error saying that the data have already been imported. After this check, it runs the GetData and TransPerAcct procedures described above.

The procedure next checks to see if the importer successfully brought in the state of the transaction. If it did, the procedure creates two new columns, and uses a "VLOOKUP" to determine the correct tax rate and from there calculate the total tax liability. If the state column is not successfully found, I looped across the top of the columns to find the first empty column, then uses the average US tax rate on the "State Tax Rates" sheet to calculate the tax liability.

Regardless of whether the state tax rates are present or not, the first of these two columns calculates the total tax liability. The second modifies that liability based on parameters entered on the "Main" page. For each column, after the values have been filled to the end of the data, each range is given a name. ("Tax" and "TaxAdj" respectively). Once these ranges are named, the macro runs a histogram on the transaction data, and returns the user to the "Main" sheet.

Histogram

This is a simple procedure that takes the numbers from the data table, then pastes them into "no-man's-land" on the "RawData" tab and names the range "HistBin". This is not an elegant solution, but I got errors when having the data on one sheet and the bin values in another, and this way worked. The Histogram is also very picky about having **only** numeric values in its data. To scrub the data, I created another column, and requested that non-numeric values be given a value of -1. This ensured both that the histogram wouldn't have problems with them, and also that those values wouldn't show up in the histogram data itself (as the bin values started at 0).

Learning and Difficulties

There were several points where I got stuck putting this together. One of the more interesting problms had to do with the step in the "TransPerAcct" sub where I am deleting rows out of the workbook. I had initially tried to make it work with a For Each loop, but received an error on the loop after the first deletion. After trying several ways around, I eventually went back to the familiar For Next loop, but even that caused me difficulties. As I tested that portion of my code, it was deleting every other line, but if there were 2 together needing deletion, it would skip the second. Setting up a watch here was crucial to understand that because I had used my counter variable as my row variable as well, when a row was deleted, the counter needed to be set back one as well.

I have also had problems throughout the semester on any project that required us to manipulate workbooks as objects. I had the same problem again during this project, but I feel as though I finally understand how to make it work. Getting a better understanding of how the Dir function works, what it returns, and why helped greatly in understanding how to make the whole thing work together.
Other difficulties included being able to having the "Find" method return a Boolean value. This ended up being much less straightforward that I had assumed it would be. Eventually I was able to make it work by using the COUNTIF function, and testing whether it was > 0, but it seems like a very inelegant solution, and I expect there is a better one out there. Because the files that we were pulling in were not standard, I also was able to become much more familiar with named ranges, and I can happily report that I have been converted. Having those ranges named made the code easier to write, made my formulas easier to write, and I felt that they made the entire design of the workbook more robust.

I toyed with the idea of putting the parameters into a userform, but decided against it, as these values are meant to be changed on the fly and analyzed, not “submitted” to some central process. This project was a lot less ambitious than the one I had previously proposed, but I was able to finally iron out some stubborn wrinkles in my understanding of how to make things work in VBA.

Appendix 1: Sub GetData()

```
 Dim ColNames(3) As String
 ColNames(0) = "FIN.Transaction Date"
 ColNames(1) = "ACC.Account Name"
 ColNames(2) = "FIN.Transaction Amount"
 ColNames(3) = "ACC.State / Province"
Dim strFile As String
Dim strPath As String
Dim i As Integer
Dim x As Integer
Row = 1
    strPath = ThisWorkbook.Path & "\Data\"
    strFile = Dir(strPath)
    While strFile <> ""
        Application.Workbooks.Open (strPath & strFile)
            For x = 0 To 2
                Application.Workbooks(strFile).Activate
                If Application.CountIf(Range("A1:BB2"), ColNames(x)) = 0 Then
                    MsgBox ("Cannot find column " & ColNames(x) & ".")
                    Exit Sub
                Else
                    Cells.Find(ColNames(x)).Select
                    Range(ActiveCell, ActiveCell.End(xlDown)).Select
                    Selection.Copy
                    ThisWorkbook.Activate
                    Sheets("RawData").Cells(Row, x + 1).Activate
                    ActiveSheet.Paste
                End If
            Next
                Application.Workbooks(strFile).Activate
                If Application.CountIf(Range("A1:BB2"), ColNames(x)) = 0 Then
                Else
                    Cells.Find(ColNames(x)).Select
                    Range(ActiveCell, ActiveCell.End(xlDown)).Select
                    Selection.Copy
                    ThisWorkbook.Activate
                    Sheets("RawData").Cells(Row, x + 1).Activate
                    ActiveSheet.Paste
                End If
                ThisWorkbook.Activate
                Selection.End(xlDown).Select
                Row = Selection.Row + 1
                Application.Workbooks(strFile).Activate
                Application.CutCopyMode = False
                Application.Workbooks(strFile).Close (False)
        strFile = Dir
    Wend
End Sub
```

Appendix 2: TransPerAcct()

```
Worksheets("ScanCounts").UsedRange.ClearContents
Dim LastRow As Integer
Dim AccCol As Integer
Dim AmtCol As Integer
Dim MonCol As Integer

Worksheets("RawData").Activate

  Range("a1:bb3").Find("FIN.Transaction Date").Select
  ActiveCell.EntireColumn.Offset(0, 1).Insert
  MonCol = ActiveCell.Column + 1
  AccCol = Range("A1:BB2").Find("ACC.Account Name").Column
  AmtCol = Range("A1:BB2").Find("FIN.Transaction Amount").Column

  Cells(1, MonCol).Value = "Month"
  Columns(AmtCol).Name = "TransAmt"

 ' Dim TransAmtx As Range
 ' Set TransAmtx = Range("TransAmt")
 ' Dim Amtx As Range
  Dim counter As Long
  Dim z As Integer

'  For Each Amtx In TransAmtx.Cells
 For counter = 2 To Row
      With Sheets("RawData").Cells(counter, AmtCol)
        If (IsNumeric(.Value) = False Or .Value = "" Or .Value < -1000000) Then
          .EntireRow.Delete
          counter = counter - 1
          z = z + 1
        End If
      End With
    If z > 100 Then Exit For
Next counter

  Columns(AccCol).Select
  Selection.Copy
  Worksheets("ScanCounts").Activate
  Range("A1").Select
  ActiveSheet.Paste
  Application.CutCopyMode = False
  If Range("A1").Value = "" Then ActiveCell.EntireRow.Delete
  ActiveSheet.Range("A:A").RemoveDuplicates Columns:=1, Header:= _
    xlYes
  'Fills the # of transactions over the threshold
  Range("B1").Select
  ActiveCell.FormulaR1C1 = "Transactions Per Acct"
```

```
   Range("B2").Select
   ActiveCell.FormulaR1C1 = "=COUNTIFS('RawData'!C[" & AccCol - ActiveCell.Column & "],RC[-
1],'RawData'!C[" & _
   AmtCol - ActiveCell.Column & "],"">"" & ThresholdCol)"
   Range("B2").Select
   LastRow = Cells(Rows.Count, "A").End(xlUp).Row
   Selection.AutoFill Destination:=Range("B2:B" & LastRow)
   'Fills the % of transactions over the threshold
   Range("C1").Select
   ActiveCell.FormulaR1C1 = "% Over Threshold"
   Range("C2").Activate
   ActiveCell.FormulaR1C1 = "=RC[-1]/CountIf('RawData'!C[" & AccCol - ActiveCell.Column & "],RC[-2])"
   LastRow = Cells(Rows.Count, "A").End(xlUp).Row
   Selection.AutoFill Destination:=Range("C2:C" & LastRow)
   'Fills the Months across the top
   Range("D1").Select
   Dim x As Integer
   For x = 1 To 12
   ActiveCell.Value = MonthName(x)
   ActiveCell.Offset(0, 1).Activate
   Next
   'Inserts Month Names
   Dim TransMonth As String
   x = 1
   Dim blanks As Integer
   blanks = 0
  Do Until blanks = 100
  If Range("'RawData'!A" & x).Value = "" Then
  blanks = blanks + 1
  x = x + 1
  Else
   Select Case Left(Range("'RawData'!A" & x).Value, 2)
       Case "01", "1/"
       TransMonth = MonthName(1)
       Case "02", "2/"
       TransMonth = MonthName(2)
       Case "03", "3/"
       TransMonth = MonthName(3)
       Case "04", "4/"
       TransMonth = MonthName(4)
       Case "05", "5/"
       TransMonth = MonthName(5)
       Case "06", "6/"
       TransMonth = MonthName(6)
       Case "07", "7/"
       TransMonth = MonthName(7)
       Case "08", "8/"
       TransMonth = MonthName(8)
```

```
        Case "09", "9/"
        TransMonth = MonthName(9)
        Case "10"
        TransMonth = MonthName(10)
        Case "11"
        TransMonth = MonthName(11)
        Case "12"
        TransMonth = MonthName(12)
     End Select
     Worksheets("RawData").Cells(x, MonCol).Value = TransMonth
     x = x + 1
   End If
 Loop
    'Fills Transactions over threshold by month
    For x = 1 To 12
        Cells(2, 3 + x).Select
        ActiveCell.FormulaR1C1 = "=COUNTIFS('RawData'!C[" & AccCol - ActiveCell.Column & "],RC[" & -2 - x
& "]," & _
        "'RawData'!C[" & AmtCol - ActiveCell.Column & "],"">"" & ThresholdCol," & _
        "'RawData'!C[" & MonCol - ActiveCell.Column & "],R1C" & x + 3 & ")"
    Next
   Range("D2:O2").Select
    LastRow = Cells(Rows.Count, "A").End(xlUp).Row
    Selection.AutoFill Destination:=Range("D2:O" & LastRow)
    'Fills "Scans/Year Column
    Range("P1").Value = "Scans/yr"
    Range("P2").Select
    ActiveCell.FormulaR1C1 = "=Countif(RC[-12]:RC[-1],"">0"")"
    LastRow = Cells(Rows.Count, "A").End(xlUp).Row
    Selection.AutoFill Destination:=Range("P2:P" & LastRow)

End Sub
```

Appendix 3: CalculateFine()

```
If Application.CountIf(Sheets("RawData").Range("A1:BB2"), "Est Taxes") > 0 Then ' Check to see if the Est
Taxes field already exists
   MsgBox ("Data already imported.")
   Exit Sub
End If
Application.ScreenUpdating = False
Sheets("RawData").Activate
GetData
transPerAcct
Sheets("RawData").Select
If Application.CountIf(Sheets("RawData").Range("A1:BB2"), "ACC.State / Province") = 0 Then ' Check to
see if we know the state
      Range("A1").Activate                                    ' if no, then insert the tax fields to the left
and use the average US tax rate.
      Do Until ActiveCell.Value = "" And ActiveCell.Offset(1, 0).Value = ""
      ActiveCell.Offset(0, 1).Activate
      Loop
      ActiveCell.Value = "Est Taxes"
      ActiveCell.Offset(0, 1).Value = "Adjusted Tax"
   ActiveCell.Offset(1, 0).Activate
   r = ActiveCell.Row
   c = ActiveCell.Column

   TranCol = Cells.Find("FIN.Transaction Amount").Column
   ActiveCell.NumberFormat = "General"
   ActiveCell.FormulaR1C1 = "=IFERROR(RC" & TranCol & "*AvgTaxRate,0)"
   ActiveCell.Cells.Replace What:="=", Replacement:="="
Else
   Cells.Find("ACC.State / Province").Select
   ActiveCell.EntireColumn.Offset(0, 1).Insert
      ActiveCell.EntireColumn.Offset(0, 1).Insert
      ActiveCell.Offset(0, 1).Value = "Est Taxes"
      ActiveCell.Offset(0, 2).Value = "Adjusted Tax"

   ActiveCell.Offset(1, 1).Activate
   r = ActiveCell.Row
   c = ActiveCell.Column

   TranCol = Cells.Find("FIN.Transaction Amount").Column
   StateCol = Cells.Find("ACC.State / Province").Column
   ActiveCell.NumberFormat = "General"
   ActiveCell.FormulaR1C1 = "=IFERROR(RC" & TranCol & "*(VLOOKUP(RC" & StateCol & ",'State Tax
Rates'!A:B,2,FALSE)),0)"
   ActiveCell.Cells.Replace What:="=", Replacement:="="
End If

Dim LastRow
```

```
LastRow = Cells(Rows.Count, c - 1).End(xlUp).Row
ActiveCell.AutoFill Destination:=Range(Cells(r, c), Cells(LastRow, c))
Range(Cells(r, c), Cells(LastRow, c)).Name = "Tax"

Range(Cells(r, c + 1), Cells(r, c + 1)).Select
ActiveCell.NumberFormat = "General"
ActiveCell.FormulaR1C1 = "=IFERROR(IF(RC" & TranCol & "<ThresholdResearch,RC[-
1]*'Main'!R15C2*'Main'!R14C2,0),0)"

ActiveCell.AutoFill Destination:=Range(Cells(r, c + 1), Cells(LastRow, c + 1))
Range(Cells(r, c + 1), Cells(LastRow, c + 1)).Name = "TaxAdj"

Histogram
Sheets("Main").Select
Range("A1").Activate
Application.ScreenUpdating = True


End Sub
```

Appendix 4: Histogram()

```
    Sheets("Main").Select
    Range("A26:A51").Select
    Selection.Copy
    Sheets("RawData").Select
    Range("CC1").Activate
    ActiveSheet.Paste
    Selection.Name = "HistBin"
    Cells(1, TranCol).Activate
    ActiveCell.EntireColumn.Offset(0, 1).Insert
    Cells(1, TranCol + 1).Select
    ActiveCell.FormulaR1C1 = "=IF(ISNUMBER(RC[-1]),RC[-1],-1)"
    r = ActiveCell.Row
    c = ActiveCell.Column
    Dim LastRow
    LastRow = Cells(Rows.Count, c - 1).End(xlUp).Row
    ActiveCell.AutoFill Destination:=Range(Cells(r, c), Cells(LastRow, c))
    Range(Cells(r, c), Cells(LastRow, c)).Name = "HistData"
     Application.Run "ATPVBAEN.XLAM!Histogram", Range("HistData"), "", _
        Range("HistBin"), False, False, True, False

End Sub
```