Automated Sales Tax Calculator

David Prescott

ISYS 520

Brigham Young University

Executive Summary

For the past year, I have worked in the accounting department for an alarm installation company and I have been responsible for sales tax reporting. The alarm installation company installs and maintains low-voltage alarm systems. They partner with major alarm companies to install and service alarm systems in all 50 states. We have to file and pay sales tax in every state and by doing so are required to report our sales for each county in those states.

Filing sales tax is a time consuming process and each year we would spend more than 400 hours to completely file sales tax. For this project, I set out to automate the entire sales tax process to save time and improve accuracy. By using VBA I have created an automated calculator that has already proven to be very efficient. By implementing it in the last month, we have already seen a significant decrease in the time spent on sales tax. I am expecting that we will be able to lower the filing time down from days to hours each month and even reduce the total time spent on sales tax to less than 90 hours annually. This calculator is already reducing the time by more than 75% and is going to save the company more than $4,500 annually.

The purpose of the automated sales tax calculator to organize the company's sales data, calculate the amount of sales and deductions in each state and county, and display the results effectively to make filing sales tax significantly easier. The calculator is an asset to the company and has been dynamically developed in case of any changes to our system or changes with state tax laws. Therefore, allowing it to adapt to changes as they are made.

Background

Sales tax is a tax that is required to be paid on the sale of a good or service. Every state has different filing periods that are monthly, quarterly, bi-annually, annually and/or at other arbitrary periods. Each state has tax laws that differently define what items are taxable. For example, categories in the alarm industry, such as personal property (equipment), and labor (service or install) are taxed differently according to each state.

Generally, the seller charges the customer for sales tax and then pays that to the state. The seller in this situation is the company I work for and is considered a third party contractor and does not ever directly charge the customer. We instead, charge the partner company sales tax, who then charges the customer. Since we are the party who provides the equipment and labor and charge the contracted partner company, we invoice the majority of companies for the sales tax and file the sales tax amount. There are some companies who give us tax exemption forms, which means that we don't charge them sales tax and they become 100% responsible.

We use a software system called SecurityTrax to track and maintain all of our data pertaining to sales, payables, receivables, billing, etc. Each month, we download to Excel the desired time period from SecurityTrax and then go through each state and company and identify which companies are tax exempt, what are the state tax requirements, and where we did business. Previously, we would then have to use a rudimentary calculator to do basic lookups of certain criteria, but would have to do all the filtering and calculation manually. It was a long process that on the months we would just filed monthly returns; it would take one or two people about three days to complete. The quarterly filings would take seven to six days and for annual returns we would have three people working solely on sales tax for about eight to nine days. That adds up to more than 400 annual hours spent purely on sales tax.

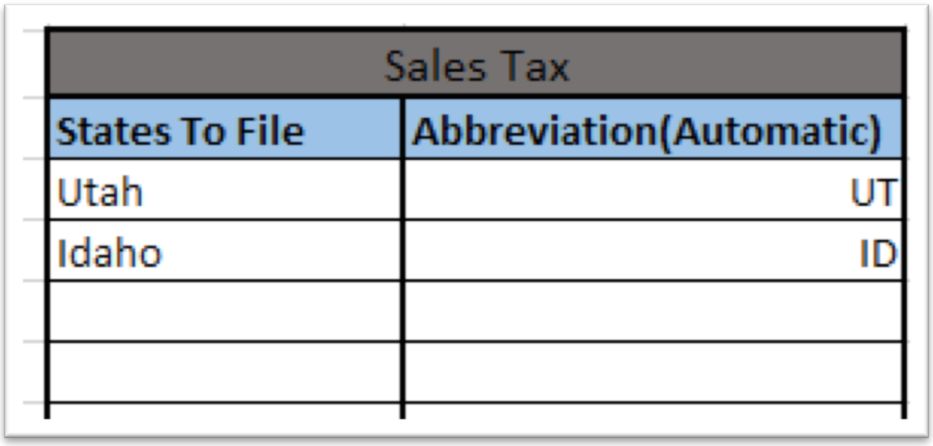
Problem:

The problem I set out to fix was two-fold. First, reduce the amount of time to calculate, file and pay sales tax and second, improve accuracy of calculating sales tax. The second issue is important because if there is a difference between filed and invoiced amounts, the company loses money and we have to spend time adjusting future invoices.

Solution:

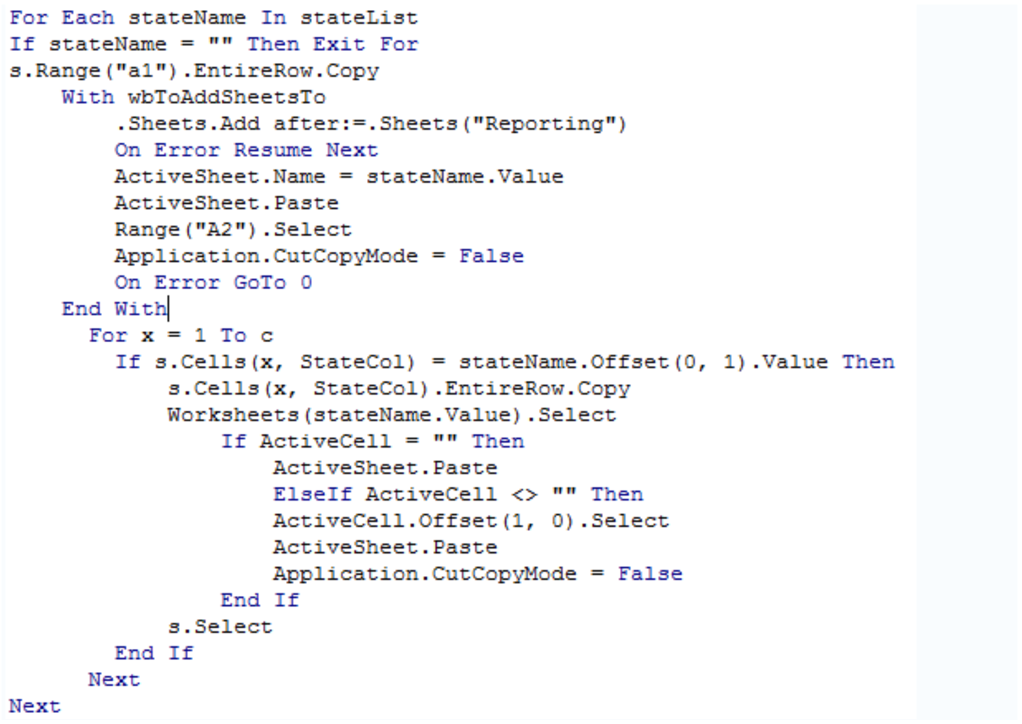
1) I started with cleaning up the front end of the Excel workbook by eliminated unnecessary columns and hiding certain columns to make the sheets easier to navigate (there were originally 70+ columns). This work was mainly done on the Calc worksheet. The previous format wasn't conducive to running a macro, so I prepared the worksheet by editing formulas, organizing columns, and fixing multiple mistakes in the calculator. By cleaning up the front end of the workbook, it is now capable of running a macro to calculate and display the results.

2) I first created a reporting worksheet and reporting table so the user can enter in the desired states needed to file sales tax (image of table is below). I considered using a user form but I found that using a table would be the best option for this project due to the fact that the user can simply copy and paste the monthly states or quarterly states into the table. Also the states change their filing periods quite frequently so it is best to give the user the flexibility to manually enter in different states for each period or the ability to perform other state tax calculations throughout the year. Additionally, we have another workbook that has all of the state login information, and filing periods so it is convenient to copy and paste information into the table.



Sales Tax	
States To File	Abbreviation(Automatic)
Utah	UT
Idaho	ID

3) Next I created a sub procedure called SalesTaxTFN that loops through the states on the sales tax table (example of Utah and Idaho in the image above) to create new worksheets with the state name. After a worksheet is created, the macro pulls all of the state specific data from the calculator on the Calc worksheet and places it into the newly created state-named worksheet. There is a worksheet that is now created for each state that needs to be filed that has all of the information to calculate sales tax. I have also included data validation tools to ensure that the states are spelled correctly as well as prevent duplicate entries. One part of the code uses the state abbreviations to lookup the tax criteria for that state and stores them as variables for later use to look up taxability criteria. Below is the code that loops through the table, creates a new worksheet and creates a sales tax calculator for the state specific information.



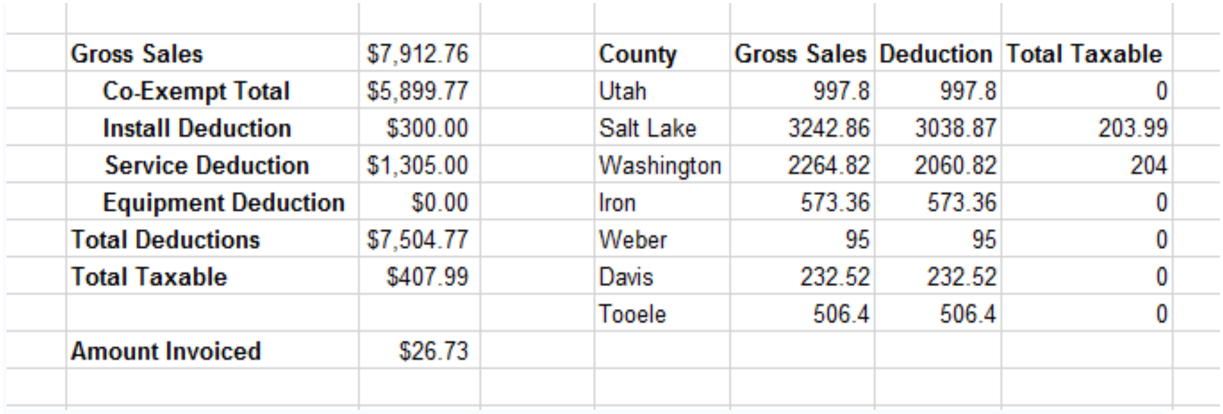
```
For Each stateName In stateList
If stateName = "" Then Exit For
s.Range("a1").EntireRow.Copy
    With wbToAddSheetsTo
        .Sheets.Add after:=.Sheets("Reporting")
        On Error Resume Next
        ActiveSheet.Name = stateName.Value
        ActiveSheet.Paste
        Range("A2").Select
        Application.CutCopyMode = False
        On Error GoTo 0
    End With
      For x = 1 To c
        If s.Cells(x, StateCol) = stateName.Offset(0, 1).Value Then
            s.Cells(x, StateCol).EntireRow.Copy
            Worksheets(stateName.Value).Select
                If ActiveCell = "" Then
                    ActiveSheet.Paste
                    ElseIf ActiveCell <> "" Then
                    ActiveCell.Offset(1, 0).Select
                    ActiveSheet.Paste
                    Application.CutCopyMode = False
                End If
            s.Select
        End If
      Next
Next
```

4) Next, I created a sub procedure called Calc that calculates the state gross income, exemptions deductions, install, service and equipment deductions, in order to find total deductions, taxable amount and total amount invoiced. The Calc sub procedure first loops through each state's tax requirements for each category (Install, Equipment, and Service) and saves those values to be referenced in later code. Every state requires filing at the state level and so I created nested If Then statements, and For Loops to calculate the total exemptions for each category, depending on the state's tax requirements.

   The If Then statements would search to see if the category is taxed, and if so it would then look to see if there are any exemption forms, and if not it would go through each line and sum the total amount for that category. It would do this for total deduction categories: exemption totals, equipment totals, install totals, and service totals. The macro would go through each line and calculate the totals, store them as variables and then display the results on the right side of the worksheet (image below shows results of step 5).

5) After calculating the state filing amounts, the sub procedure Counties calculates tax for each specific county by calculating gross sales, deductions, and total taxable amount. I used an array to select unique counties from the counties column and then calculated gross sales, deductions, and total taxable amounts by using For Loops and If Then statements. When filing sales tax, states require companies to report the overall state and each jurisdiction totals of gross sales and deductions, and then the state website calculates a tax payable amount. On the right side of the image below are

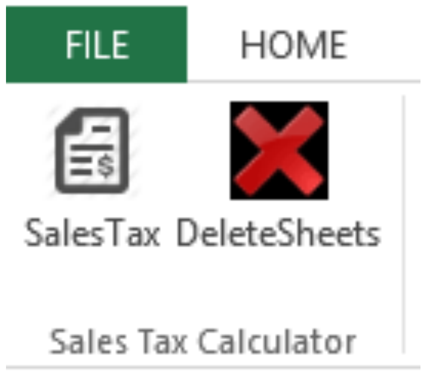
displayed results of this step, and shows each unique county and its specific amount.



Gross Sales	$7,912.76
Co-Exempt Total	$5,899.77
Install Deduction	$300.00
Service Deduction	$1,305.00
Equipment Deduction	$0.00
Total Deductions	$7,504.77
Total Taxable	$407.99
Amount Invoiced	$26.73

County	Gross Sales	Deduction	Total Taxable
Utah	997.8	997.8	0
Salt Lake	3242.86	3038.87	203.99
Washington	2264.82	2060.82	204
Iron	573.36	573.36	0
Weber	95	95	0
Davis	232.52	232.52	0
Tooele	506.4	506.4	0

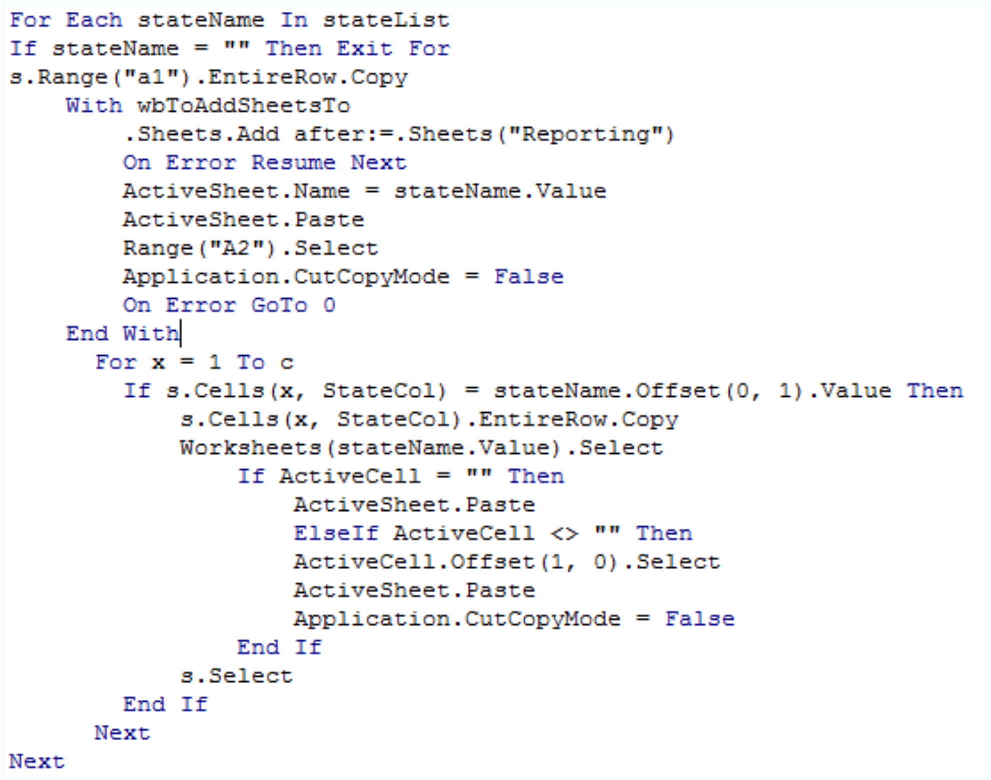
6) The last step was to customize the ribbon to make it easy for the user to run the automated calculator. By using Office Custom UI editor, I customized the ribbon to have a Sales Tax tab and have two underlying buttons: Sales Tax and Delete Sheets. The Sales Tax button is linked to sub a procedure that automates the calculator and displays the results in each created state worksheet. The delete sheets button deletes all sheets except the necessary calculation sheets (Calculator, Reporting, Exemption Forms and Zip Code Tax Rates) in order make it convenient to clean up the workbook and delete unnecessary state sheets. The ribbon helps the worksheet look more clean and professional and it adds to the accessibility of running the macro.



Learning and Conceptual difficulties Encountered

A difficulty I faced was that the downloaded file column locations changed in the middle of my project and will continue to change in the future. I had originally used columns locations as references (K:K, A:A, etc.), but realized I needed to reference the title of a column. Regardless of the column changes my code is dynamic and will reference the columns and ranges, which hold specific names such as State or County. After trial and error I found that I could run a loop to name each column based on the first cell in that row and then I would reference that named column or range for that worksheet.

However, the biggest difficulty I encountered was creating an array for unique counties and then using the data in that array to perform other calculations such as gross sales, deductions and taxable amounts. I am a novice at using arrays and it took a long time to conceptualize what I needed to accomplish. It was difficult to construct the entire configuration of the code to work properly because there are several criteria to meet and I had to use multiple For Loops, If Then statements, embedded For Loops and embedded If Then statements. I used my course book as a reference and built the array. The image below shows the building and use of the array and the final outcome.



```
For Each stateName In stateList
If stateName = "" Then Exit For
s.Range("a1").EntireRow.Copy
    With wbToAddSheetsTo
        .Sheets.Add after:=.Sheets("Reporting")
        On Error Resume Next
        ActiveSheet.Name = stateName.Value
        ActiveSheet.Paste
        Range("A2").Select
        Application.CutCopyMode = False
        On Error GoTo 0
    End With
      For x = 1 To c
        If s.Cells(x, StateCol) = stateName.Offset(0, 1).Value Then
            s.Cells(x, StateCol).EntireRow.Copy
            Worksheets(stateName.Value).Select
                If ActiveCell = "" Then
                    ActiveSheet.Paste
                    ElseIf ActiveCell <> "" Then
                    ActiveCell.Offset(1, 0).Select
                    ActiveSheet.Paste
                    Application.CutCopyMode = False
                End If
            s.Select
        End If
      Next
Next
```

Primarily, I learned that I can actually create sub procedures to solve business problems. Although I need to continue practicing and improving, I feel that I have a strong basis of VBA principles that I can use to create automated solutions/ programs for business, and personal use. I learned how to efficiently use arrays, object variables, For Loops, If Then statements, nested If Then statements and VBA syntaxes.

Assistance

I completed the project my self and did not receive substantial assistance. I used my course book (Albright, S. Christian. *VBA for Modelers)* to better understand and correctly using VBA syntax. I used the Internet to understand any errors I encountered.