Dalan Rohrer
April 2015
VBA Class

Stock Visualization and Socialization (**SVS**) Tool

Executive Summary

Sharing relevant and professional content on social media sites like Twitter and LinkedIn is absolutely essential to building a positive professional reputation and becoming a valued member of a professional social community. The benefits of belonging to such a community are job opportunities, personal brand-building, and education.

Sourcing that content, however, can be time consuming and overwhelming. Enter the SVS Tool! The SVS tool automatically pulls historical stock price information for up to 11 companies. Time period can be manually selected as can the increment of time measure. The stock price information is then automatically graphed. The color scheme can be easily adjusted to best suit the individual graph.

Once the graph has been created you can easily save and/or share the graph on social media sites. The SVS tool automates the process by allowing you to post the image of your graph to LinkedIn or Twitter with the click of a button.

Ultimately the SVS tool will enable more relevant professional content to be posted to your social media sites to better position you as a valuable member of a social community, increasing your ability to find job opportunities, build your personal brand, and educate yourself.

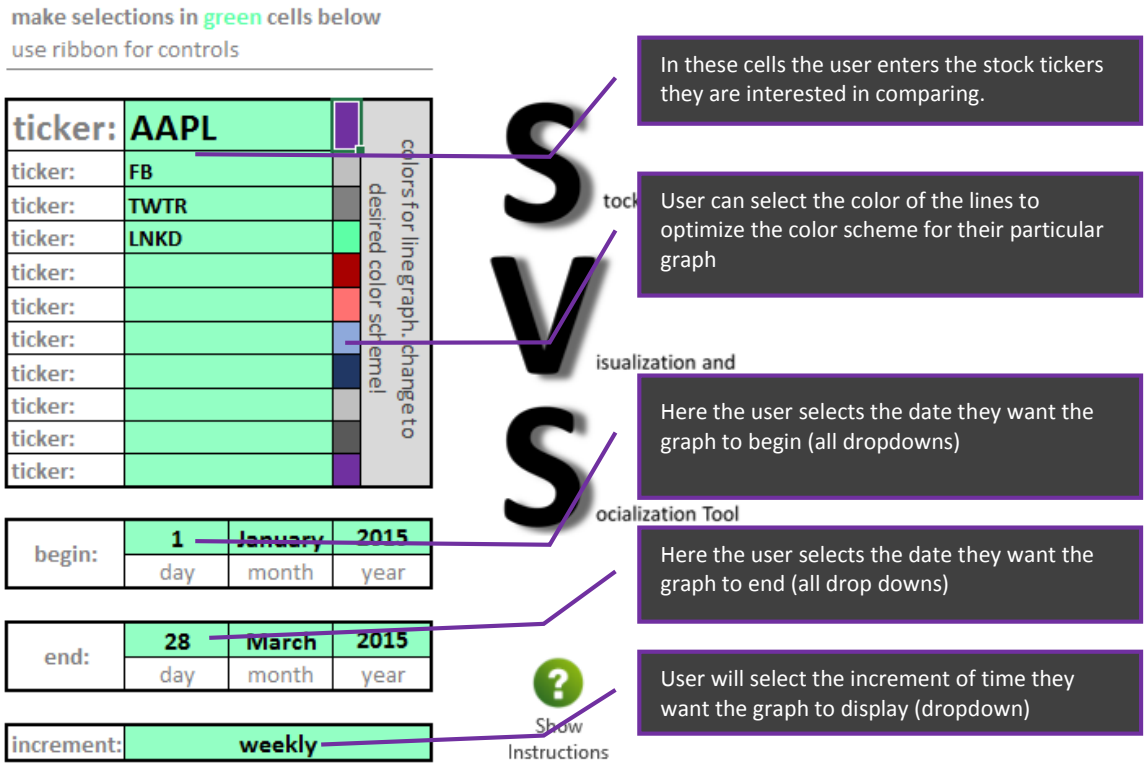
Dalan Rohrer
April 2015
VBA Class

Implementation Documentation

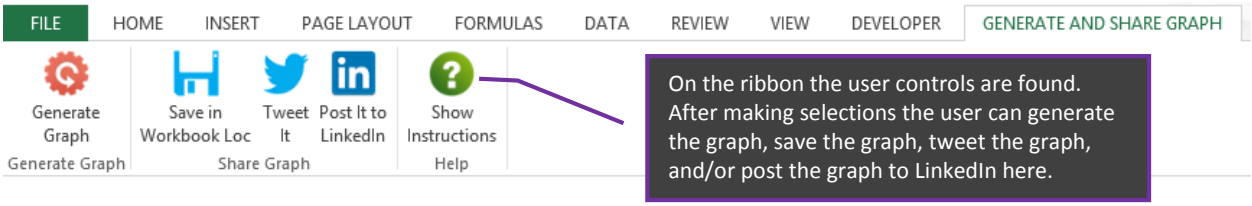
I will first describe the user interface, and then discuss the code behind the user interface.

I will first walk through the user interface: specifically the entry form, the user controls, the user forms, and the output.

Entry form

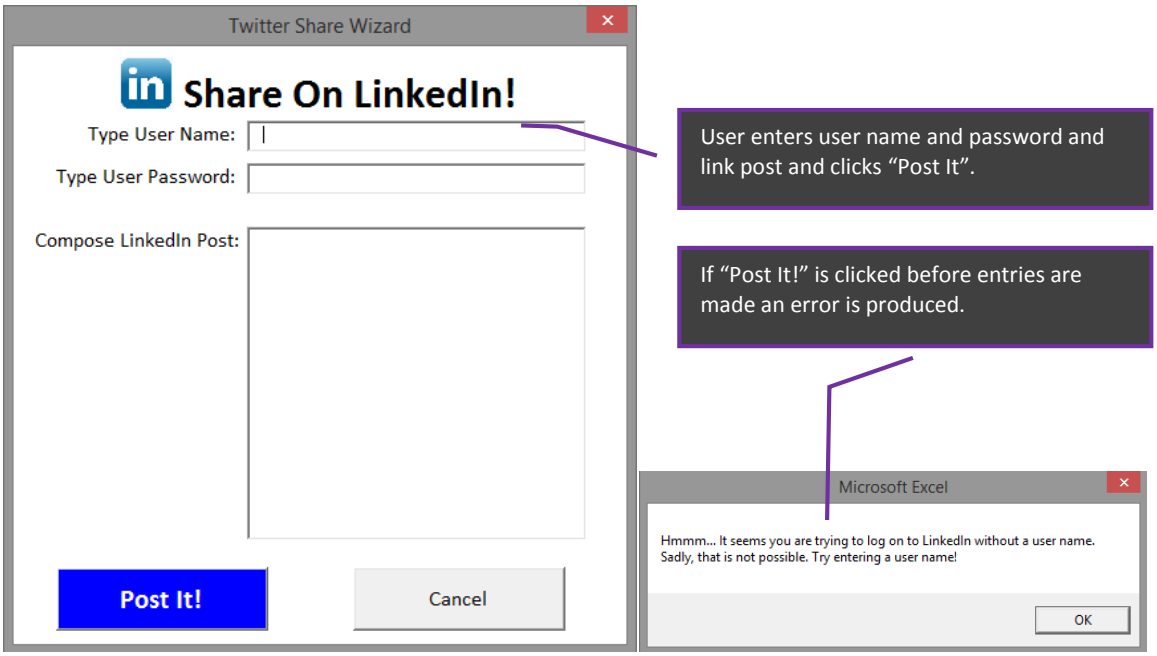


User controls

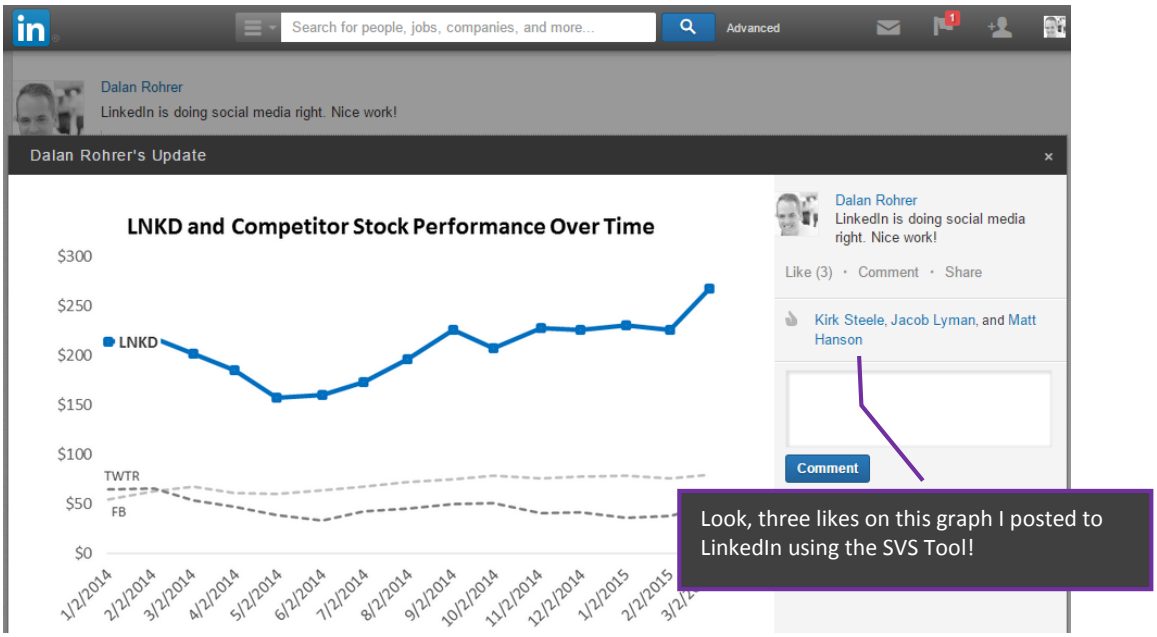


Dalan Rohrer
April 2015
VBA Class

User form



Output



Dalan Rohrer
April 2015
VBA Class

Below are brief summaries of the sub procedures that make the SVS Tool work. I will go into greater detail on each sub below the summaries.

- **generateChart**: runs the four sub procedures below with the ultimate goal of pulling historical price information from the internet on the stock tickers entered and graphing the those in a visually appealing way.
    - **allCaps**: a simple sub to make sure all the stock ticker are in capital letters. This is cosmetic.
    - **getTickerData**: this is the sub that goes to the internet to download the historical price information from the internet.
    - **fillChart**: grabs the information downloaded from the internet and fills the existing chart object with it.
    - **formatChart**: formats the existing chart object with the new data so the "main ticker" stands out from the other tickers and the lines are formatted according to the pre-selected color scheme.
- **saveChart**: saves the newly minted and formatted chart to the workbook location as an image.
- **loginLinkedIn**: takes the image of the new chart and posts it to LinkedIn along with whatever comment you dictate.
- **loginTwitter**: takes the image of the new chart and tweets in along with whatever comment you dictate.
- **showInstructions**: shows the instructions
- **ribbon controls**: sub procedures that are connected to the ribbon buttons and run the subs above.

generateChart

Dalan Rohrer
April 2015
VBA Class

```
Sub generateChart()

Application.ScreenUpdating = False

    allCaps
    getTickerData
    fillChart
    formatChart

Sheets("Company Overview Tool").Activate

Application.ScreenUpdating = True

End Sub
```

Makes sure the macro runs "behind the scenes" so it doesn't cause confusion

These four sub procedures run to pull the data and fill the chart

These four sub procedures run to pull the data and fill the chart

fillChart

```
Sub fillChart()
    Dim endRow As String
    Dim endRngAddress As String
    Dim X As Integer
    Dim numSer As Integer

    numSer = Sheets("Company Overview Tool").ChartObjects("Chart 1").Chart.SeriesColl

        On Error GoTo RS
        For X = 1 To numSer
            Sheets("Company Overview Tool").ChartObjects("Chart 1").Activate
            ActiveChart.SeriesCollection(1).DataLabels.Select
            Selection.Delete
        Next

RS:
        For X = 1 To numSer
            Sheets("Company Overview Tool").ChartObjects("Chart 1").Activate
            ActiveChart.SeriesCollection(1).Select
            Selection.Delete
        Next

    endRow = Sheets("Data").Range("a2").End(xlDown).Address
    endRngAddress = Sheets("data").Range(endRow).End(xlToRight).Address

    Sheets("Company Overview Tool").ChartObjects("Chart 1").Chart.SetSourceData Source:=Sheets("Data").Range("a1:" & endRng
```

This looks for existing data labels and deletes them. If there are none it causes an error so I skip this code on an error

Deletes all the lines in the existing graph. This is important to ensure formatting is accurate

Finds the address range of the data and assigns that address as the chart range.

allCaps

```
Sub allCaps()
Dim X As Integer
Dim upCase As String

    For X = 1 To 11
    upCase = (Sheets("Company Overview Tool").Range("o" & X + 4).Value)
    upCase = UCase(upCase)
    Sheets("Company Overview Tool").Range("o" & X + 4).Value = upCase
    Next

End Sub
```

Changes all stock tickers to capital letters for beautification purposes.

Dalan Rohrer
April 2015
VBA Class

getTickerData

```
Sub getTickerData()
    Dim tickerCount As Single
    Dim X As Integer
    Dim lastRow As String
    Dim dataLink As String
    Dim dataFile As String

    tickerCount = Application.WorksheetFunction.CountA(Sheets("Company Overview Tool").Range("o5:o15"))
    Sheets("Data").Cells.ClearContents

    For X = 1 To tickerCount

        Sheets("Company Overview Tool").Range("o" & X + 4).Copy _
            Destination:=Sheets("DataSearch").Range("d3")

        Sheets("DataSearch").Activate
        dataLink = "http://real-chart.finance.yahoo.com/table.csv?s=" & Range("d3").Value & "&a=" & Range
        dataFile = "table.csv"

        Workbooks.Open Filename:=dataLink
        Windows(dataFile).Activate

        lastRow = ActiveSheet.Range("a1").End(xlDown).Address
        ActiveSheet.Range("a2:" & lastRow).Copy _
            Destination:=Workbooks("Stock Visualization and Socialization Tool (SVS Tool)").Sheets("Data'
        Application.CutCopyMode = False

        lastRow = ActiveSheet.Range("a1").End(xlDown).Offset(0, 1).Address
        ActiveSheet.Range("b2:" & lastRow).Copy _
            Destination:=Workbooks("Stock Visualization and Socialization Tool (SVS Tool)").Sheets("Data'
        Application.CutCopyMode = False

        Workbooks("Stock Visualization and Socialization Tool (SVS Tool)").Act
        Sheets("DataSearch").Range("d3").Copy _
            Destination:=Sheets("Data").Range("a1").Offset(0, X)
        Application.CutCopyMode = False

        Application.DisplayAlerts = False
        Windows(dataFile).Close False
```

Counts the number of tickers entered

Copies the nth iteration of stock ticker and pastes to be used in web query

Downloads cvs file and modifies URL based on what was entered in the Excel spreadsheet

Copies the information in the cvs file and pastes it into the data tab of the excel workbook.

formatChart

```
Sub formatChart()
    Dim X As Integer
    Dim tickerCount As Single
    Dim color As Long
    Dim red As Integer
    Dim green As Integer
    Dim blue As Integer

tickerCount = Application.WorksheetFunction.CountA(Sheets("Company Overview Tool").Range(



For X = 1 To tickerCount

    color = Sheets("Company Overview Tool").Range("q5").Offset(X - 1, 0).Interior.color

    red = color Mod 256
    green = color / 256 Mod 256
    blue = color / 65536 Mod 256

    Sheets("Company Overview Tool").ChartObjects("chart 1").Activate

        ActiveChart.HasTitle = True
            With ActiveChart.ChartTitle
                .text = Sheets("Company Overview Tool").Range("b4")
                .Font.Size = 20
                .Font.Bold = msoTrue
            End With

    ActiveChart.FullSeriesCollection(X).Select
        With Selection.Format.Line
            .visible = msoTrue
            .ForeColor.RGB = RGB(red, green, blue)
            If X = 1 Then
                .DashStyle = msoLineSolid
                .Weight = 4
                Else
                .DashStyle = msoLineSysDash
                .Weight = 2.5
            End If
```

Counts tickers and begins a for loop that iterates through each line.

Finds the interior color of the cell adjacent to the ticker and transforms that color to an RGB to be applied to each line.

Changes the title of the chart to be a cell value that automatically returns the first stock ticker and competitor stock prices and sets the font size to 20.

Changes the lines to the rgb color, changes the weight of the lines, and for all but the first line makes them dashes.

(continued on next page)

```
        With Selection
            If X = 1 Then
                .MarkerStyle = xlMarkerStyleSquare
                .Points(1).ApplyDataLabels
                .Points(1).DataLabel.Select
                .HasLeaderLines = False
                Selection.ShowSeriesName = -1
                Selection.ShowLegendKey = 0
                Selection.ShowValue = 0
                Else
                .MarkerStyle = -4142
                .Points(1).ApplyDataLabels
                .Points(1).DataLabel.Select
                .HasLeaderLines = False
                Selection.ShowSeriesName = -1
                Selection.ShowLegendKey = 0
                Selection.ShowValue = 0
            End If

            With .DataLabels.Format.Fill
                .visible = msoTrue
                .ForeColor.RGB = RGB(255, 255, 255)
                .Transparency = 0
                .Solid
            End With

            If X = 1 Then
                .DataLabels.Format.TextFrame2.TextRange.Font.Size = 14
                .DataLabels.Format.TextFrame2.TextRange.Font.Bold = msoTrue
                Else
                .DataLabels.Format.TextFrame2.TextRange.Font.Size = 12
                .DataLabels.Format.TextFrame2.TextRange.Font.Bold = msoFalse
            End If
        End With

Next

End Sub
```

Assigns a data label to the line that is the series name, not the number.

Puts a white background on the data labels so it can sit on top of the line.

Changes data label for first to be larger and bold and for all others to be smaller and not bold

saveChart

```
Sub saveChart()
Dim currentFilePath As String

currentFilePath = ActiveWorkbook.path

    Sheets("Company Overview Tool").ChartObjects("chart 1").Activate
    ActiveChart.Export currentFilePath & "\stockChart.png"
```

Saves to workbook location

Dalan Rohrer
April 2015
VBA Class

loginLinkedIn

```
Sub loginLinkedIn()

Static a As agent
Dim tag As Object
Dim pos As Integer
Dim userName As String
Dim pass As String
Dim post As String

Application.ScreenUpdating = False

saveChart

Set a = Nothing
Set a = New agent

'gets info from user form
    userName = Workbooks("Stock Visualization and Socialization Tool (SVS Tool)").Sheets("DataSearch").Range("a60")
    pass = Workbooks("Stock Visualization and Socialization Tool (SVS Tool)").Sheets("DataSearch").Range("a61")
    post = Workbooks("Stock Visualization and Socialization Tool (SVS Tool)").Sheets("DataSearch").Range("a62")

'opens web page
    a.visible = True
    a.openpage "https://www.linkedin.com", True

'checks to see if logged in
If Len(a.document.url) < 30 Then

'logs in if not logged in as determined above
    a.document.all("session_key-login").Value = userName
    a.document.all("session_password-login").Value = pass
    a.waitForLoad
    a.document.all("signin").Click
End If

'switches IE to logged in window
a.waitForURLPartInDifferntBrowser ("?dnr")

'upload's photo
    For Each tag In a.document.all
      If LCase(tag.tagname) = "label" Then
        Debug.Print tag.innerhtml
        If tag.innerhtml = "Upload a photo" Then
            tag.Click
            a.waitForLoad
            Exit For
        End If
      End If
    Next
a.waitForLoad

'can I select the file for them?

'write's post
    a.document.all("postmodule-text").Value = post

'clicks the submit button for the post
    For Each tag In a.document.all
      If LCase(tag.tagname) = "button" Then
        pos = InStr(1, tag.outerhtml, "postmodule-submit", vbTextCompare)
            If pos > 0 Then
                tag.Click
                a.waitForLoad
                Exit For
            End If
      End If
    Next
    a.waitForLoad

Application.ScreenUpdating = True

End Sub
```

Gets the users information saved on the “datasearch” tab when they clicked the “post it” button in the user form and assigns it to these variables.

This opens the web page in IE and checks to see if the URL is longer than 30 characters (non-logged in URL is less and logged in is more, so this checks to see if user is already logged in). If the user is not logged in, this logs them in by finding the ID of the tags for the entry areas.

IE keeps both pages open, so this switches IE to the window with “dnr” in the URL, which is always the logged in window.

Clicks the upload photo button by finding the “label” tag that has “Upload a photo” as the inner HTML.

Puts the text in the post and finds the submit button by looking for the outer html of “postmodule-submit”, which is unique to the submit button.

Dalan Rohrer
April 2015
VBA Class

loginTwitter

```
Sub loginTwitter()
Dim tag As Object
Static a As New agent
Dim userName As String
Dim pass As String
Dim post As String
Dim pos As Integer
Dim X As Long

Application.ScreenUpdating = False

saveChart

Set a = Nothing
Set a = New agent

'gets info from user form
    userName = Workbooks("Stock Visualization and Socialization Tool (SVS Tool)").Sheets("DataSearch").Range("a60")
    pass = Workbooks("Stock Visualization and Socialization Tool (SVS Tool)").Sheets("DataSearch").Range("a61")
    post = Workbooks("Stock Visualization and Socialization Tool (SVS Tool)").Sheets("DataSearch").Range("a62")

'opens web page
    a.visible = True
    a.openpage "https://www.twitter.com", True


'checks to see if logged in
    a.position = 1
    If a.moveTo("signin-email") Then

'log in
        a.document.all("signin-email").Value = userName
        a.document.all("signin-password").Value = pass
        a.waitForLoad
'click sign in
        For Each tag In a.document.all
          If LCase(tag.tagname) = "button" Then
            If tag.innerhtml = "Log in" Then
              tag.Click
              Exit For
            End If
          End If
        Next
        a.waitForLoad
    End If

'clicks the camera button on the twitter feed to upload photo
        For Each tag In a.document.all
          If LCase(tag.tagname) = "span" Then
            pos = InStr(1, tag.outerhtml, "class=""visuallyhidden"">Add Photo", vbTextCompare)
            If pos > 0 Then
              tag.Click
              a.waitForLoad
              Exit For
            End If
          End If
        Next
        a.waitForLoad

'write's post
    a.document.all("tweet-box-home-timeline").innerhtml = "<div>" & post & "</div>"

'clicks the submit button for the post
    For Each tag In a.document.all
      If LCase(tag.tagname) = "button" Then
        pos = InStr(1, tag.outerhtml, "btn primary-btn tweet-action tweet-btn js-tweet-btn", vbTextCompare)
            If pos > 0 Then
                tag.Click
                a.waitForLoad
                Exit For
            End If
```

Gets the users information saved on the "datasearch" tab when they clicked the "post it" button in the user form and assigns it to these variables.

This opens the web page in IE and checks to see if "signin-email" is anywhere in the HTML, which is only on the logged in page. If the user is not logged in, this logs them in by finding the ID of the tags for the entry areas and searching for "log in" in a button tag to click

Clicks the camera button by finding a specific string in the outer HTML and clicking when it finds that tag.

Puts the text in the the ID of "tweet-box-home-timeline", and then putting the post in between the div (because this is a div not a button or label).

Clicks the submit button by finding the "button" tag that has unique inner HTML.

showInstructions

```
Sub showInstructions()
    frmInstructions.Show
End Sub
```

Shows the form I created with the instructions in it.

user form sub procedures (for LinkedIn and Twitter)

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

Private Sub cmdPost_Click()

        If txtUserNameLI.Value = "" Then
            MsgBox ("Hmmm... It seems you are trying to log on to LinkedIn without a user name. Sadly, that is not possible
            Exit Sub
        End If

        If txtPassLI.Value = "" Then
            MsgBox ("Passwords are certainly important to enable log on. Looks like you forgot or are trying magic. Either
            Exit Sub
        End If

        If txtPostLI.Value = "" Then
            MsgBox ("Just a heads up, you are Posting
            Exit Sub
        End If

        Workbooks("Stock Visualization and Socialization Tool (SVS Tool)").Sheets("DataSearch").Range("a60") = txtUserNameL
        Workbooks("Stock Visualization and Socialization Tool (SVS Tool)").Sheets("DataSearch").Range("a61") = txtPassLI.Va
        Workbooks("Stock Visualization and Socialization Tool (SVS Tool)").Sheets("DataSearch").Range("a62") = txtPostLI.Va

        loginLinkedIn

        Workbooks("Stock Visualization and Socialization Tool (SVS Tool)").Sheets("DataSearch").Range("a60:a62").ClearConte

Unload Me
End

End Sub
```

For the user forms this sub closes the form on cancel

Looks for any of the entry fields to be blank and produces an error message with instructions if the field is blank.

Runs the login sub procedure for whichever form

Discussion of Learning

This has been the most educational and rewarding portion of the class for me. Of course, it is only made possible by everything else we've done in class, but I have really enjoyed flexing my VBA muscles to see what all that work and learning can do, and I'm really excited about the skills I've gained in this course. This project specifically has engrained a few key lessons about writing VBA code: Google is my friend, recording macros can be better than Google, often solving the problem requires creativity, and debugging is the most important skill of all.

Google is my friend, I've learned. Much of my code is inspired by things I learned from sites like Stack Overflow. These sites give me direct access to professionals who just want to answer questions about VBA for free. One simple example is saving the chart as an image to a specific file location. Not only did I learn how to save it, I also saw the idea to always have it save to the current workbook location, which has become a big part of my code.

Recording macros can be far better than Google, however. This lesson was learned in absolute clarity when I wanted to format my chart in some fairly advanced ways (like changing the data labels to series labels and changing the weight and style of the lines). After spending a couple of minutes searching around the internet I simply turned on macro recorder, made all the changes I wanted to make, and checked out the code. Of course it needed some cleaning up and adjusting, but it gave me what I needed really quickly.

I also learned that what matters is that the code works so I could and should get creative when necessary. I definitely encountered this lesson when trying to find and click buttons and fields in IE using the agent class-level module that Dr. Allen created. While it may not have been the most elegant solution, I found that I could search for a string of text in the html by identifying the position of that text with instr() and then writing an if statement to see if the position was greater than 0 and click it.

Above all, though, I learned that debugging is the number one most important skill. There were dozens of times when I quickly found the general methodology I needed to use to achieve what I wanted to achieve, but it didn't work in my code. I had to use the immediate window to see what the code was evaluating to, and I also used the locals window to track what my variables are evaluating to. Often my code would "break" and I would have to step into each line of code to see where it broke and when I found the particular location I had to try a lot of different solutions and repeat the process over and over.

There were a couple of places where I got seriously stuck. One of those places was manipulating the LinkedIn page once logged in. I needed help from Dr. Allen (after several hours of troubleshooting with no luck) to ultimately discover that IE was keeping the login page open in the background even after logging in, and so I wasn't able to manipulate the logged in page. I also thought it would be cool to allow the user to select the file location where the file is saved and automatically choose that file when uploading to LinkedIn and Twitter, but I couldn't find an easy way to make that happen and I ran out of time.

Assistance

The only assistance I received was from Dr. Allen. He helped me with manipulating the IE pages, especially finding and clicking the appropriate portions of the pages.