Programming Microsoft Excel using VBA

A Brief Overview for AAII Quantitative Investing SIG

Vijay Vaidyanathan
Return Metrics Inc.
vijay@ReturnMetrics.com
WARNING!

This is a very brief overview of a semester-long course I teach to Master’s level students in Finance.

We’ll go too fast to be of immediate practical use.

Silicon Valley Techies may find it too superficial.

If you are not one of those, I hope you will get a flavor of what is possible with VBA for Excel.
Agenda

- Memory Lane: If you don’t know where you’ve been ...
- What’s in a macro?
- The VBA Environment: Editor, Debugger, Help, Browser
- Straight Lines: Assignment, Functions, Variables, Constants
- Zigging and Zagging: Branching and Looping
Agenda (contd)

- Reaching Inside: Excel Objects and Collections
- Basic Blocks: Ranges and Cells
- Example: Accessing SIPro Data via VBA
Excel VBA: Historic Synchronicity!

- BASIC language
- Kemeny and Kurtz, approx 1964
- Beginner’s All Purpose Symbolic Instruction Code
- Alternative to FORTRAN
- BASIC interpreter for M.I.T.S. ALTAIR
- Affordable “Kit Computer” for enthusiasts
PROJECT BREAKTHROUGH!

World’s First Minicomputer Kit to Rival Commercial Models...
“ALTAIR 8800”  SAVE OVER $1000

ALSO IN THIS ISSUE:
• An Under-$90 Scientific Calculator Project
BASIC for the Altair 8800

Need: make it more easy to program

Idea: Port BASIC

Hobbyists could then program Altair 8800 in BASIC!

Written by now-famous Harvard dropouts
A Clue?
VisiCalc™

How did you ever
do without it?

© 1979 PS Inc.

Circle 302 on inquiry card.

Solve your personal energy crisis.
Let VisiCalc™ Power do the work.

With a calculator, pencil and paper you can spend hours planning, projecting, writing, estimating, calculating, revising, erasing and recalculating as you work toward a decision.

Or with VisiCalc and your Apple II you can explore many more options with a fraction of the time and effort you’ve spent before.

VisiCalc is a new breed of problem-solving software. Unlike prepackaged software that forces you into a computerized straight jacket, VisiCalc adapts itself to any numerical problem you have. You enter numbers, alphabetic titles and formulas on your keyboard. VisiCalc organizes and displays this information on the screen. You don’t have to spend your time programming.

Your energy is better spent using the results than getting them.

Say you’re a business manager and want to project your annual sales. Using the calculator, pencil and paper method, you’d lay out 12 months across a sheet and fill in lines and columns of figures on products, outlets, salespeople, etc. You’d calculate by hand the subtotals and summary figures. Then you’d start revising, erasing and recalculating. With VisiCalc, you simply fill in the same figures on an electronic “sheet of paper” and let the computer do the work.

Once your first projection is complete, you’re ready to use VisiCalc’s unique, powerful recalculating feature. It lets you ask “What if?” examining new options and planning for contingencies. “What if” sales drop 20 percent in March? Just type in the sales figure. VisiCalc instantly updates all other figures affected by March sales.

Or say you’re an engineer working on a design problem and are wondering “What if that oscillation were dampened by another 10 percent?” Or you’re working on your family’s expenses and wonder “What will happen to our entertainment budget if the heating bill goes up 15 percent this winter?” VisiCalc responds instantly to show you all the consequences of any change.

Once you see VisiCalc in action, you’ll think of many more uses for its power. Ask your dealer for a demonstration and discover how VisiCalc can help you in your professional work and personal life.

You might find that VisiCalc alone is reason enough to own a personal computer.

VisiCalc is available now for Apple II computers, with versions for other personal computers coming soon. The Apple II version costs just $99.90 and requires a 32K disk system.

For the name and address of your nearest VisiCalc dealer, call (408) 743-7841 or write to Personal Software, Inc., Dept. P, 592 Weddell Dr., Sunnyvale, CA 94086. If your favorite dealer doesn’t already carry Personal Software products, ask him to give us a call.

CIRCLE 2
Spreadsheet History

First “Real App” for PCs
First real “Killer App”
500 copies per month to 12,000 (1979 -> 1981)
Dan Bricklin and Bob Frankston
Visicalc for Apple II
By 1980 Apple had 50% of the PC market!
Software never patented
Visical->SuperCalc->Lotus123
What is VBA

An embedded interpreter for Visual BASIC

Embedded in the Application

Excel, PowerPoint, Word

Allows “programmatic control”

Customize and Extend, Build User Interfaces ...

You are:

A software developer and computer programmer!
Basic Concepts

- You write code in BASIC
- BASIC interpreter reads your instructions
- The interpreter has access to the insides of Excel
- It performs the task you tell it to
- You give a sequence of statements
- Interpreter goes from Top to Bottom
- One statement at a time!
Examples

- MsgBox "Hello World!"
- Range("A1") = "Wow, it works!"
- MsgBox Application.UserName
- Range("A1:A2") = Application.UserName
- Range("A1:A2").ClearContents
- Range("B2").Offset(-1, -1) = _
  Range("B2").Value
Basic Concept: You can do just two things!

**Statements: “Do Something”**

**Expressions: “Return a value”**

Simplest “Do Something” - Assign a value to a variable

Assigning to “special” objects, modifies the spreadsheet!
Simplest way to get started

Let Excel write some code for you!

Macro Recording

Example: Time Stamping

Record a macro to TimeStamp the Worksheet

Basic Idea:

Insert today’s date on Top Left Cell

Let’s do it!
Look at the Generated Code

- Sequential
- Verbose!
- “Dumb” code
- Cannot “branch”
  - E.g. if today is Saturday, make it red
- Cannot “Loop”
  - E.g. TimeStamp all open worksheets
More serious problem

- It is wrong!
- Date gets updated everytime the Worksheet is opened!
- How could we fix it?
- Simplest way is to edit the generated code
Sub DateStamp()
  
  ' DateStamp Macro
  ' Macro recorded 10/12/2008 by Vijay Vaidyanathan
  
  ' Keyboard Shortcut: Ctrl+d
  
  Range("A1").FormulaR1C1 = "Last Review"
  Range("B1").FormulaR1C1 = "=TODAY()"
End Sub
Sub DateStamp()
    ' DateStamp Macro
    ' Macro recorded 10/12/2008 by Vijay Vaidyanathan
    ' Keyboard Shortcut: Ctrl+d
    ' Write the current date into Top-Left Cells
    Range("A1").Value = "Last Review"
    Range("B1").Value = Date
End Sub
For you to play with later:
Add Intelligence

If it is Saturday, make it RED

How could you do this with the Macro recorder?

Answer: You really can’t!
Sub DateIt()

    ' Write the current date into Top-Left Cell
    Range("A1").Value = Date

    ' Check if it is a Saturday
    If DatePart("w", Date) = vbSaturday Then
        Range("A1").Font.Bold = True
        Range("A1").Font.Color = RGB(255, 0, 0)
    End If

End Sub

Now, it seems to work as intended!

BUT - At least two problems with this code!
Sub DateIt()

    ' Write the current date into Top-Left Cell
    Range("A1").Value = Date

    ' Check if it is a Saturday
    If DatePart("w", Date) = vbSaturday Then
        Range("A1").Font.Bold = True
        Range("A1").Font.Color = RGB(255, 0, 0)
    Else
        Range("A1").Font.Bold = False
        Range("A1").Font.Color = RGB(0, 0, 0)
    End If

End Sub  ' Why is this Better? What’s wrong with this?
Bug Report

Occasionally, does not turn RED, but Date is correct

Hint: Only happens when user is working late!

Hint: The problem is more obvious if I follow some coding “best practices”
Sub DateIt()

    ' Write the current date into Top-Left Cell
    Range("A1").Value = Date()

    ' Check if it is a Saturday
    If DatePart("w", Date()) = vbSaturday Then
        Range("A1").Font.Bold = True
        Range("A1").Font.Color = RGB(255, 0, 0)
    Else
        Range("A1").Font.Bold = False
        Range("A1").Font.Color = RGB(0, 0, 0)
    End If

End Sub ' Why is this Better? What’s wrong with this?
Diagnosis

Answer: a sort of a “Race Condition”

The Date might change between the first and second calls to Date()

Moral: Debugging is tricky, good bug reports are valuable!
Sub DateIt()

    Dim rightNow as Date

    rightNow = Date()

    ' Write the current date into Top-Left Cell
    Range("A1").Value = rightNow

    ' Check if it is a Saturday
    If DatePart("w", rightNow) = vbSaturday Then
        Range("A1").Font.Bold = True
        Range("A1").Font.Color = RGB(255, 0, 0)
    Else
        Range("A1").Font.Bold = False
        Range("A1").Font.Color = RGB(0, 0, 0)
    End If

End Sub ' This works as intended
Getting Started with the VBE/IDE
The Visual Basic Editor

- Project Explorer window
- Properties window
- Immediate Window
- Code Area (Currently empty)
The Toolbar

“Run” button

Note “VCR” like buttons
The Project Explorer Window

- Project Explorer
- Microsoft Excel 2003 objects
- VBA code modules
The Properties Window

Select a property by clicking its name.

Click on "+" or "-" or the folder to expand the item and see the contained items.

Select an Object.

Double-click a property box to set its value. This is the Object's name.

Try renaming the sheet from "Sheet1" to "Test Sheet".

Warning: This is confusing: Property "Name" is different from the Name of the Object!
The “Immediate” Window

This is a weird box, since it is both an Input Box and an Output Box!
Try deleting the text in the box and entering new text
You can enter VBA commands followed by the RETURN key to execute the command
The Code Window

View buttons: Full Module or just a single Procedure

Procedure Name

Declarations Section

Object box

Procedure box

Option Explicit

Sub DateTime()

' DateTime Macro
' Macro recorded 10/12/2008 by Vijay Vaidyanathan

' Keyboard Shortcut: Ctrl+d

' Write the current date into Top-Left Cells
  Range("A1").Value = "Review Date"
  Range("B1").Value = Date

End Sub
Editing VBA Programming Code

Note how the Syntax Directed Editor and Integrated Help can tell you what it expects.

Warning: It is usually helpful, but not completely reliable!
Dim rightNow As Date
rightNow = Date
'Write the current date into Top-Left Cell
Range("A1").
'Check if
If DatePart
  Range
  Range
Else
  Range("i")
End If
End Sub 'WHY IS THIS "EVEN BETTER??"
"Calling" (aka "Invoking" or "Running") your code ...

Several ways to call/invoke/run a procedure

- choosing a menu sequence
- entering the assigned shortcut keystrokes
- clicking a macro button embedded in a worksheet
- clicking a macro button added to a new or existing toolbar.
Stepping through code

The yellow arrow and code highlighting indicate the line to be executed next.

Margin area to create breakpoints.
Setting a Breakpoint

The red circle and highlighting indicate that a breakpoint has been set.
The VB Language
VB Language Basics

- Variables and Objects
- Subs and Functions
- Branching
- Looping
- Excel Object Model
Variables hold values

e.g. \( x = 4, \ y = 2, \ z = x+y, \ \text{circ} = \pi*d \)

In VB, you **need** not declare variables, but you **should**!

Declarations:

```
Dim <varname> As <Type>
```

Types: Integer, String etc.

```
Dim Var as Type, ... , Var as Type
```
Main VBA Types

- **Boolean, Byte**: e.g. Dim enuf as Boolean
- **Integer, Long**: Integer is 2 bytes, Long is 4 bytes
- **Single, Double**: Floating Point
- **Date**: 8 bytes, nightmare, two different conventions!
- **String**: Variable length with limit depending on OS ver
- **Object**: Built-in Excel objects (user definable as well)
- **Variant**: “Untyped” – can take on type at runtime
- **User Defined types are possible, we won’t get to them**
Declare and Assign

```
Dim factorial as Long
factorial = factorial*i
```

Assignment updates the value

Use "=" to assign an expression value to a variable
Always Declare your Variables

Visual Basic does not (by default) require you to declare variables.

If it doesn't find one, it just makes up a declaration for you.

This is an extremely terrible idea, you’ll see why.

Use “Option Explicit” in your module.

VBA preference “Tools/Options”
WkstNumber = 10

if (WsktNumber = 10) Then
    Debug.Print "Yes, it is 10"
else
    Debug.Print "No, it is not 10"
End If
Do it NOW: (Tools/Options)
Objects: Data + Methods

A simple variable can only contain a value, but you have to write the code to modify it’s value, or use it in an expression.

An Object contains the code to modify or retrieve it’s own properties (i.e. data values).

An Object has a Class, not a Type. The Class provides the Template for all Objects of that Class.

e.g. Debug.Print Worksheets.count
Comments

Any part of a line after a single apostrophe (’) is treated as a comment

It is COMPLETELY IGNORED by VBA

It is only useful as documentation

Use it generously!

Good code is commented code!
Summary: Variables and Objects

- Variables are of basic built-in types: Integer, Long, String, Boolean
- Declared using DIM
- Assigned using “=”
- Printed or assigned to cells

- Objects encapsulate property data with methods to manipulate them
- Compound structures
- Assigned using SET
- Cannot be printed or assigned to cells
- Access parts of it using the “.” operator and the WITH keyword
Expressions

Expressions are of a Type or Class as well

e.g. \(i \times \text{factorial}\) is of type Integer since \(i\) and \(\text{factorial}\) are of type Integer

Arithmetic: \(*\), \(+\), \(-\), \(/\), \(\backslash\) (integer div), \(^\), \(\mod\)

String: Concatenation with &

Logical: \(\text{AND}, \text{OR}, \text{NOT}, =, <, >, <=, >=, <>\)

Ternary and n-ary: iif, switch
Use Debug.Print to play!

Debug.Print "There are " & Sheets.Count & " Sheets"
Debug.Print "The OS is " & Application.OperatingSystem
Debug.Print "4 x 5 is " & (4 * 5)
Debug.Print "The answer is " & Iif(good, ":-)", ":-(")
Debug.Print "File Type" & switch(Ext="pdf", "Adobe PDF", _
    ext="txt", "Text File", ext="xls", "Excel Spreadsheet")
Arrays and Collections

The types we have seen so far are scalars

Often we want to store several values in a variable

Two options: Arrays and Collections
Arrays

Arrays variables hold more than one value

Each value is accessed through an index

e.g. Dim interest_rates(1 to 10) as Double
    balance = balance * interest_rates(4)

Multidimensional Arrays are okay too!

e.g. Dim Covar(1 to 10, 1 to 10) as Double
Collections

Collection is a special type of container Class

The plural form (e.g. Sheets) is a hint it is a Collection

if c is a Collection, you can access its Count property

You can also index it like an array, but it is NOT an array

You can index it by its Name

Debug.Print Sheets(1).Name
Debug.Print "How many sheets does we have?"
Debug.Print Sheets.Count
Debug.Print Sheets("Sheet 1").UsedRange.Count
Branching: If-Then-Else, Case

To control program flow:

If-Then, If-Then-Else, Select-Case

If Boolean Condition Then
Some Statements
End If

If Boolean Condition Then
Some Statements
Else
Some Other Statements
End If

Select Case expression
Case value1
Some Statements
Case value2
Other Statements:
Case Else
Default Statements
End Select

See Help (F1) for more
Branching with IF

If (n mod 2) = 0 Then
    Debug.Print "n is an even number"
Else
    Debug.Print "n is an odd number"
End if

Case Select ext
    Case "xlt"
        FileType = "Template"
    Case "xls"
        FileType = "Worksheet"
    Case "xla", "utl"
        FileType = "Addin"
    Case Else
        FileType = "Mystery"
End Select
Interacting with the User

VBA provides two functions for use in programming simple user interaction.

- **MsgBox**
- **InputBox**

These functions enable you to display and collect information using built-in dialog box forms.
Iteration/Looping

- Do-Loop with While or Until
- For-Next-Step
- For Each-Next
- Exit-For and Exit-Do
Simple Loops: Do-While

1. **Do While** Boolean Condition
   - Some Statements
2. Loop

3. **Do Until** Boolean Condition
   - Some Statements
2. Loop

4. **Do**
   - Some Statements
   - Loop While Boolean Condition

5. Ctrl-Break is your FRIEND!
   - (Mac Emulators, beware!)
Exact checks for equality of floating point numbers is dangerous

Control-Break (Mac Emulation Users-Watch out!)
For-Next

Very popular kind of loop

Built-in incrementing and termination condition

For Var = InitVal To FinalVal
Some Statements Using Var
Next Var

Dim i as Integer
For i = 1 to Sheets.Count
    Debug.Print "Sheet " & i & " is named " & Sheets(i).Name
Next i
For-Each

Cool way to iterate through Collections

Much less error-prone

Can NOT have Step

Dim Wkst as Worksheet
For Each Wkst in Sheets
    Debug.Print "I have a sheet named " & Wkst.Name
Next Wkst
Nested Loops

You often nest loops within each other

Dim row as Integer, col as Integer

for row = 1 to 10
  for col = 1 to 10
    Cells(row, col).Value = row * col
  next col
next row
Subs and Functions

We have already seen Sub

Program Control flows to the Subroutine Code

On exit from the Sub, returns to where it was called from

Functions are similar, but return Values to Expressions

Functions can take parameters just like Sub

We’ve already seen built in functions and subs

MsgBox (sub), UBound (function)
Function FunctionName(Param as Type, ..., Param as Type) as Type

Some Statements that assign to FunctionName

End Function

Function IsPrime (Number as Integer) as Boolean

    Dim try as Integer
    IsPrime = True 'assume T until a divisor is found
    If (Number mod 2) = 0 Then
        'it is even, so can't be prime
        IsPrime = False
    Else 'try Odd numbers
        for try = 3 to Number - 1 Step 2
            If (Number mod try) = 0 Then
                IsPrime = False
                Exit For 'we know this is not a prime
            End If
        Next try
    End If
End Function

print IsPrime(3) True
print IsPrime(101) True
print IsPrime(89) True
print IsPrime(81) False
print IsPrime(100) False
Function SharpeRatio(R As Double, Rf As Double, StdDev As Double)
    SharpeRatio = (R - Rf) / StdDev
End Function

Debug.Print SharpeRatio (.08, .04, .035)
    1.14285714285714
Debug.Print SharpeRatio (Rf := 0.04, StdDev := 0.035, R := 0.08)
    1.14285714285714

Function SharpeRatio(R As Double, Optional Rf As Variant, Optional StdDev As Variant)
    if (IsMissing(Rf)) Then Rf = 0.04
    if (IsMissing(StdDev)) Then StdDev = 0.035
    SharpeRatio = (R - Rf) / StdDev
End Function

Debug.Print SharpeRatio(.08)
Debug.Print SharpeRatio(.08,,.035)
User Defined Formulas

Functions in Modules are available as User-Defined Formulas

A Private Function is not visible as a formula

A GREAT reason why functions should be side-effect free!
Sub (Param as Type, ... , Param as Type)
Some Statements
End Sub

Sub PrintPrimeTable(start As Integer, Optional Num As Variant)
    Dim i As Integer
    Const startRow = 1
    Dim rowNum As Integer
    rowNum = startRow
    If (IsMissing(Num)) Then Num = 10
    For i = start To start + Num
        Cells(rowNum, 1).Value = i
        Cells(rowNum, 2).Value = IsPrime(i)
        rowNum = rowNum + 1
    Next i
End Sub

<table>
<thead>
<tr>
<th></th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FALSE</td>
</tr>
<tr>
<td>2</td>
<td>FALSE</td>
</tr>
<tr>
<td>3</td>
<td>FALSE</td>
</tr>
<tr>
<td>4</td>
<td>TRUE</td>
</tr>
<tr>
<td>5</td>
<td>FALSE</td>
</tr>
<tr>
<td>6</td>
<td>TRUE</td>
</tr>
<tr>
<td>7</td>
<td>FALSE</td>
</tr>
<tr>
<td>8</td>
<td>FALSE</td>
</tr>
<tr>
<td>9</td>
<td>FALSE</td>
</tr>
<tr>
<td>10</td>
<td>TRUE</td>
</tr>
<tr>
<td>11</td>
<td>FALSE</td>
</tr>
<tr>
<td>12</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
The Excel Object Model
Excel and Ranges

There are hundreds of Excel Objects, but ...

- Workbooks
- Worksheets/Sheets
- Range
- Cells
- Offset
Workbooks

Application.Workbooks collection

Private Sub testit()
    Dim wb As Workbook
    For Each wb In Workbooks
        Debug.Print "Name: " & wb.name
    Next wb
End Sub

Immediate

Print Workbooks.Count
2
Name: TradeAnalyzer-v11.xls
Name: Book1
Worksheets and Sheets

Application.Worksheets

Private Sub testit()
    Dim ws As Worksheet
    For Each ws In Worksheets
        Debug.Print "Name: " & ws.name
    Next ws
End Sub

Private Sub testit()
    Dim ws As Worksheet

    For Each ws In Application.Workbooks("Book2").Worksheets
        Debug.Print "Name: " & ws.name
    Next ws
End Sub
Range

Excel’s Workhorse Object

Range("A2")

Range("A2:C4")

Range("A2:C4").Range("B1")

Range("A2:C4").Range("B1").Address
Experiment with Range

```plaintext
print Range("A2").Address
$A$2

print Range("A2:C4").Address
$A$2:$C$4

print Range("A2:C4").Count
9

print Range("A2:C4").Cells(7).Address
$A$4

print Range("A2:C4").Range("B1").Address
$B$2
```
Cells

- Collection of Cells in the specified Range Object
- Returns a Range Object
- Can access by Row and Column, or Linearly
  - `Range("B2:D5").Cells(1).Address = $B$2`
- `Cells(0)` can give weird behavior, watch out!
Cells - Linear Access

Private Sub testit()
    Dim c As Range
    Dim i As Integer
    i = 1
    For Each c In Range("B2:D5")
        c.Value = i
        i = i + 1
    Next c
End Sub

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>
Private Sub testit()
    Dim r As Integer, c As Integer
    For r = 1 To 3
        For c = 1 To 3
            Range("B2:D5").Cells(r, c) = r & "," & c
        Next c
    Next r
End Sub
Useful way to compute relative cell location

Range.Offset(0, 0) is the range itself, -ve offsets are okay

print Range("C2").Offset(0, 0).Address
$C$2

Print Range("E4:G6").Offset(-2, -1).Address
$D$2:$F$4

Offset(-2, -1)
Some Programming Examples
Simple Example: AltRows

Setting the colors of alternate rows in a range

Strategy:
- Find the Selected Region
- Skip through them in pairs
- Assign the Color to that row

Objects
Sub AltRows()
    Dim theRow As Range
    Dim rowNum As Integer
    For rowNum = 1 To Application.Selection.Rows.Count Step 2
        Set theRow = Application.Selection.Rows(rowNum)
        theRow.Interior.Color = RGB(128, 128, 128)
    Next rowNum
End Sub

"Application" is always visible
Can omit, but try not to
RGB(128, 128, 128) is gray (or grey!)
SIPro is a Visual Foxpro Database

Download the (free) ADODC driver from Microsoft

You can then query the Database using ADO and SQL

Do we have time? Simple SIPro-VBA Demo