Northeastern

VBA Training

Healthcare Systems Engineering Institute

Northeastern University, Boston MA

June 10, 2015

www.hsye.org

Center for Medicare & Medicaid INNOVATION

CHOT Center for Health Organization Transformation

VERC New England Veterans Engineering Resource Center

HSyE

Why VBA?

- Pros
  - Built into Excel (yay!)
  - More powerful than Excel formulas
- Cons
  - Difficult to debug
  - Weaker and slower than C++/Java/MATLAB
  - Doesn't print well

Two Schools of Thought

- Sam

"Anything that can be done in Excel should be done in Excel, even complex formulas. Only use VBA when necessary, and use it to drive Excel"

- Samy

"VBA is a coding language, so use it as such. If it's simpler to code something in VBA than Excel, do so. Use Excel as I/O, and let VBA do the heavy lifting"

VBA Basics 1

- Variables
  - Integer
  - Double
  - Boolean
  - String
- Input/output
  - Range
  - Named Ranges
  - Cells
- Logic
  - If then/else if/else end if
  - For (next)
- Arrays
  - Declare
  - Redim
  - Cell reference
- Random variables
  - RND
  - Calculations in VBA

VBA Basics 2

- Solver
  - Add-in
  - Run
- Security
  - Data validation
  - Protect/unprotect
- Worksheet Functions
  - Selection
  - Activation
  - Worksheet function
- Misc.
  - Option explicit
  - Screen updating
  - Calculate and manual/auto
  - Status bar and clear
- Cell Functions
  - Copy/paste or =
  - Paste special
  - Offset
  - Value
  - Select
  - Clear contents/formats
  - Delete
  - Hidden/visible
- Macros
  - Call
  - Function
  - Recording
  - Run (Alt-F8)
  - Code (Alt-F11)
  - Step through (F8)

Case Study

- You run an outpatient infusion clinic and wish to cover your patient demand in a cost-effective manner
- Problem Structure:
  - Open 6a-6p
  - Up to 100 patients
  - Two types (short, long)
  - Nurse to patient ratio = 2
  - Cost per 8-hr shift = $400
  - LOS distribution is Normal

Simulation Pseudo Code

Define variables and arrays
Assign user inputs to variables
Turn off screen updating and auto calculations

For all replications
    For all patient appointments
        If the appointment type is “long”
            Get a random LOS from the long appointment distribution
        Else (the appointment type is “short”)
            Get a random LOS from the short appointment distribution
        End if
    Next patient
    Add output (patient demand by hour) to the output array
    (at the end this will be the sum of demand per hour over all replications)
Next replication

Divide output array values by total replications to get average demand
Write output to optimization sheet
Turn on screen updating and auto calculations

```
'*************************************************
'Nurse Demand Simulation Model
'Developed by Kendall Sanderson and Sam Davis
'Contact Information: sanderson.k@husky.neu.edu
'Last Modified 6/9/2015
'*************************************************


'*****This forces you to define each variable explicitly before using it which prevents numerous errors when debugging
Option Explicit

'*****This is the name of the subroutine and the name used to reference this macro
Sub Simulate()

'Instantiate variables that will be user-defined, all of which are integers
Dim Reps As Integer
Dim Patients As Integer
Dim TotalHours As Integer

'Set variables to user-entered values
'*****Note how we need to first instantiate and then set variables
'*****These pull from named ranges in Excel. If they weren't named, "Reps" would be "D2"
Reps = Worksheets("Simulation").Range("Reps")
Patients = Worksheets("Simulation").Range("Patients")
TotalHours = Worksheets("Simulation").Range("TotalHours")

'Variables used as loop counters
'*****We normally don't define these in coding languages, but need to b/c "Option Explicit" above
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim m As Integer

'Name of output array
'*****Will be used to store results from each replication
Dim num_pts_per_hour() As Integer

'Define size of array
'*****Note how we need to size the array AFTER defining it. This is b/c VBA doesn't let you define and size an array at the same time if the size is a variable
'*****When defining array size, define the number of buckets. Remember that referencing this array starts at 0!
ReDim num_pts_per_hour(TotalHours, 1)

'*****Both of these commands will decrease runtime but don't forget to turn them back on later
'*****This turns off screen updating
Application.ScreenUpdating = False
'*****This sets all calculations to manual which prevents unecessary calculations
Application.Calculation = xlCalculationManual

'Clear previous simulation results and selects the simulation sheet
'*****Very important if you use cell ranges. You must be on the right tab
Worksheets("Optimization").Range("demand").ClearContents
Worksheets("Simulation").Select
```

```
'Loop for all replicataions
For i = 1 To Reps

    'Loop through all patients
    '*****Look below for A+ indenting. There is only A+ and F in this class for indenting
    For j = 1 To Patients

        'Turn on the status bar and have it display which replication is currently being simulated
        Application.DisplayStatusBar = True
        '*****Note how we have both text and variable in the status bar, and not a ridiculous amount of information. Use & to concatenate
        Application.StatusBar = "Replication " & i & " of " & Reps

        'If statement to determine appointment type and generate a LOS with the appropriate parameters
        '*****Note the offset function, and how it uses parentheses to define the row and column
        If Worksheets("Simulation").Range("TypeTop").Offset((j - 1), 0) = "Long" Then
            'For patients with long appointments, generate LOS from normal distribution with mean=2.75, st. dev=0.75 and round to nearest 15 minutes for simplicity
            '*****This code would get a speeding ticket. Note how we have hard-coded values in the equation. Improve this by defining these variables above and referencing them
            Worksheets("Simulation").Range("LengthTop").Offset((j - 1), 0) = Application.Round((Application.NormInv(Rnd(), 2.75, 0.75)) / 0.25, 0) * 0.25

        Else
            'For patients with short appointments, generate LOS from normal distribution with mean=1.75, st. dev=0.25 and round to nearest 15 minutes for simplicity
            '*****Another speeding ticket. What would happen if you had 100 LOS distributions? Always think about scaling
            Worksheets("Simulation").Range("LengthTop").Offset((j - 1), 0) = Application.Round((Application.NormInv(Rnd(), 1.75, 0.25)) / 0.25, 0) * 0.25

        '*****Note how you have to end an if statement. Not all languages have this!
        End If
    '*****End of patients for loop
    Next

    'Since automatic sheet calculations were turned off, make sure all in-sheet formulas calculate before saving output
    '*****If you're going to add code to improve efficiency, make sure you do it correctly. You don't want to debug this sort of thing
    '*****Also notice that we are still on manual calculation, we have simply told the sheet to calculate
    Worksheets("Simulation").Calculate

    'Loop through array and save output
    'For each hour of operation we sum the # of patients at the clinic across all reps and will later divide by total reps to get the average # of patients per hour
    For k = 1 To TotalHours
        '*****Would this formula break if a new user added a row or column to the spreadsheet? Could this be more dynamic and reliable?
        num_pts_per_hour(k, 1) = num_pts_per_hour(k, 1) + Cells(7 + k, 12).Value
    Next
Next

'After all replications are complete - cycle through array, divide each value by total reps, and write the result to the optimization sheet
'*****This calculates the average demand per hour. What if you wanted to get the distribution? Or Variance? Think about what data structures would allow for this reporting
For m = 1 To (TotalHours - 1)
    Worksheets("Optimization").Range("DemandTop").Offset((m - 1), 0) = (num_pts_per_hour(m, 1)) / Reps
Next

'Set decision variables for optimization to 0, may not be 0 from previous runs
'*****If you choose to expand the optimization model, you'll need to clear more of these and add them to the optimization solver
Worksheets("Optimization").Range("shift_6a").Value = 0
Worksheets("Optimization").Range("shift_10a").Value = 0
```

```
'Select the optimization sheet so that the user will see the results and can then optimize staffing
Worksheets("Optimization").Select

'Clear the status bar and turn automatic sheet calculations and screen updating back on
'*****Super important. All your codes ever will end with these 3 lines
Application.StatusBar = False
Application.Calculation = xlCalculationAutomatic
Application.ScreenUpdating = True

End Sub
```

```
'*****************************************************
'Nurse Demand Optimization Model
'Developed by Sam Davis and Kendall Sanderson
'Contact Information: davis.sam@husky.neu.edu
'Last Modified 6/9/2015
'*****************************************************
Sub Optimize_Staff()

'Go to optimization worksheet
Worksheets("Optimization").Select

'Call solver
'*****Note how we are not updating solver, just call it. This would be much better if we updated solver in order to avoid user error
SolverSolve True

End Sub
```

Guide to Good VBA Coding

- Always comment your code always
- Indent, give spaces, separate functions
- When in doubt, use your friend Google
- When recording, make sure to clean up
- Never reference static cells
- Use but don’t abuse your status bar
- Use friendly variable names (i.e. not “thisvar”)
- Always comment your code always

Get Your Hands Dirty

- Essential Next Steps
  - Fix cell references that are not named ranges
  - Create user-defined LOS parameter dashboard
- Optional Next Steps
  - Calculate standard deviation of demand by hour
  - Scale # of patient types (“n” types instead of 2)
  - Record and output demand distribution by hour
  - Optimize nurse shifts (i.e. add more times and types)
  - Optimize patient schedule? (hint: Tabu search)
  - Incorporate no-shows/lateness/earliness