INTRO TO VBA IN EXCEL WORKSHOP

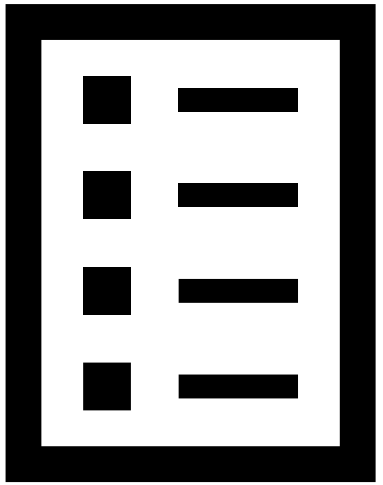
AGENDA

01
Workshop Case Study

02
Getting Started with VBA

03
Subroutines (Macros)

04
Writing VBA Code



01. Workshop Case Study

BACKGROUND

- **File link:** http://tinyurl.com/bas-vba
- You are an actuary for Bruin Health, a health insurance company based in Los Angeles insuring personal health claims across Southern California.
- Prior to your team's analysis of coverage and reserves, you notice that the policy data is not formatted in the desired manner.
- Each membership is only listed once, regardless of how many times it has been renewed.

- Problem: You need to be able to identify the specific 1-year term you're looking at when conducting the analyses.

BACKGROUND (CONT.)

- Task: The actuarial department head wants you to update the way the company's data is stored.

- If a membership has been renewed twice (i.e. it was in force for 3 terms), there should be 3 rows, one for each term.
- The listed start date should be the start date of each term (i.e. a new row is created upon renewal).

OBJECTIVE

1. Write a VBA macro to accomplish this task with the provided short excerpt of data.
2. Then, once you've ensured the macro works correctly, run it on the full set of membership data, splitting the 13,041 membership plans into 39,215 rows.

02. Getting Started with VBA

WHAT IS VBA?

- VBA: Visual Basic with Applications
- Object-based language (similar to object-oriented languages)
- Used in Excel to simplify repetitive or complex tasks

ENABLING DEVELOPER TAB

- On the menu bar, click:
  - File
  - -> Options
  - -> Customize Ribbon
  - -> Developer

  - Excel
  - -> Preferences
  - -> Ribbon & Toolbar
  - -> Developer

THE VBA ENVIRONMENT

- Multiple ways to open the VBA Editor…
  1) Go to Developer tab → select Visual Basic
  2) Keyboard shortcut
     1) Windows: Alt + F11
     2) Mac: Opt + F11
  3) Right-click on a worksheet → select View Code

GETTING STARTED

- Module: window where you can write VBA code
- It's always a good idea to put "Option Explicit" in the declaration at the top of your module to force yourself to declare all your variables
  - You can either:
    - Manually type "Option Explicit", or
    - Go to Tools → Options → Check "Require Variable Declaration"

03. Subroutines (Macros)

VBA IN EXCEL

- Subroutine (Sub): a program that accomplish tasks that do not return values
- Function: a program that takes in inputs and returns some output
- VBA objects include: workbooks, worksheets, and ranges
  - i.e. Application.Workbooks("Book1.xlsm").Worksheets("Sheet1").Range("A1")
- Objects have properties and methods, the most important of which is: Range("A1").Value

SUBROUTINES (CONT.)

- Most of your VBA code will be in subroutines (aka. Macros), which can be used to perform calculations, change formatting, and copy and paste among other repetitive tasks
- We enclose our code in the following:

```
Sub MyRoutine()

End Sub
```

- In this case, we have named our subroutine “MyRoutine” (but we can name this whatever we want)

SUBROUTINES (CONT.)

- There are multiple ways to call a subroutine
  - Click “Macros” in the Developer tab
  - Add a button from the Developer tab

04. Writing VBA Code

VBA IN EXCEL

- 2 commonly-used methods in VBA: Select and Activate
- Select allows you to select one or more objects
  - Worksheets(“Sheet1”).Select
  - Range(“A1:A3, C1:C3, E1:E3”).Select
- Activate allows you to select one object and make it active. If you already have multiple objects selected, this allows you to select on object within them.
  - Range(“A1”).Activate
- Ex: You can select a worksheet where you want your code to run and then activate the first cell in that worksheet that you want to apply changes to

FUNCTIONS

- Syntax:

```
Function MyFunction(param1 As type, param2 As type,...) As type
    some statements
    MyFunction = value
End Function
```

- Ultimately, you assign a return value by setting the name of the function to some value
- These functions can be called from workbook cells, like how you use COUNTIF(), VLOOKUP(), and other functions

MACRO RECORDER

- This is the easiest way to “write” a macro
- Under the Developer tab, click “Record Macro”
- All your actions will be translated into code, which you can find under Modules in the VBA Editor
- You can look at and modify this code to suit your purposes
- Often slow and clunky but can be a very powerful tool if used right (e.g. no one remembers how to filter/sort data in VBA, but this tool helps!)

VARIABLES

- Variable types
  - Integer, Double, String, Boolean, Date, Currency, and more
- We use the keyword Dim to declare a variable
  - Dim myString As String
  - Dim dbl As Double
  - Dim num as Integer
  - Dim rng As Range
- We assign them with "="
  - myString = "Hello"
  - num = 5
- However, to assign a range, we use:
  - Set rng = Range("A1:B3")

VARIABLES (CONT.)

- If the data type is not specified, the variable will be declared as a variant
- A variant can contain any kind of data, and the data type can change at any point
- Try to avoid these if possible (they require a lot of memory)

ARRAYS

- Arrays are created with parentheses to indicate size:
  - Dim myArray(5) As Integer
- The size can be changed:
  - ReDim myArray(10)
- But this will delete the data. To preserve the data inside, use:
  - ReDim Preserve myArray(15)
- Individual elements can be accessed and modified with parentheses:
  - myArray(0) = 5
- There's a lot more to be learned with arrays, but you can look online for more details. We'll work primarily with Workbook objects instead

IF STATEMENTS

- Syntax:

```
If condition Then
        ifStatements
ElseIf elseifCondition Then
        elseifStatements
Else
        elseStatements
End If
```

- If condition is true, run ifStatements. If condition is false but elseifCondition is true, run elseifStatements. Otherwise, runs elseStatements
- There can be as many ElseIf's as needed

WHILE LOOPS

- Syntax:

```
While condition
        statements
Wend
```

- Runs statements until condition evaluates to FALSE
- Make sure that condition will not be TRUE forever, or you will have an endless while loop

FOR LOOPS

- Syntax:

```
For counter = start To end [Step increment]
        statements
Next [counter]
```

- Typically, counter is an Integer that we increment
  - "For i = 0 to 3" will run 4 times (i = 0, 1, 2, 3)
  - "For i = 2 to 7 Step 2" will run 3 times (i = 2, 4, 6)
  - "For i = 5 to 0 Step -3" will run 2 times (i = 5, 2)

FOR EACH LOOPS

- Syntax:

```
For Each cell In range
        statements
Next
```

- A quick and simple way to loop through all the cells in range
- cell and range must both be Range objects
- range should be initialized to some Range, cell need not be

MESSAGE BOXES

- Syntax:

  MsgBox("Prompt")

- You can replace "Prompt" with whatever you want to appear in the message box
- Can be very useful in debugging to display the value of variables

WORD OF CAUTION

- Once you run a macro, you cannot undo it!
- Make sure to save your workbook before running a macro to avoid losing your work in case the macro doesn't work as intended.

TAKEAWAYS SLIDE

KEY TAKEAWAYS

- VBA is a powerful programming language used in Microsoft Excel to perform automated tasks
- Enable macros and the Developer tab, and use online documentation and the macro recording functionality to expand your VBA knowledge

ANNOUNCEMENTS

- Introductory R Workshop on Thurs. 2/22

bruinactuaries@gmail.com

www.math.ucla.edu/actuary/

@bruinactuaries

BRUIN ACTUARIAL SOCIETY

Thank you

Any questions?