

# Programming IB – loops

Doing simple things many times fast

# Loops

- Loops are ways of telling the computer to repeat an operation until a condition is met
- The condition can be several different things:
  - A fixed number of repeats
  - A run through a list of arguments
  - A criteria that needs to be satisfied
- Loops begin and end with key words
- Two types we will learn:
  - Do While loops
  - For Next loops

# Do loops

- Do loops can have two different forms:
  - Do while  
...  
loop
  - Do  
...  
loop until
- The criterion can be tested before or after the loop instructions within the loop are executed
- Once the criterion is met the program leaves the loop and continues on to the next instruction



```
Sub MantelTest()  
|  
| MantelTest Macro  
| Conduct a Mantel test on the geographic and genetic distances.  
|  
| Keyboard Shortcut: Ctrl+Shift+M  
|  
Range("C1:D22").Select  
ActiveWorkbook.Worksheets("Sheet2").Sort.SortFields.Clear  
ActiveWorkbook.Worksheets("Sheet2").Sort.SortFields.Add Key:=Range("D2:D22") _  
    , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal  
With ActiveWorkbook.Worksheets("Sheet2").Sort  
    .SetRange Range("C1:D22")  
    .Header = xlYes  
    .MatchCase = False  
    .Orientation = xlTopToBottom  
    .SortMethod = xlPinYin  
    .Apply  
End With  
Range("C24").Select  
Selection.Copy  
Range("F2").Select  
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _  
    :=False, Transpose:=False  
End Sub
```

The code that Excel  
recorded for Mantel test

# The header

```
|  
| MantelTest Macro  
| Conduct a Mantel test on the geographic and genetic distances.  
|  
| Keyboard Shortcut: Ctrl+Shift+M  
|
```

*The apostrophes are **comment** characters. Anything after them is ignored by the interpreter.*

*Used to make notes about what the program does.*

*Comments are a Good Thing.*

# Subroutines are marked by Sub, End Sub

```
Sub MantelTest()
```

```
Instructions to execute...
```

```
End Sub
```

*At least one subroutine must be present for the macro to run.*



```
Sub MantelTest()
```

```
|
```

```
' Keyboard Shortcut: Ctrl+Shift+M
```

```
'
```

```
Range("C1:D22").Select
```

```
ActiveWorkbook.Worksheets("Sheet2").Sort.SortFields.Clear
```

```
ActiveWorkbook.Worksheets("Sheet2").Sort.SortFields.Add Key:=Range("D2:D22") _
```

```
, SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
```

```
With ActiveWorkbook.Worksheets("Sheet2").Sort
```

```
.SetRange Range("C1:D22")
```

```
.Header = xlYes
```

```
.MatchCase = False
```

```
.Orientation = xlTopToBottom
```

```
.SortMethod = xlPinYin
```

```
.Apply
```

```
End With
```

```
Range("C24").Select
```

```
Selection.Copy
```

```
Range("F2").Select
```

```
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
```

```
:=False, Transpose:=False
```

```
End Sub
```

*Select the range of data  
to sort (not needed)*

*Clear out any old sort  
keys in memory*

*Execute the sort*

*Identify the sort  
key to use, and the  
order (ascending)*

*Select and copy the  
Mantel statistic from C24*

*Paste the copied Mantel  
stat to cell F2*

The subroutine – the code  
that does the work

## Now, make it run repeatedly

- The rand() function selects a new set of random numbers each time the sheet recalculates
- Sorting recalculates the sheet
- As soon as the numbers are sorted, there are new random numbers for sorting again
- All that's needed is to tell the macro to repeat the operation until the numbers are in order
- This is done with a **loop**



# Do loops

- Do loops can have two different forms:
  - Do while (*condition*)    ← *Test condition at the beginning will not execute at all unless the condition is met*  
    ...  
    loop
  - Do  
    ...  
    loop until (*condition*)    ← *Test condition at the end will execute at least once*
- The condition is a logical test
  - Comparison between cells
  - Comparison of a cell to a number
  - Value of a cell that can either be TRUE or FALSE
- In both cases the program leaves the loop and continues on to the next instruction once the condition is met

# For...next loops

- Useful for executing an operation on a defined list of inputs, or a fixed number of repetitions
- Syntax is:

For i in 1 to 10

*Things for the loop to do*

Next i

- i is a variable – value changes each time through the loop
- How many times will this loop execute? Plickers...
- We'll use these a lot for randomization testing and bootstrapping



# Infinite loops

- Avoid these
- If you use a loop in which the ending condition cannot ever be met, it will run forever
- If this happens, VB allows you to interrupt a running macro
- Childishly easy to create!



# An infinite Do...loop

- The following is an infinite Do loop

```
Range("A2").Value = False
```

```
Do while Range("A2") = False
```

```
Loop
```

- A2 is never changed, so it can never become True
- This will execute forever, until you stop it or the computer dies
- Stop a program with the Escape key (Esc)

# Application: randomization testing

- A “nonparametric” approach to analyzing data
- Generally used when the usual parametric approaches (t-tests, ANOVA, regression, etc.) aren't appropriate because of violated assumptions
- A p-value is calculated from a sampling distribution derived by randomly shuffling the data

# Modify the macro to loop

- Two changes:
  - Add a “For...next” loop
  - Each time through the macro, need to store the measure of association
- Currently, copying/pasting measure of association to F2 – looping will make us replace this number each time through
- But, the counter (“i”) increases by 1 each iteration – if we paste to cell F(i+1), we will write to a new row each time



Add a  
loop,  
record  
each result

```
Sub MantelTest()  
'  
' MantelTest Macro  
' Conduct a Mantel test on the geographic and genetic distances.  
'  
' Keyboard Shortcut: Ctrl+Shift+M  
'  
  
For i = 1 To 1000  
    Range("C1:D22").Select  
    ActiveWorkbook.Worksheets("Sheet2").Sort.SortFields.Clear  
    ActiveWorkbook.Worksheets("Sheet2").Sort.SortFields.Add Key:=Range("D2:D22") _  
        , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal  
    With ActiveWorkbook.Worksheets("Sheet2").Sort  
        .SetRange Range("C1:D22")  
        .Header = xlYes  
        .MatchCase = False  
        .Orientation = xlTopToBottom  
        .SortMethod = xlPinYin  
        .Apply  
    End With  
    Range("C24").Select  
    Selection.Copy  
    Range("F" & i + 1).Select  
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _  
        :=False, Transpose:=False  
Next i  
End Sub
```

Select and copy the  
Mantel statistic

Select a location to record it

Paste-special the  
value

F	
Sums of products	
680.4899364	
691.633857	
694.7870141	
695.480302	
703.5632657	
707.3579128	
708.6603744	
708.7029923	
709.2008845	
709.2259746	
714.464126	
716.5807235	
719.5179922	
903.9585579	
908.6388034	
909.1593751	
909.5246373	
909.8134634	
910.7131166	
912.239676	
913.8250635	
914.4788926	
918.1006404	
918.1550153	
921.2731812	
922.473072	
927.1757834	
932.7133386	
937.300003	

# Run the macro, sort the results

*How many exceeded the observed?*

*We're only interested in whether there was a greater association than observed, so can just look at values bigger than observed (one-tailed test)*



	A	B	C	D	E	F	G
27	Observed sum of products		886.388625			729.5221284	
28						729.5840747	
358						882.8188554	
359						883.0791281	
360						885.5101171	
361						886.5968693	
362						886.6395865	
363						886.9566006	
364						887.5672563	
365						887.6931162	
366						888.1998953	
367						888.759341	
368						889.0449391	
369						891.3669924	
370						891.7571737	
371						892.8692841	
372						893.0389275	
373						894.7192509	
374						895.6236515	
375						895.8392815	
376						896.0025476	
377						896.2078633	
378						896.4582357	
379						897.1162263	
380						899.7294216	
381						900.0607245	
382						900.2007867	
383						901.6036995	
384						903.2106488	
385						903.6495732	
386						903.9585579	
387						908.6388034	
388						909.1593751	
389						909.5246373	
390						909.8134634	
391						910.7131166	
392						912.239676	
393						913.8250635	
394						914.4788926	
395						918.1006404	
396						918.1550153	
397						921.2731812	
398						922.473072	
399						927.1757834	
000						932.7133386	
001						937.300003	
002							

# Calculate p

$$p = \frac{\text{Number exceeding observed} + 1}{\text{Number of iterations} + 1}$$

$$p = \frac{41 + 1}{1000 + 1} = 0.042$$

*Reject the null – there is a (weak but) significant association between genetic distance and geographic distance*