Excel Analytics and Programming

Fall 2012 Workshop

George Zhao
gz2165@columbia.edu

Goals of the Workshop

- Learn Excel tools by utilizing them in various cases
  - Tools and materials covered here are merely a sample of Excel functionality
- Understand the logic and syntax behind Visual Basic programming, which interacts with the Excel interface
  - No programming background required
- Create dynamic algorithms to approach cases
  - When data is changed, but retains its original format, the algorithm should be able to automatically handle the transition appropriately

Workshop Structure

- Instead of providing function and programming syntax to memorize, this workshop emphasizes case studies, through which the skills are utilized
  - Cases: applicable situational tasks
  - Tutorials: supplemental teaching material to understand foundational materials
- Best recommended to follow along and do the exercises with accompanying spreadsheets

Workshop Resources

http://www.columbia.edu/~gz2165/excel.html

Lesson materials:

- Learning Slides[.pdf]
- Exercises - Blank [.xlsx]
- Exercises - Filled [.xlsm]

Contents Overview

- Case 1: Multiplication Table
- Case 2: Percentile Calculations
- Tutorial 1: Variables and Arrays
- Case 3: Hello World
- Tutorial 2: Functions
- Tutorial 3: Loops and Decisions
- Case 4: Gradebook Tallying
- Case 5: Loop through Data
- Tutorial 4: Recording Macro
- Case 6: Select, Pull, Display
- Tutorial 5: Userform
- Case 7: Subway Data All-Around Analysis

Case 1: Multiplication Table

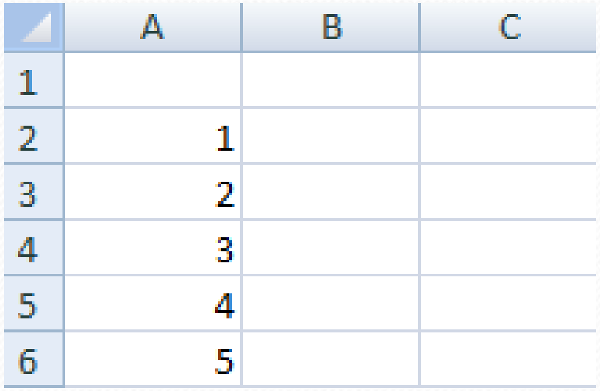
Multiplication Table

- Task: If given the following on an Excel worksheet, how do we fill this up?

	1	2	3	4	5
1					
2					
3					
4					
5					

First Task: Building the Table

- Suppose we are only given one side of the table initially:



	A	B	C
1			
2	1		
3	2		
4	3		
5	4		
6	5		

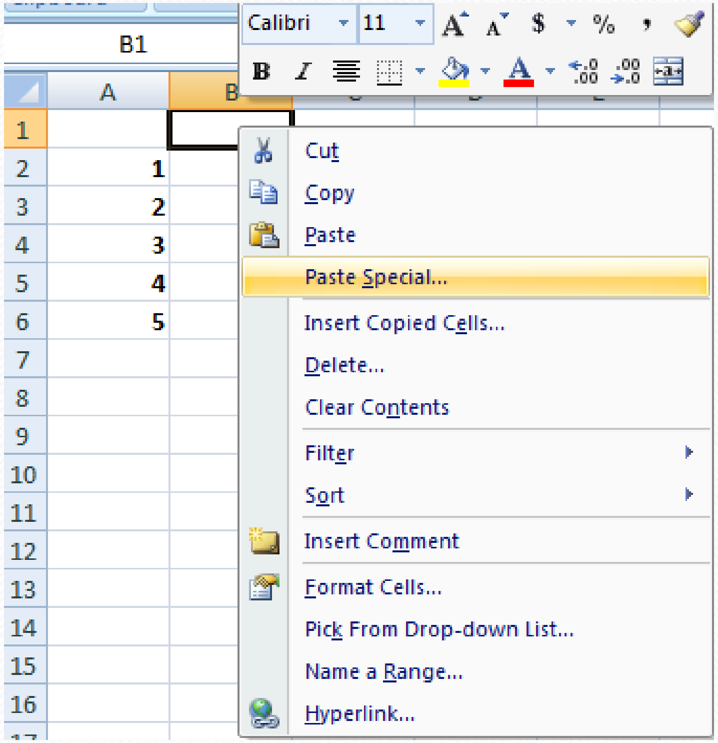
- We need 1, 2, 3, 4, 5 to be filled up on the top row, beginning in cell B1, going rightward
- We also want both sets of numbers to be bolded

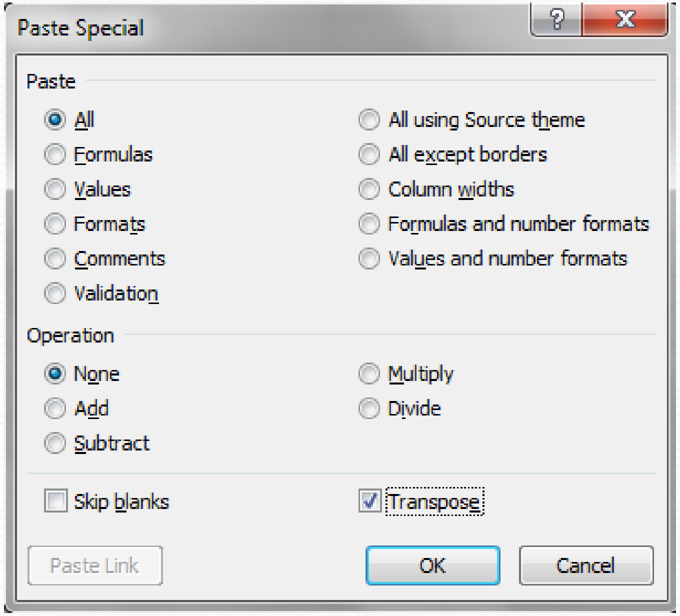
Shift + Control + Arrow

- Begin by selecting cell B2
- Shift + Control + Down arrow to select all elements until an empty cell (or the end of capacity limit of the worksheet) is reached
  - Shift + Control + (Up / Down / Left / Right) arrow all work similarly
- Control + B for bold
- Control + C for copy

Paste Special with Shortcut

- Use arrow to move to cell B1
- Can Right Click > Paste Special
- Or simply Alt + E + S
- Select Transpose (E)




Paste Special

Paste

● All
○ Formulas
○ Values
○ Formats
○ Comments
○ Validation
○ All using Source theme
○ All except borders
○ Column widths
○ Formulas and number formats
○ Values and number formats

Operation

● None
○ Add
○ Subtract
○ Multiply
○ Divide

□ Skip blanks
☑ Transpose

Where to Start

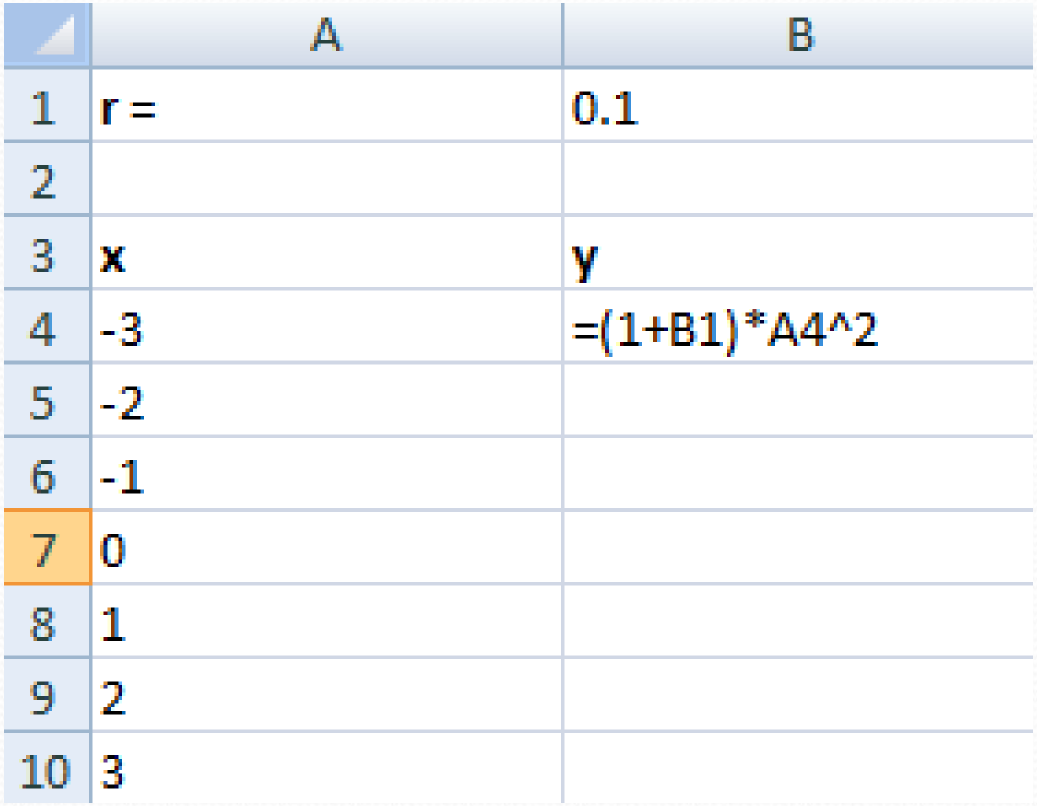
- Use fixed reference cells

	1	2	3	4	5
1					
2					
3					
4					
5					

1-Dimensional Fixed Reference

- Example: y = (1+r)*x^2, given a fixed value r

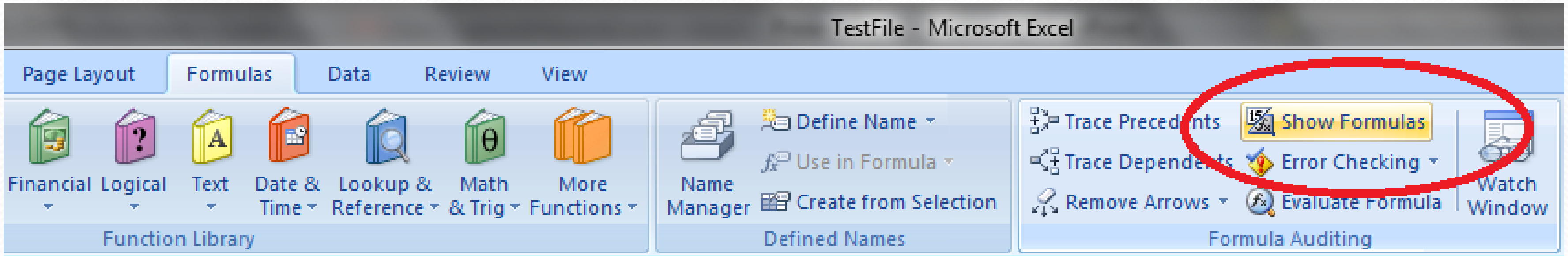
r =	0.1
x	**y**
-3	9.9
-2	
-1	
0	
1	
2	
3	



	A	B
1	r =	0.1
2		
3	**x**	**y**
4	-3	=(1+B1)*A4^2
5	-2	
6	-1	
7	0	
8	1	
9	2	
10	3	

Show Formulas

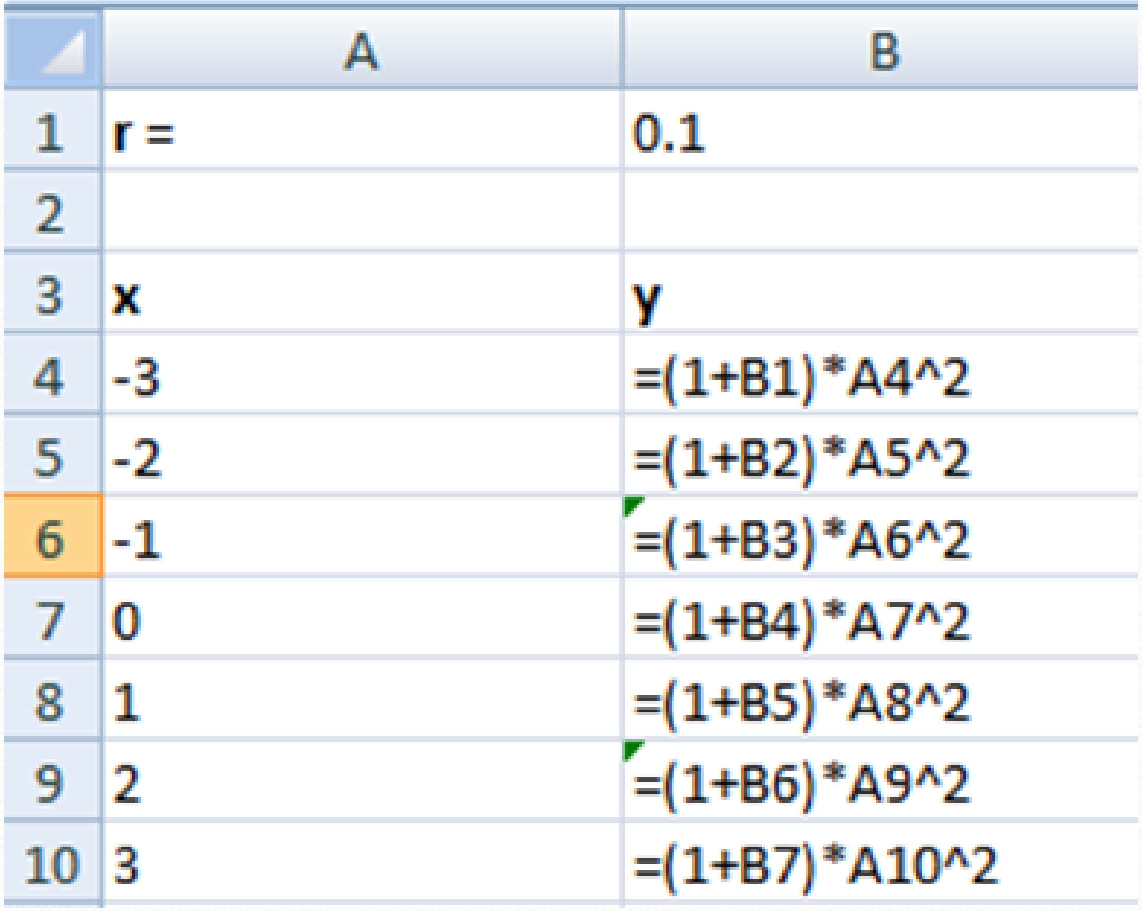
- Formulas > Show Formulas
- Toggle on and off between showing / not showing



Error: No Fixing Reference

- Error

r =	0.1
x	y
-3	9.9
-2	4
-1	#VALUE!
0	0
1	5
2	#VALUE!
3	9

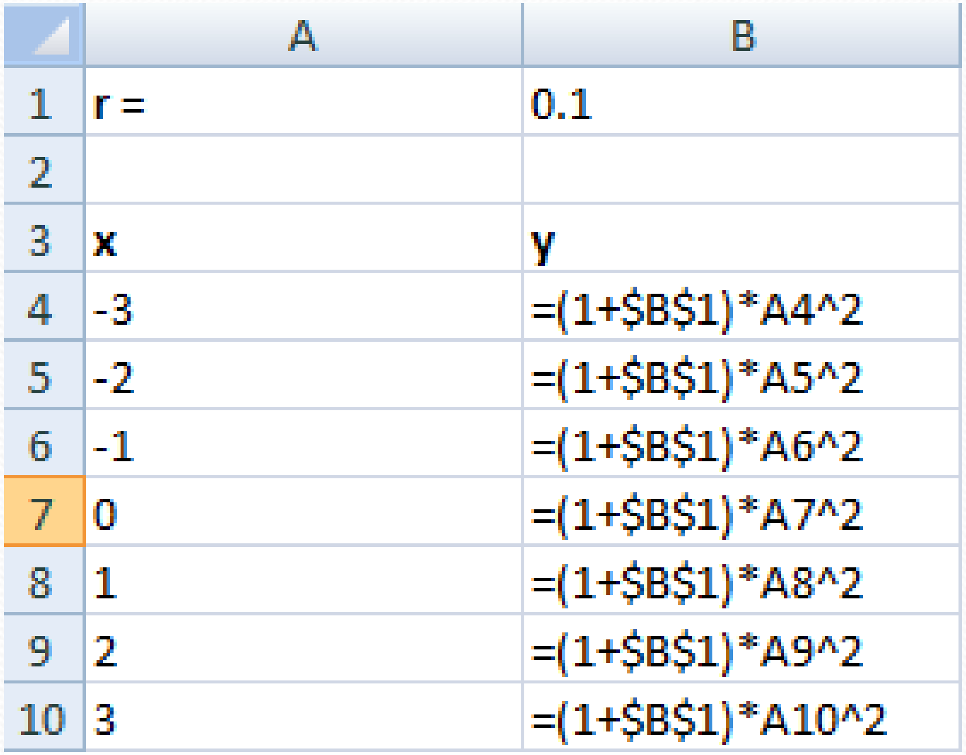


	A	B
1	r =	0.1
2		
3	x	y
4	-3	=(1+B1)*A4^2
5	-2	=(1+B2)*A5^2
6	-1	=(1+B3)*A6^2
7	0	=(1+B4)*A7^2
8	1	=(1+B5)*A8^2
9	2	=(1+B6)*A9^2
10	3	=(1+B7)*A10^2

1-Dimensional Fixed Reference

- Fix cell reference in cells A1, B2, BB32: $A$1, $B$2, $BB$32

r =		0.1
x	**y**	
	-3	9.9
	-2	4.4
	-1	1.1
	0	0
	1	1.1
	2	4.4
	3	9.9



	A	B
1	r =	0.1
2		
3	**x**	**y**
4	-3	=(1+$B$1)*A4^2
5	-2	=(1+$B$1)*A5^2
6	-1	=(1+$B$1)*A6^2
7	0	=(1+$B$1)*A7^2
8	1	=(1+$B$1)*A8^2
9	2	=(1+$B$1)*A9^2
10	3	=(1+$B$1)*A10^2

Deeper Look into Fixing Reference

- A1: not fixing column A nor row 1
- $A$1: fixing column A and row 1
- $A1: fixing ONLY column A, not row 1
- A$1: fixing ONLY row 1, not column A

- Whatever (row and/or column) right after the $ sign is fixed
- If fixing BOTH row and column, press F4 while cursor is over the reference in the formula editing

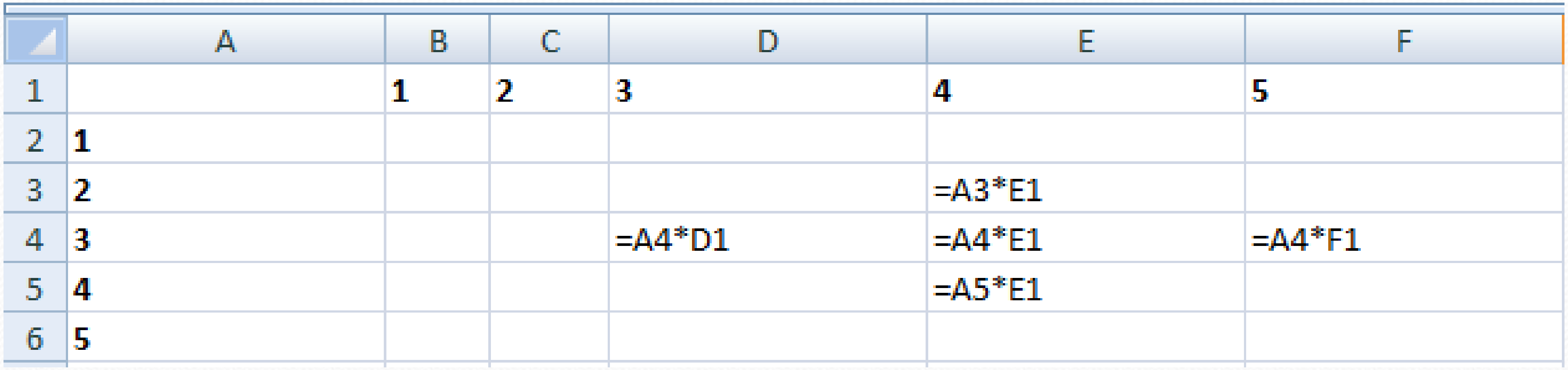
Algorithm: Multiplication Table

- Multiply the row index by the column index
- Do this for each cell in the row, and then for each row

	1	2	3	4	5
1					
2					
3				12	
4					
5					

Example: Multiplication Table

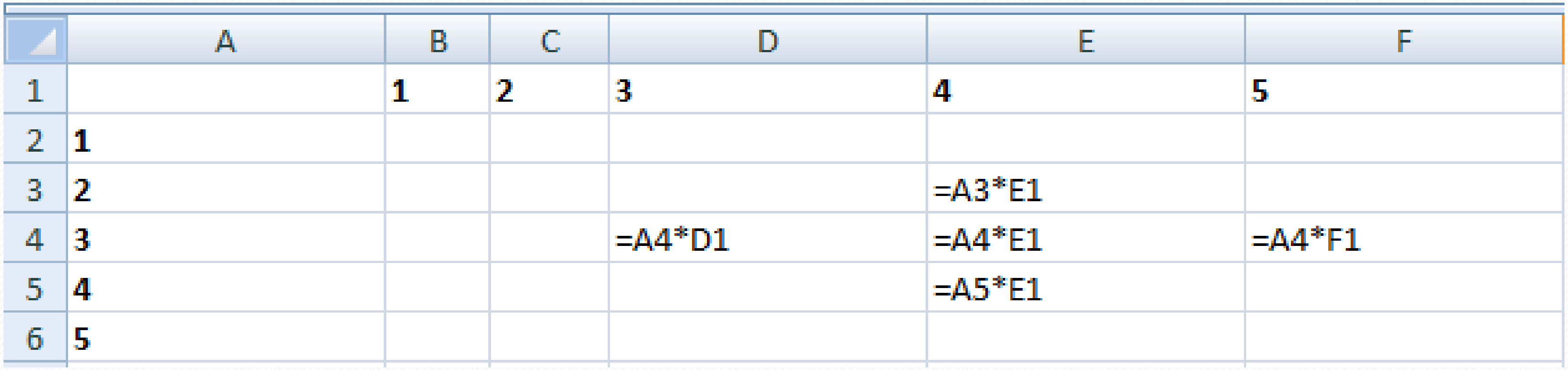
- Focus again on cell E4: (12 = 3 x 4)
  - All entries on row 4: product of 3 (A4) and ___
  - All entries on column E: product of 4 (E1) and ___



	A	B	C	D	E	F
1		1	2	3	4	5
2	1					
3	2				=A3*E1	
4	3			=A4*D1	=A4*E1	=A4*F1
5	4				=A5*E1	
6	5					

Think About It

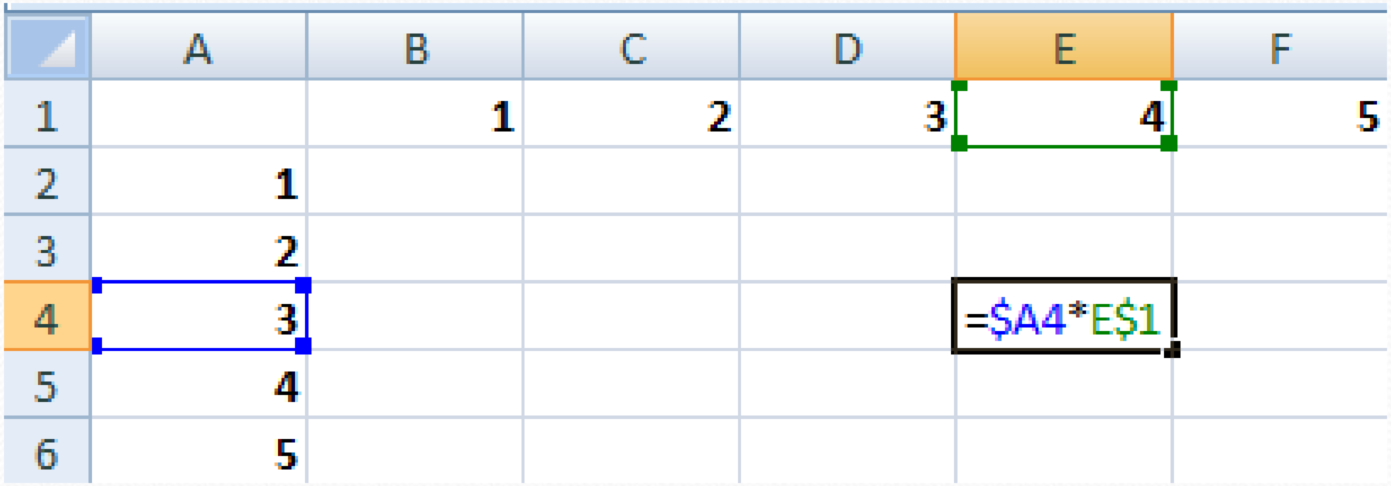
- Focus on any single row:
  - We are traversing through various columns, but want to fix the first term (A4), so fix the column letter (A)
- Focus on any single column:
  - We are traversing through various rows, but want to fix the second term (E1), so fix the row number (1)



	A	B	C	D	E	F
1		1	2	3	4	5
2	1					
3	2				=A3*E1	
4	3			=A4*D1	=A4*E1	=A4*F1
5	4				=A5*E1	
6	5					

Result of Fixing

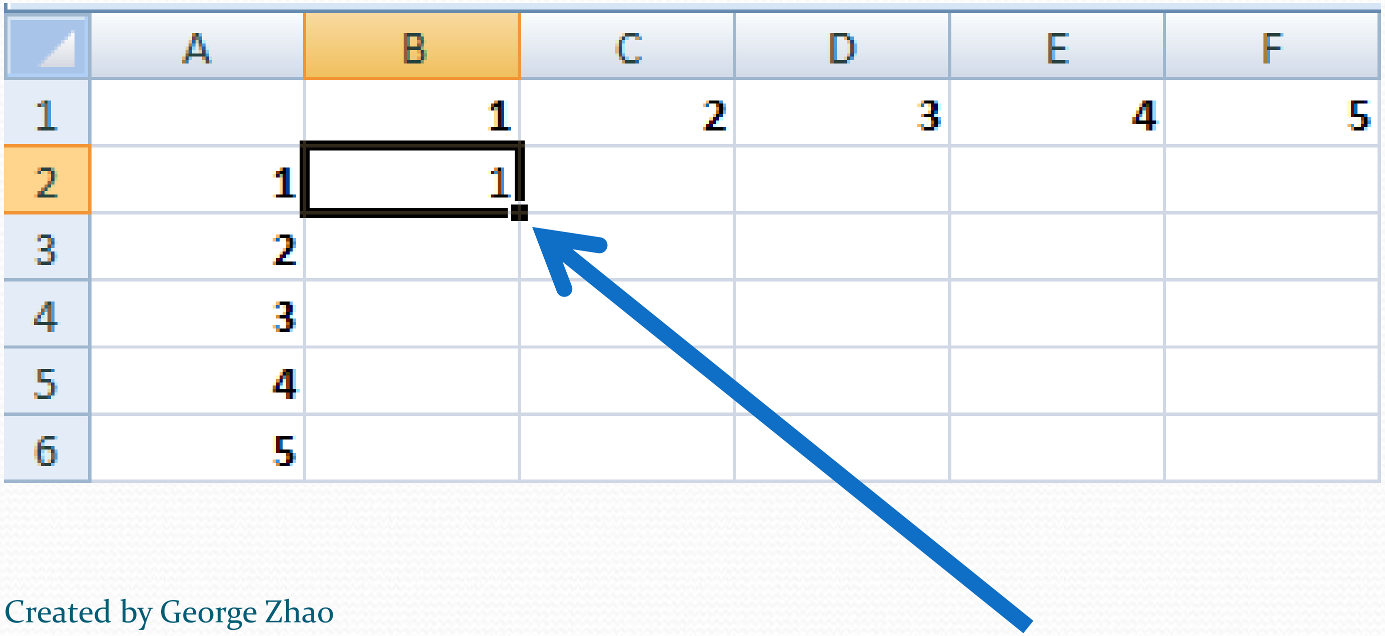
- Fix the column (A) from the first reference, and the row (1) from the second reference
- F2 to illustrate the formula and the references (colored)



	A	B	C	D	E	F
1		1	2	3	4	5
2	1					
3	2					
4	3				=$A4*E$1	
5	4					
6	5					

Shortcuts to Paste Formula

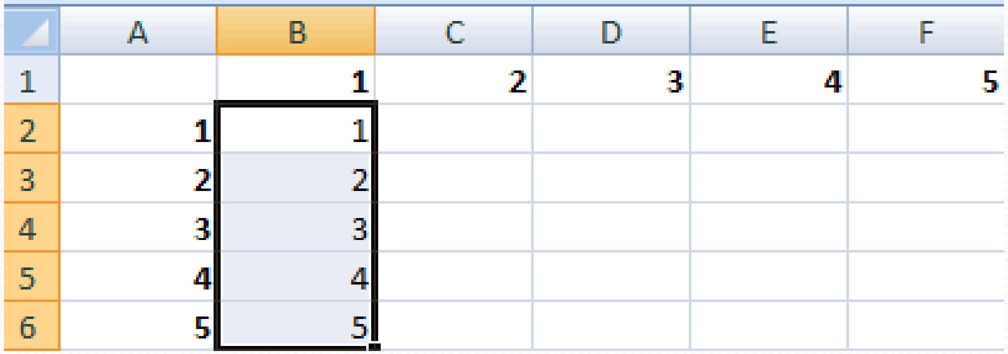
- Double click on the square at the lower-right corner of the cell
- This pastes the formula down the column, avoids the effort of dragging the formula down across rows



	A	B	C	D	E	F
1		1	2	3	4	5
2	1	1				
3	2					
4	3					
5	4					
6	5					

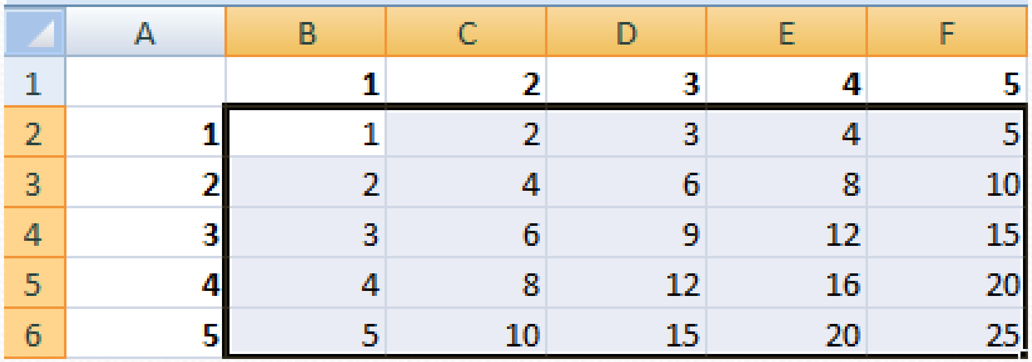
Shortcuts to Paste Formula

- Downside: won’t do the same horizontally across columns



	A	B	C	D	E	F
1		1	2	3	4	5
2	1	1				
3	2	2				
4	3	3				
5	4	4				
6	5	5				

- Have to manually drag it across the columns



	A	B	C	D	E	F
1		1	2	3	4	5
2	1	1	2	3	4	5
3	2	2	4	6	8	10
4	3	3	6	9	12	15
5	4	4	8	12	16	20
6	5	5	10	15	20	25

Performance Evaluation

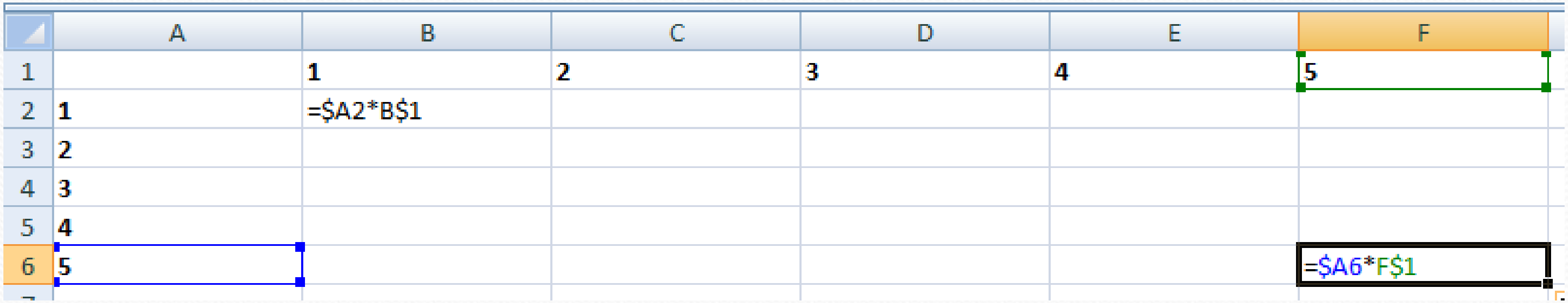
- How would performance have been if we were dealing with a 50 x 50 table instead?
  - Shift + Control + Down, copy, paste special (transpose) occurs constant time
  - Double clicking to paste to formulas down occurs constant time
  - Manually dragging the formulas across columns depends linearly on number of columns

Alt + Enter Pasting Method

- Here's the procedure of the method:
  - Paste the formula on one corner
  - Paste the formula on the other corner
  - Select all of the cells in-between to apply the same formula, using Shift + Control + arrow to get all of the wanted data
  - Click Alt + Enter
- Much more useful when dealing with 1-dimensional list than 2-dimensional table

Alt + Enter Illustrated

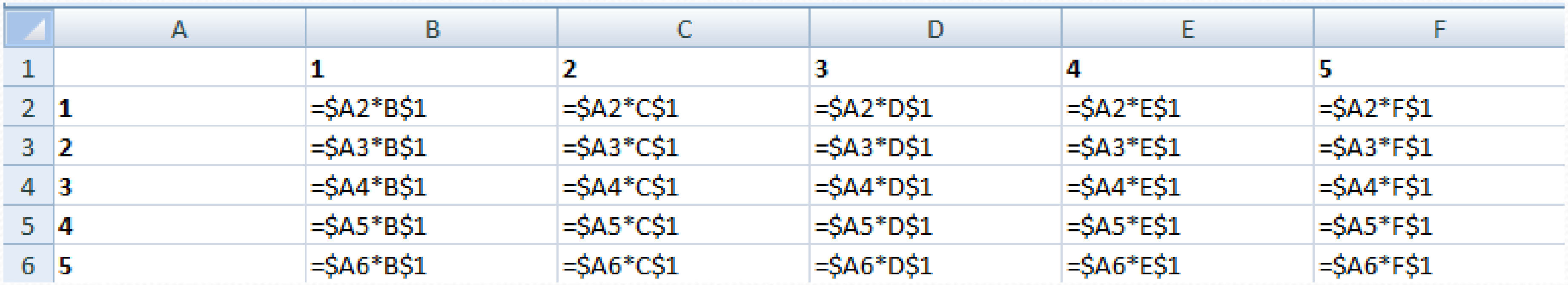
- Copy the formula in cell B2, paste into F6, select everything in-between, and hit Alt + Enter
- Difficult to capture the desired region efficiently with the Shift + Control + arrow method



	A	B	C	D	E	F
1		1	2	3	4	5
2	1	=$A2*B$1				
3	2					
4	3					
5	4					
6	5					=$A6*F$1

Quick Way to Paste Formula

	1	2	3	4	5
1	1	2	3	4	5
2	2	4	6	8	10
3	3	6	9	12	15
4	4	8	12	16	20
5	5	10	15	20	25



	A	B	C	D	E	F
1		**1**	**2**	**3**	**4**	**5**
2	**1**	=$A2*B$1	=$A2*C$1	=$A2*D$1	=$A2*E$1	=$A2*F$1
3	**2**	=$A3*B$1	=$A3*C$1	=$A3*D$1	=$A3*E$1	=$A3*F$1
4	**3**	=$A4*B$1	=$A4*C$1	=$A4*D$1	=$A4*E$1	=$A4*F$1
5	**4**	=$A5*B$1	=$A5*C$1	=$A5*D$1	=$A5*E$1	=$A5*F$1
6	**5**	=$A6*B$1	=$A6*C$1	=$A6*D$1	=$A6*E$1	=$A6*F$1

Remarks

- When fixing cell reference, think:
  - Are we fixing the column?
  - Are we fixing the row?
  - Both?
  - Neither?
- Alt + Enter way to paste formulas is more useful in 1-dimensional situations

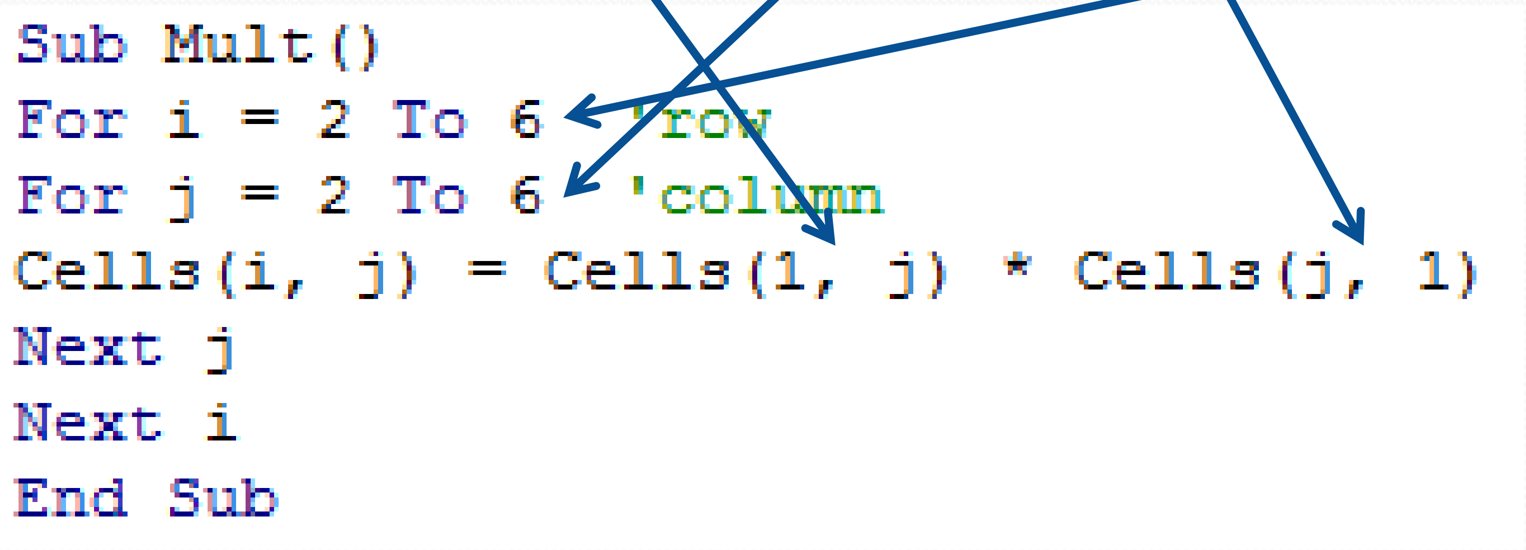
Pseudocode: The Approach

- Multiply the row index by the column index
- Do this for each cell in the row, and then for each row

	1	2	3	4	5
1					
2					
3				12	
4					
5					

Actual Code Juxtaposed

- Multiply the row index by the column index
- Do this for each cell in the row, and then for each row

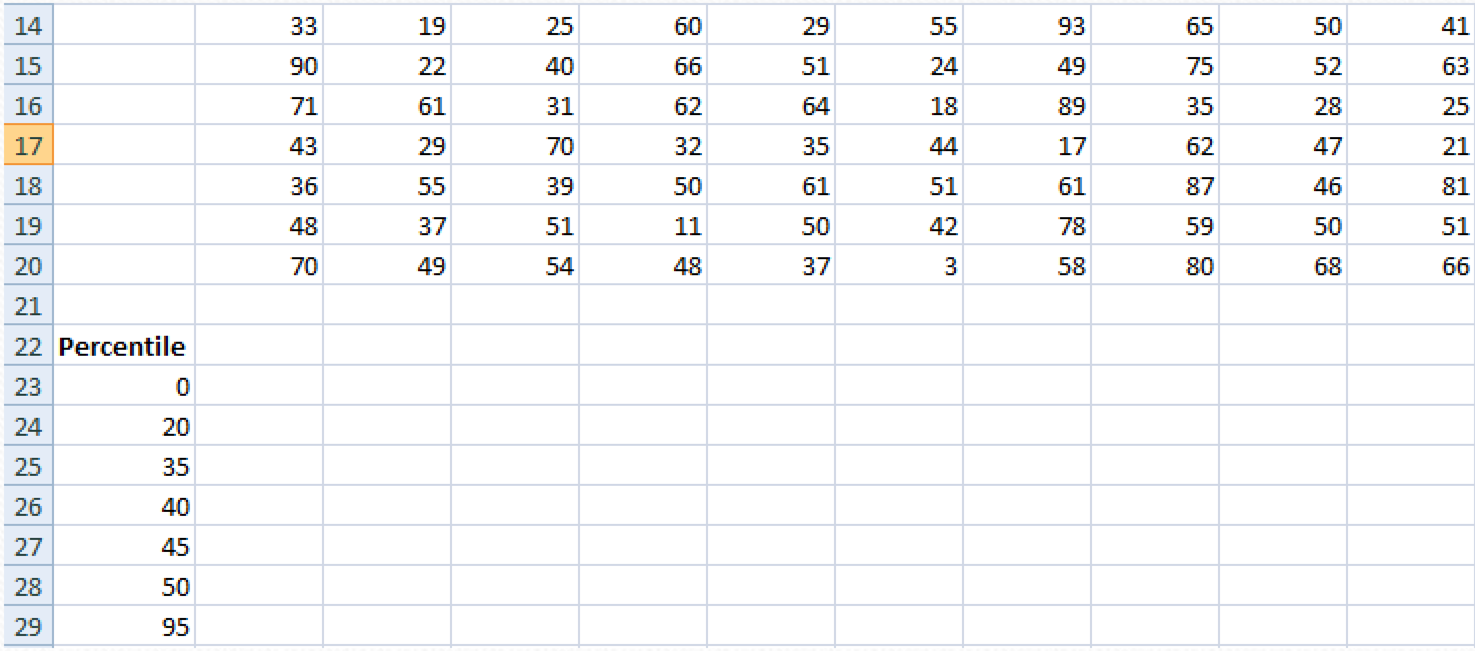


```
Sub Mult()
For i = 2 To 6 'row
For j = 2 To 6 'column
Cells(i, j) = Cells(1, j) * Cells(j, 1)
Next j
Next i
End Sub
```

Case 2: Percentile Calculations

Task 1

- We are given 20 x 10 matrix of all random numbers
- Upon supplying various integer values between 0 and 100, denoted n, give the n-th percentile of each column



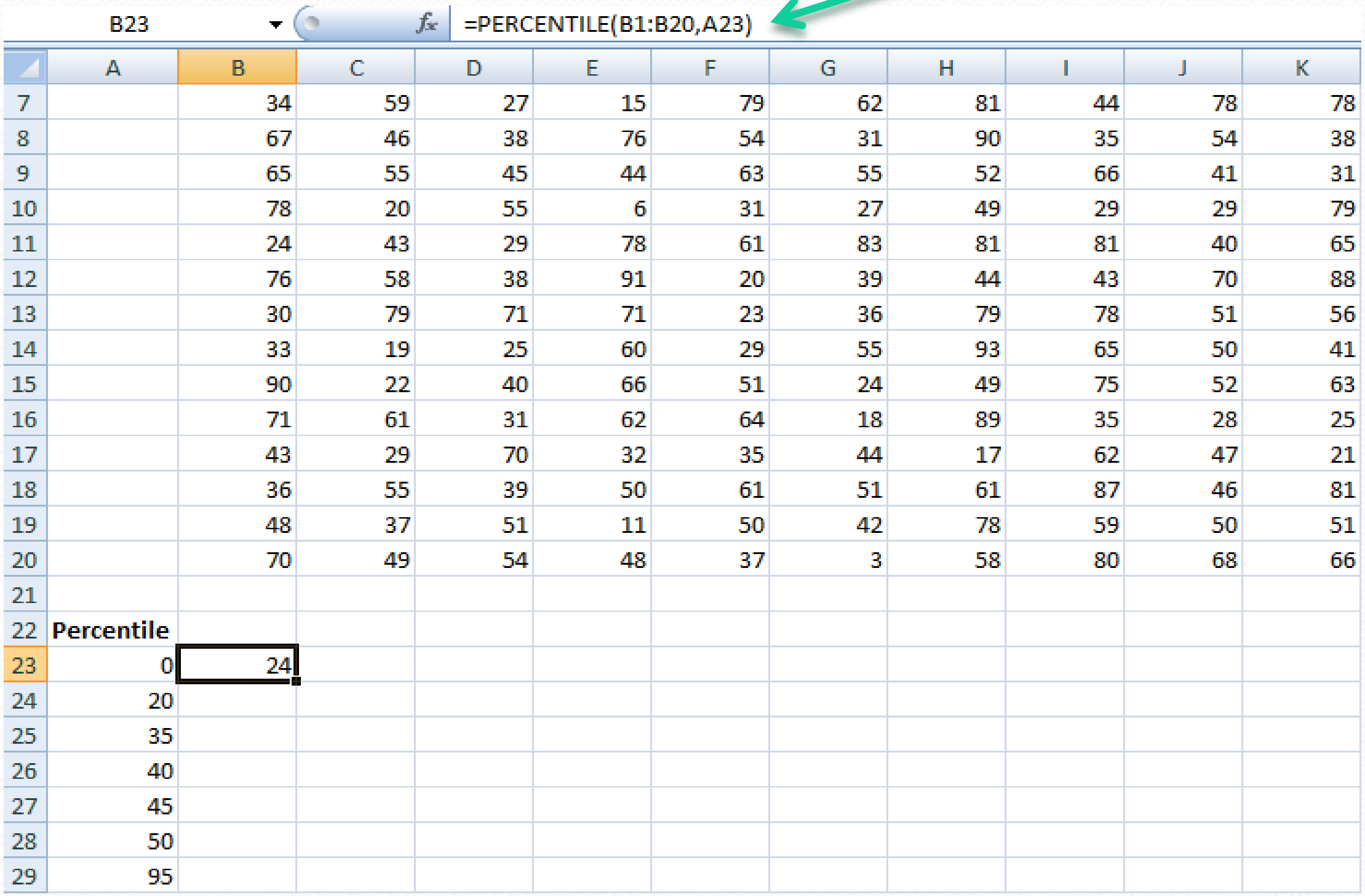
14		33	19	25	60	29	55	93	65	50	41
15		90	22	40	66	51	24	49	75	52	63
16		71	61	31	62	64	18	89	35	28	25
17		43	29	70	32	35	44	17	62	47	21
18		36	55	39	50	61	51	61	87	46	81
19		48	37	51	11	50	42	78	59	50	51
20		70	49	54	48	37	3	58	80	68	66
21											
22	**Percentile**										
23	0										
24	20										
25	35										
26	40										
27	45										
28	50										
29	95										

=PERCENTILE()

- =PERCENTILE([array], k)
- Let k be within [0, 1]

- =PERCENTILE(A1:A10, .75) gives the 75th percentile value of the data from A1 to A10
- =PERCENTILE(B1:B10, .05) gives the 5th percentile value of the data from B1 to B10

Right Formula?

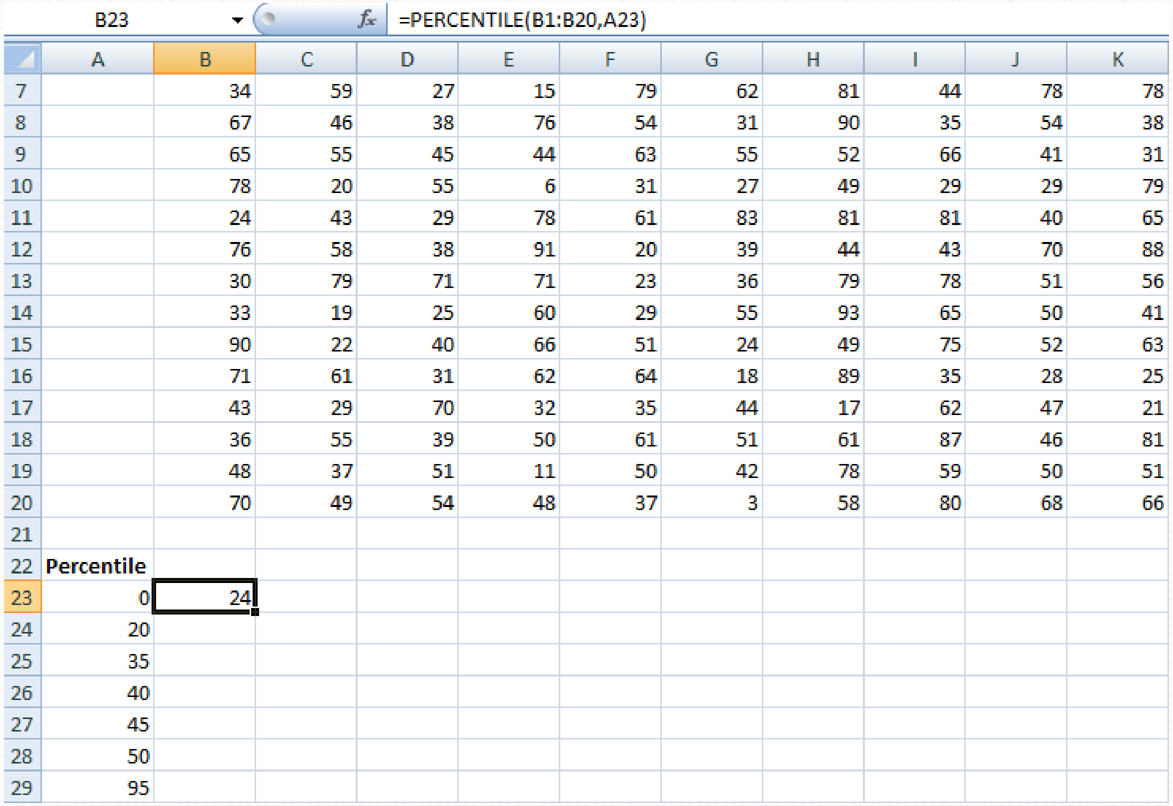


B23 =PERCENTILE(B1:B20,A23)

	A	B	C	D	E	F	G	H	I	J	K
7		34	59	27	15	79	62	81	44	78	78
8		67	46	38	76	54	31	90	35	54	38
9		65	55	45	44	63	55	52	66	41	31
10		78	20	55	6	31	27	49	29	29	79
11		24	43	29	78	61	83	81	81	40	65
12		76	58	38	91	20	39	44	43	70	88
13		30	79	71	71	23	36	79	78	51	56
14		33	19	25	60	29	55	93	65	50	41
15		90	22	40	66	51	24	49	75	52	63
16		71	61	31	62	64	18	89	35	28	25
17		43	29	70	32	35	44	17	62	47	21
18		36	55	39	50	61	51	61	87	46	81
19		48	37	51	11	50	42	78	59	50	51
20		70	49	54	48	37	3	58	80	68	66
21											
22	**Percentile**										
23	0	24									
24	20										
25	35										
26	40										
27	45										
28	50										
29	95										

Need to Fix Reference

- B1:B20 – fix the row? Column? Both? Neither?
- A23/100 – fix the row? Column? Both? Neither?



B23 =PERCENTILE(B1:B20,A23)

	A	B	C	D	E	F	G	H	I	J	K
7		34	59	27	15	79	62	81	44	78	78
8		67	46	38	76	54	31	90	35	54	38
9		65	55	45	44	63	55	52	66	41	31
10		78	20	55	6	31	27	49	29	29	79
11		24	43	29	78	61	83	81	81	40	65
12		76	58	38	91	20	39	44	43	70	88
13		30	79	71	71	23	36	79	78	51	56
14		33	19	25	60	29	55	93	65	50	41
15		90	22	40	66	51	24	49	75	52	63
16		71	61	31	62	64	18	89	35	28	25
17		43	29	70	32	35	44	17	62	47	21
18		36	55	39	50	61	51	61	87	46	81
19		48	37	51	11	50	42	78	59	50	51
20		70	49	54	48	37	3	58	80	68	66
21											
22	**Percentile**										
23	0	24									
24	20										
25	35										
26	40										
27	45										
28	50										
29	95										

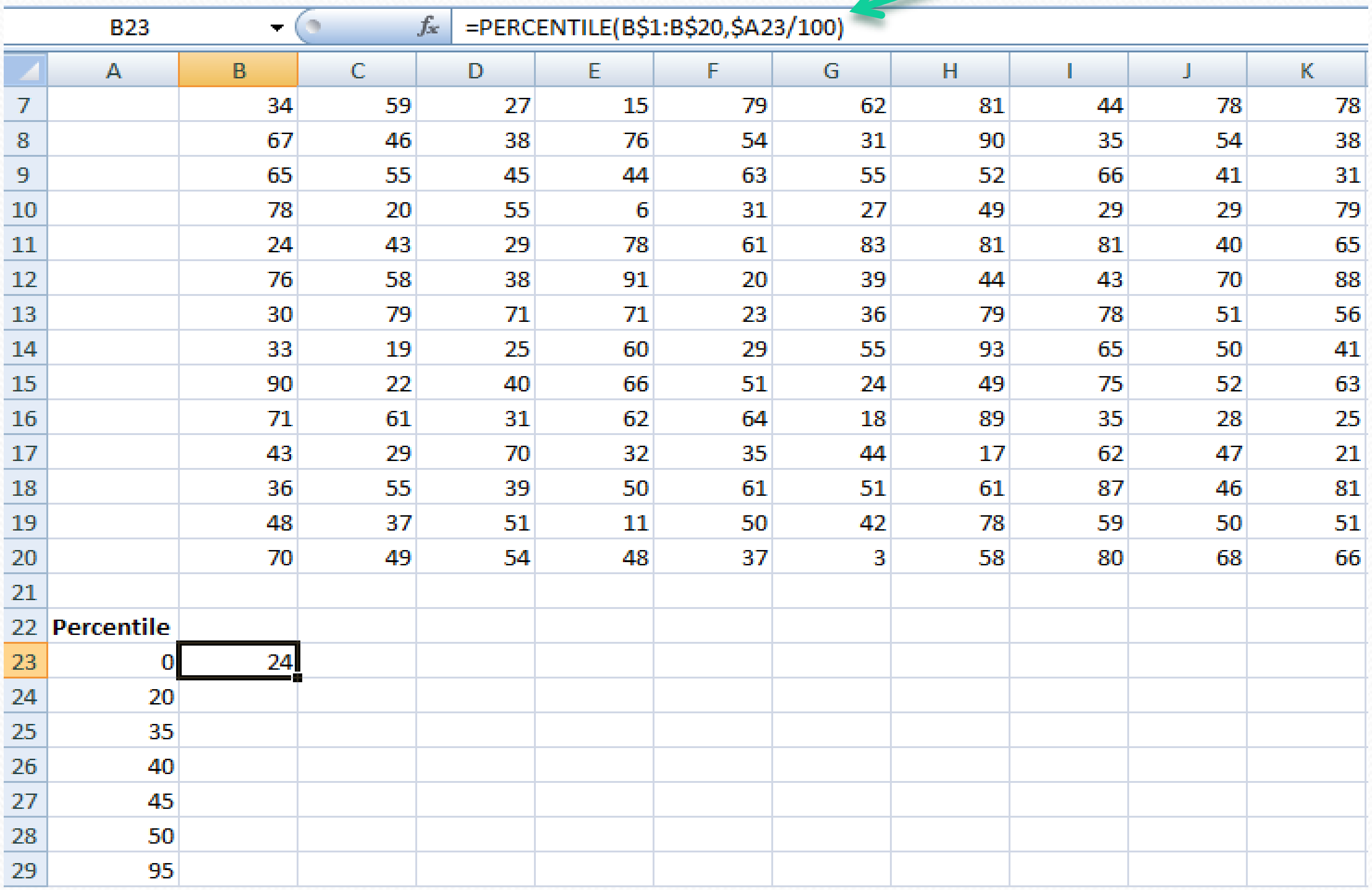
B1:B20

- Stores the column of data points to be analyzed
- Think of what happens when the formula is dragged on to adjacent cells
- DO NOT want to shift down to B2:B21 and so forth – fix the row references
- But DO want shift right to C1:C20 – do not fix the column references
- B$1:B$20

A23/100

- Stores the k value
- Think of what happens when the formula is dragged on to adjacent cells
- DO want to shift down to A24 – do not fix row references
- DO NOT want to shift right to B23 – fix the column references
- $A23/100

Refined Formula

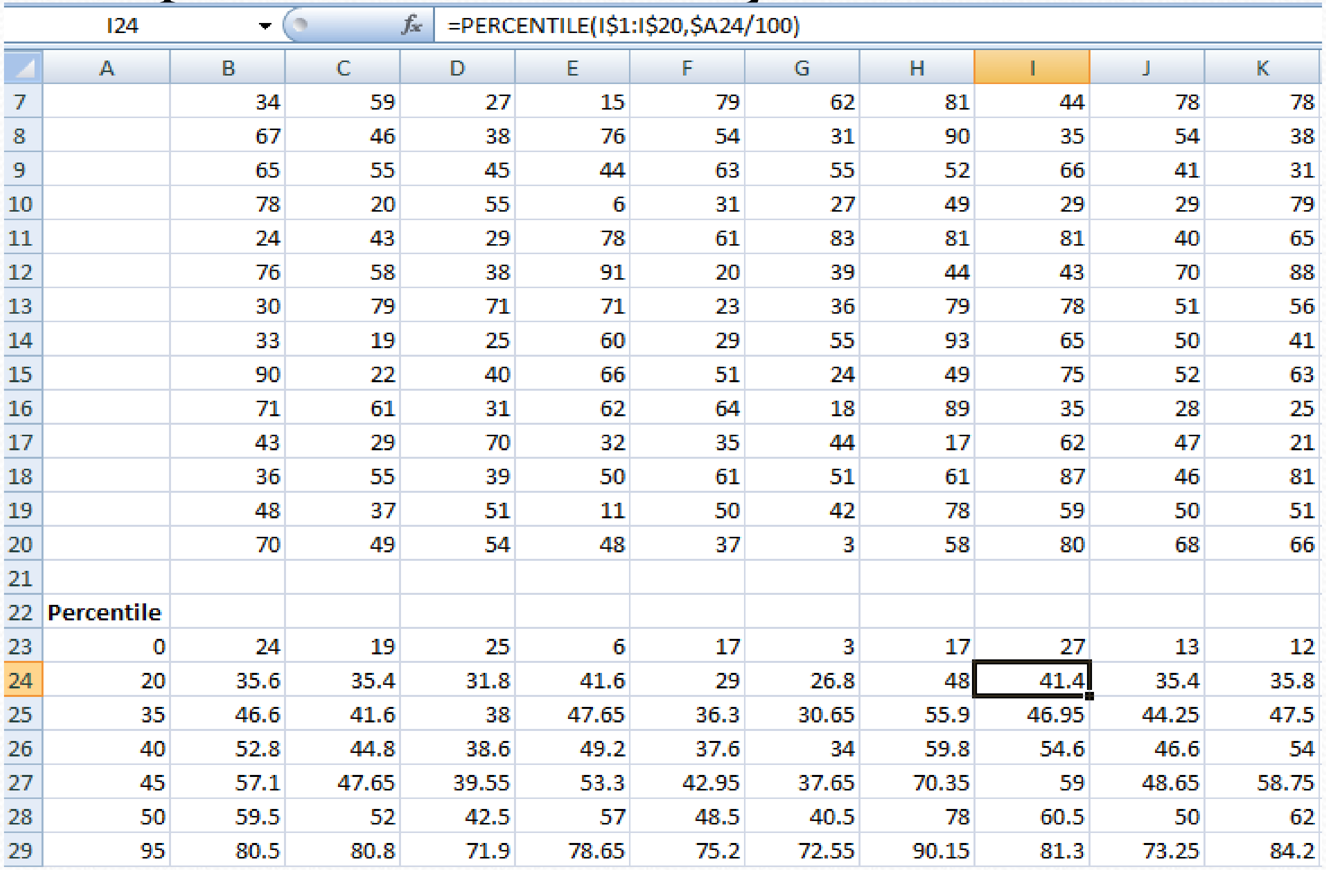


B23 =PERCENTILE(B$1:B$20,$A23/100)

	A	B	C	D	E	F	G	H	I	J	K
7		34	59	27	15	79	62	81	44	78	78
8		67	46	38	76	54	31	90	35	54	38
9		65	55	45	44	63	55	52	66	41	31
10		78	20	55	6	31	27	49	29	29	79
11		24	43	29	78	61	83	81	81	40	65
12		76	58	38	91	20	39	44	43	70	88
13		30	79	71	71	23	36	79	78	51	56
14		33	19	25	60	29	55	93	65	50	41
15		90	22	40	66	51	24	49	75	52	63
16		71	61	31	62	64	18	89	35	28	25
17		43	29	70	32	35	44	17	62	47	21
18		36	55	39	50	61	51	61	87	46	81
19		48	37	51	11	50	42	78	59	50	51
20		70	49	54	48	37	3	58	80	68	66
21											
22	**Percentile**										
23	0	24									
24	20										
25	35										
26	40										
27	45										
28	50										
29	95										

Dynamic Formulas

- Results updates automatically for different k values



I24 =PERCENTILE(I$1:I$20,$A24/100)

	A	B	C	D	E	F	G	H	I	J	K
7		34	59	27	15	79	62	81	44	78	78
8		67	46	38	76	54	31	90	35	54	38
9		65	55	45	44	63	55	52	66	41	31
10		78	20	55	6	31	27	49	29	29	79
11		24	43	29	78	61	83	81	81	40	65
12		76	58	38	91	20	39	44	43	70	88
13		30	79	71	71	23	36	79	78	51	56
14		33	19	25	60	29	55	93	65	50	41
15		90	22	40	66	51	24	49	75	52	63
16		71	61	31	62	64	18	89	35	28	25
17		43	29	70	32	35	44	17	62	47	21
18		36	55	39	50	61	51	61	87	46	81
19		48	37	51	11	50	42	78	59	50	51
20		70	49	54	48	37	3	58	80	68	66
21											
22	**Percentile**										
23	0	24	19	25	6	17	3	17	27	13	12
24	20	35.6	35.4	31.8	41.6	29	26.8	48	41.4	35.4	35.8
25	35	46.6	41.6	38	47.65	36.3	30.65	55.9	46.95	44.25	47.5
26	40	52.8	44.8	38.6	49.2	37.6	34	59.8	54.6	46.6	54
27	45	57.1	47.65	39.55	53.3	42.95	37.65	70.35	59	48.65	58.75
28	50	59.5	52	42.5	57	48.5	40.5	78	60.5	50	62
29	95	80.5	80.8	71.9	78.65	75.2	72.55	90.15	81.3	73.25	84.2

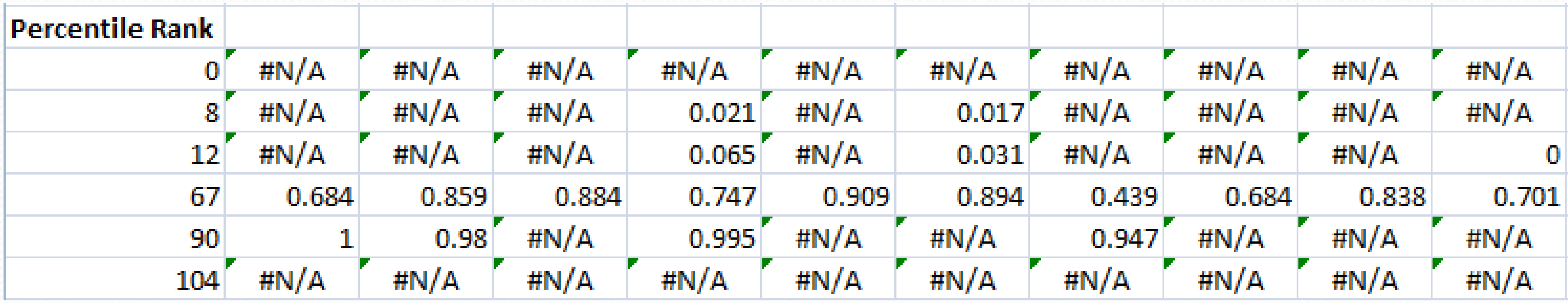
Task 2

- Given several integers, called x, calculate the percentile rank of those integers
- What percentile would these integers fit into?
- If x is out of the range, error would return
  - In that case, display a message that it’s out of the range

- =PERCENTRANK([array], x)

Without Error Trapping

- Similar cell reference fixing as previous
- #N/A errors whenever:
  - x is smaller than the minimum value in the set
  - x is larger than the maximum value in the set



Percentile Rank										
0	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
8	#N/A	#N/A	#N/A	0.021	#N/A	0.017	#N/A	#N/A	#N/A	#N/A
12	#N/A	#N/A	#N/A	0.065	#N/A	0.031	#N/A	#N/A	#N/A	0
67	0.684	0.859	0.884	0.747	0.909	0.894	0.439	0.684	0.838	0.701
90	1	0.98	#N/A	0.995	#N/A	#N/A	0.947	#N/A	#N/A	#N/A
104	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A

=IF()

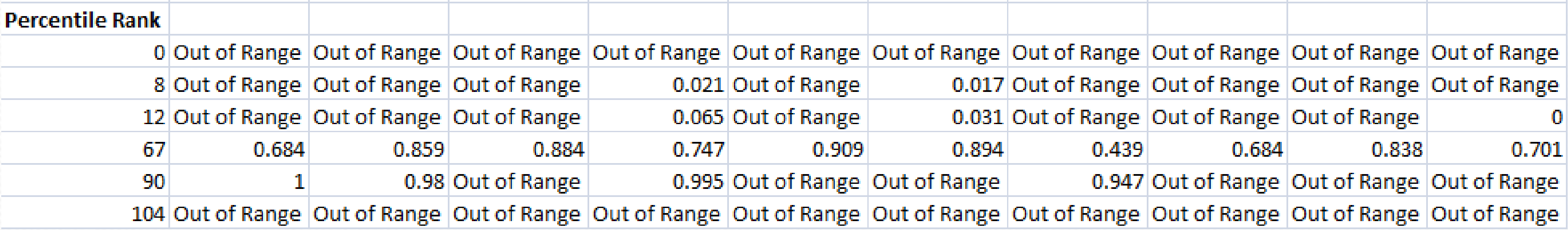
- Returns different values given the certainty of a condition
- =IF([condition],[value if true],[value if false]
  - =IF(1=1,"YES","NO") returns “YES”
  - =IF(1=2,"YES","NO") returns “NO”
- Suppose cell A1 contains “24”
  - Suppose A2 wants to show the value in A1, but only the value is divisible by 11, otherwise leave blank
  - =IF(MOD(A1,11)=0,A1,"")

Nested =IF()

- Suppose cell A1 contains “24”
  - In cell A2, type =IF(MOD(A1,2)=0,IF(MOD(A1,3)=0,"DIV BY 6","DIV BY 2"),"NOT DIV BY 2")
  - If divisible by 2:
    - If further divisible by 3, show that it’s divisible by 6
    - If not further divisible by 3, show that it’s merely divisible by 2
  - If not divisible by 2:
    - Display that it’s not divisible by 2
- Change the value in A1 and see the result

=IFERROR()

- =IFERROR([normal value], [value if error])
- For B23 cell for example, we want =IFERROR(PERCENTRANK(B$1:B$20,$A32),"Out of Range")
- Return PERCENTRANK(B$1:B$20,$A32) to B23 cell, but if that results in an error, return “Out of Range” instead



Percentile Rank										
0	Out of Range	Out of Range	Out of Range	Out of Range	Out of Range	Out of Range	Out of Range	Out of Range	Out of Range	Out of Range
8	Out of Range	Out of Range	Out of Range	0.021	Out of Range	0.017	Out of Range	Out of Range	Out of Range	Out of Range
12	Out of Range	Out of Range	Out of Range	0.065	Out of Range	0.031	Out of Range	Out of Range	Out of Range	0
67	0.684	0.859	0.884	0.747	0.909	0.894	0.439	0.684	0.838	0.701
90	1	0.98	Out of Range	0.995	Out of Range	Out of Range	0.947	Out of Range	Out of Range	Out of Range
104	Out of Range	Out of Range	Out of Range	Out of Range	Out of Range	Out of Range	Out of Range	Out of Range	Out of Range	Out of Range

=IFERROR() vs. =IF()

- Logically, =IFERROR(PERCENTRANK(B$1:B$20,$A32),"Out of Range") is essentially this:

=IF(PERCENTRANK(B$1:B$20,$A32)=#N/A,"Out of Range")

- However, we can't use the latter.
- PERCENTRANK(B$1:B$20,$A32) immediately throws an error, won't compare to #N/A
- So =IFERROR() is the only way to trap that error

=IFERROR() vs. =IF()

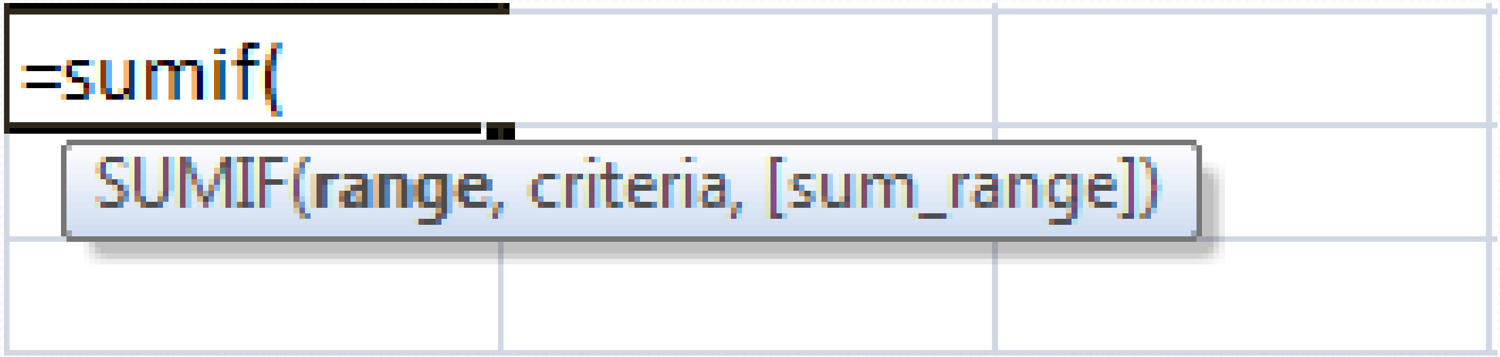
- =IFERROR() is also much cleaner
- Suppose in cell row 39, we want to display sum of rows 1 through 20
  - However, if the sum is less than 1000, display "< 1000" instead of the actual sum
- In B23 cell:

=IF(SUM(B1:B20)<1000,"< 1000",SUM(B1:B20))

- Inevitable "double-typing" the core formula

=SUMIF()

- In the previous example, output changed depending on the final output
  - If the total final out is less than 1000, display the string
- What if we want conditions for each entry?
  - In row 40, sum the entries of rows of 1 to 20, but only each individual entry is greater than 70



=sumif(

SUMIF(range, criteria, [sum_range])

Criteria

- Boolean condition within quotation marks
- Examples:
  - Less than 100: “<100”
  - Equals to 100: “=100”
  - Greater than 100: “>100”
  - Not 100: “<>100”
  - Greater than or equal to 100: “>= 100”
  - Less than or equal to 100: “<=100”
- =SUMIF(B1:B20, ">70") in this scenario

A Note on This Exercise

- No Visual Basic programming was needed for this exercise
- Most times, if a dynamic formula in a cell can give us all the information we need, use them instead of programs
  - Faster, easier debugging
- Much of the functions and algorithms done with formulas are the backbone for VBA programming foundation

Tutorial 1: Variables and Arrays

Programming = Work With…

- Data
- Data
- Data
- Data
- Data

Let’s Get Started: Variables

- Different types of data. Consider:
  - Name of city: New York (words)
  - Length of 14th St: 2.00 (decimal)
  - Population: 8,000,000 (integer)
  - MetroCard fare: 2.25 (currency)
  - Longitude: 74 ° 00’00” (longitude)
- Variables are categorized by the type of data they hold.
- There are basic types, and there can be user-defined.
- Variables and their types aren’t necessarily units.

Some Basic Variables

- Consider a course:
  - Class size: 30 (integer)
  - Class location: Hamilton Hall (string)
  - Pass/Fail allowed: false (boolean)
  - Average grade: 94.4 (double)
  - Average letter grade: A (character)

CULPA.INFO

Declaring Variables

- *Dim size as Integer*
- *Dim location as String*
- *Dim passFail as Boolean*
- *Dim avgGrade as Double*
- *Dim ltrGrade as Char*

- Dim: denote local variables (existence in the running of procedure)
- Variable Names: begin with letter, only contain letter, number, or underscore, cannot be reserved word

Initializing Variables

- Can do all the declaration in one line as follows:
- *Dim size as Integer, location as String, passFail as Boolean, avgGrade as Double, ltrGrade as Char*

- *size = 30*
- *location = "Hamilton Hall"*
- *passFail = False*
- *avgGrade = 94.4*
- *ltrGrade = 'A'*

Variants

- Variables not restricted to specific type
- No need to declare by type

- *size = 30*
- *location = "Hamilton Hall"*
- *passFail = False*
- *avgGrade = 94.4*
- *ltrGrade = ‘A’*

Multiple Similar Variables

- Suppose we want to store the price of pineapple for each day of the week
- We can declare 7 separate variables:
  - *Dim p0 as Double, p1 as Double … p6 as Double*
- Difficult to keep track of all of the variables
- Difficult to access each of the variables
- Gets particularly difficult when the number of entries grow higher

Solution: Arrays

- Array: block of pigeonholes
- 7 pigeonholes, each representing a day of week:

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Index	0	1	2	3	4	5	6
Price	$5.03	$0.13	$1.51	$7.75	$7.24	$1.99	$0.64

Arrays

- *Dim prices(6) as Double*
- *prices(1)* to retrieve entry from index 1 (second entry)
- *prices(7)* will give out-of-bounds error
- Benefit: the index can be accessed by other variables:
  - *Dim i as integer*
  - *prices(i)* gives the (i+1)th entry

Arrays

- prices(0) = 5.03
- prices(1) = 0.13
- prices(2) = 1.51
- prices(3) = 7.75
- prices(4) = 7.24
- prices(5) = 1.99
- prices(6) = 0.64

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Index	0	1	2	3	4	5	6
Price	$5.03	$0.13	$1.51	$7.75	$7.24	$1.99	$0.64

Multidimensional Array

- Row x column
- *Dim matrix(1,2) as Integer*
  - Creates 2x3 matrix with 2 rows, 3 columns

matrix(0,0)	matrix(0,1)	matrix(0,2)
matrix(1,0)	matrix(1,1)	matrix(1,2)

Dynamic Array

- *Dim sample(9) as String*
  - Creates string array of size 10
- *sample(0) = "Introduction"*
  - Now suppose we want to increase the array size to 100
- *ReDim sample(99)*
  - This would erase existing data, such as “Introduction” in cell index 0
- *ReDim Preserve sample(99)*
  - This preserves existing data and changes size

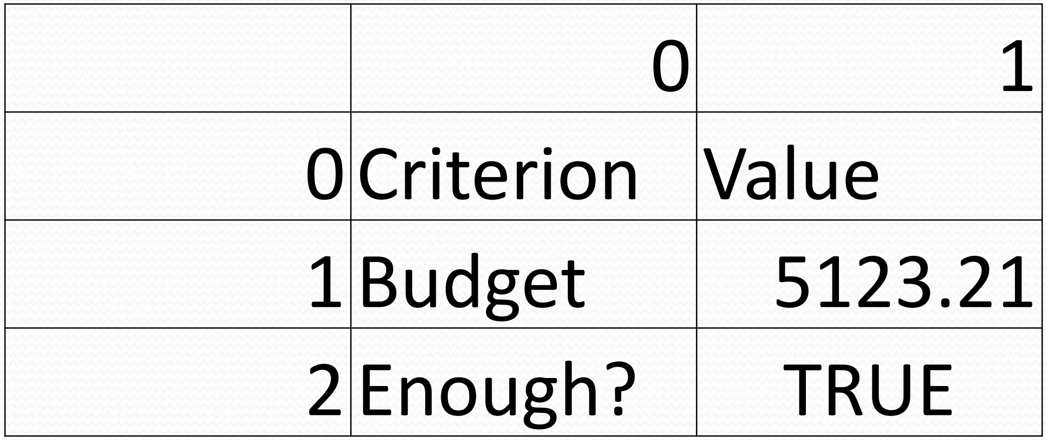
Practice Exercise

- What’s the result after executing this code?

```
Dim dat(2, 1)
dat(0, 0) = "Criterion"
dat(0, 1) = "Value"
dat(1, 0) = "Budget"
dat(1, 1) = 5123.21
dat(2, 0) = "Enough?"
dat(2, 1) = True
ReDim Preserve dat(2,1)
```

Practice Exercise

- Did you catch that: dat originally was 3x2 two-dimensional variant array? The very last line did nothing since the new dimension is the same?



	0	1
0	Criterion	Value
1	Budget	5123.21
2	Enough?	TRUE

Case 3: Hello World

Excel Hierarchy

- Workbook (“Data.xls”)
- Worksheet (“Sheet1”)
- Row / Column (1 / “A”)
- Cell (“A1”)

Moments to Run Code

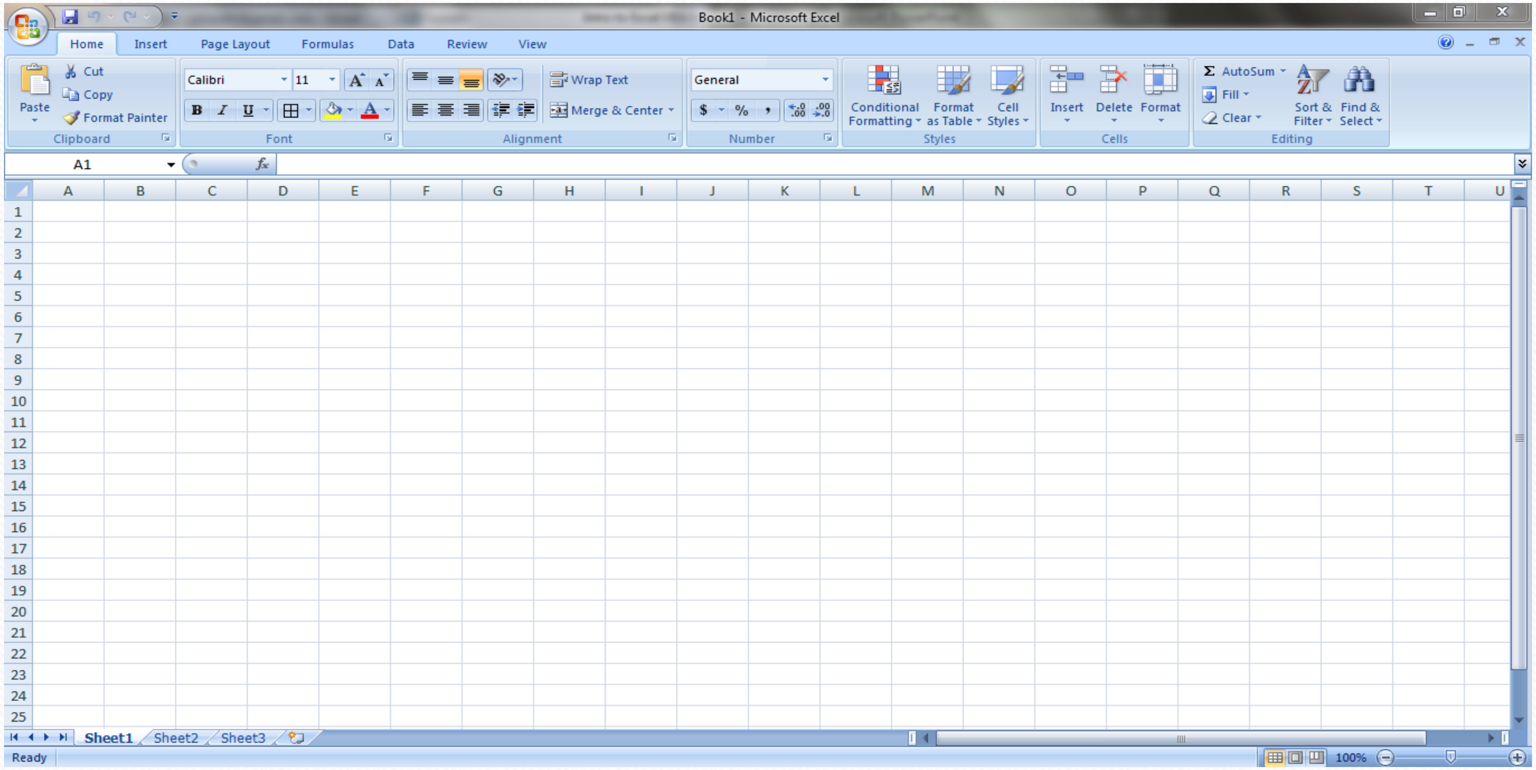
- When user selects to run: modules
  - User runs the code to tally all the numbers
  - User runs the code to form a table from entries
  - Etc.
- When some action is performed: worksheet and workbook codes
  - Code is run when the workbook is opened
  - Code is run when the worksheet is modified
  - Etc.

Note on Running VBA Codes

- Programs updates automatically
    - No need to “compile” as we would need to in other programming languages
- No undoing once programs are run and make changes
    - Highly recommend saving the file before running the code
    - Even occasionally recommend saving a new copy of the file to run the code on

Worksheet Codes

- Task: Display “Hello World” to the window when contents of “Sheet1” has been changed.

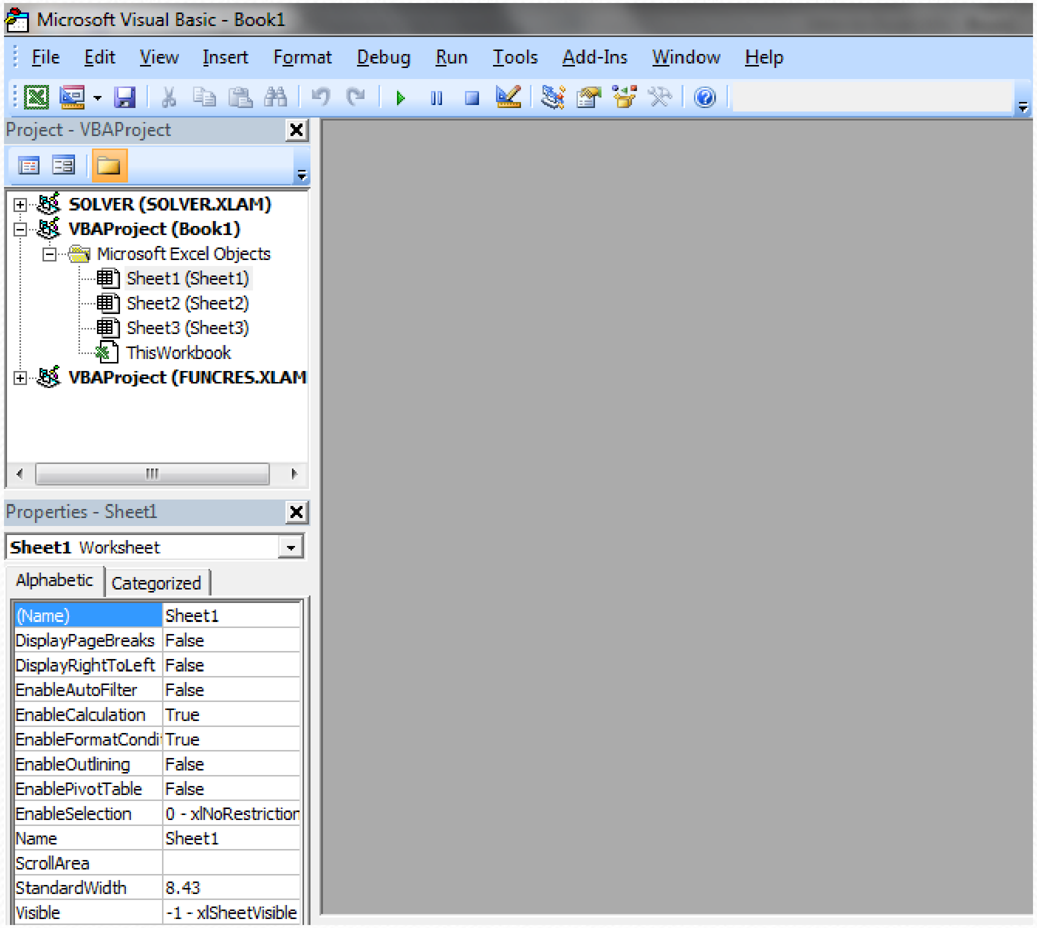


Most Important Key Sequence

- To open the screen for Visual Basic editing from the Excel workbook window:

ALT + F11

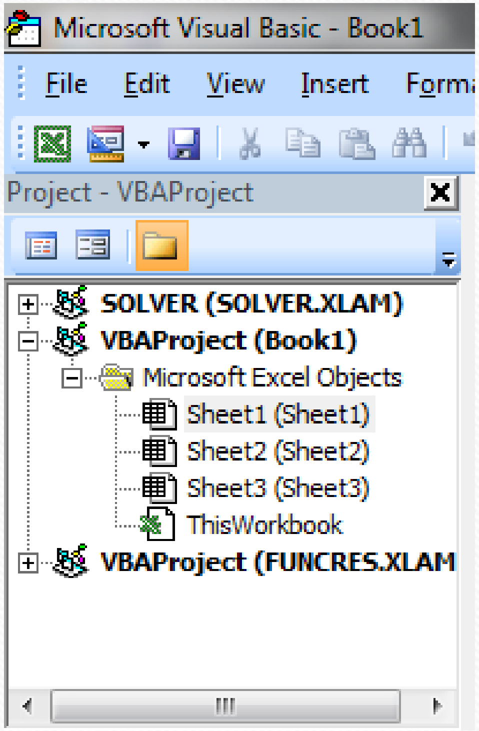
ALT + F11



(Name)	Sheet1
DisplayPageBreaks	False
DisplayRightToLeft	False
EnableAutoFilter	False
EnableCalculation	True
EnableFormatCondi	True
EnableOutlining	False
EnablePivotTable	False
EnableSelection	0 - xlNoRestriction
Name	Sheet1
ScrollArea	
StandardWidth	8.43
Visible	-1 - xlSheetVisible

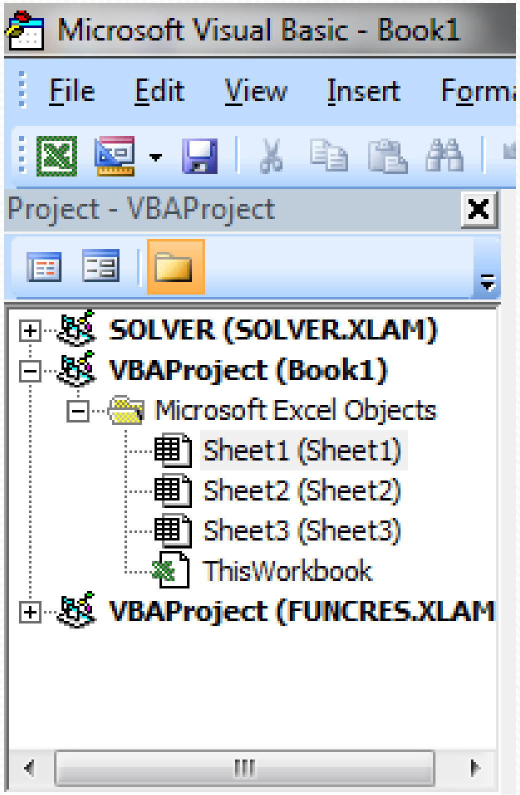
Visual Basic Window

- Each sheet has been set by default as an object
- The workbook is also an object
- Codes that runs when actions are performed, are written from here



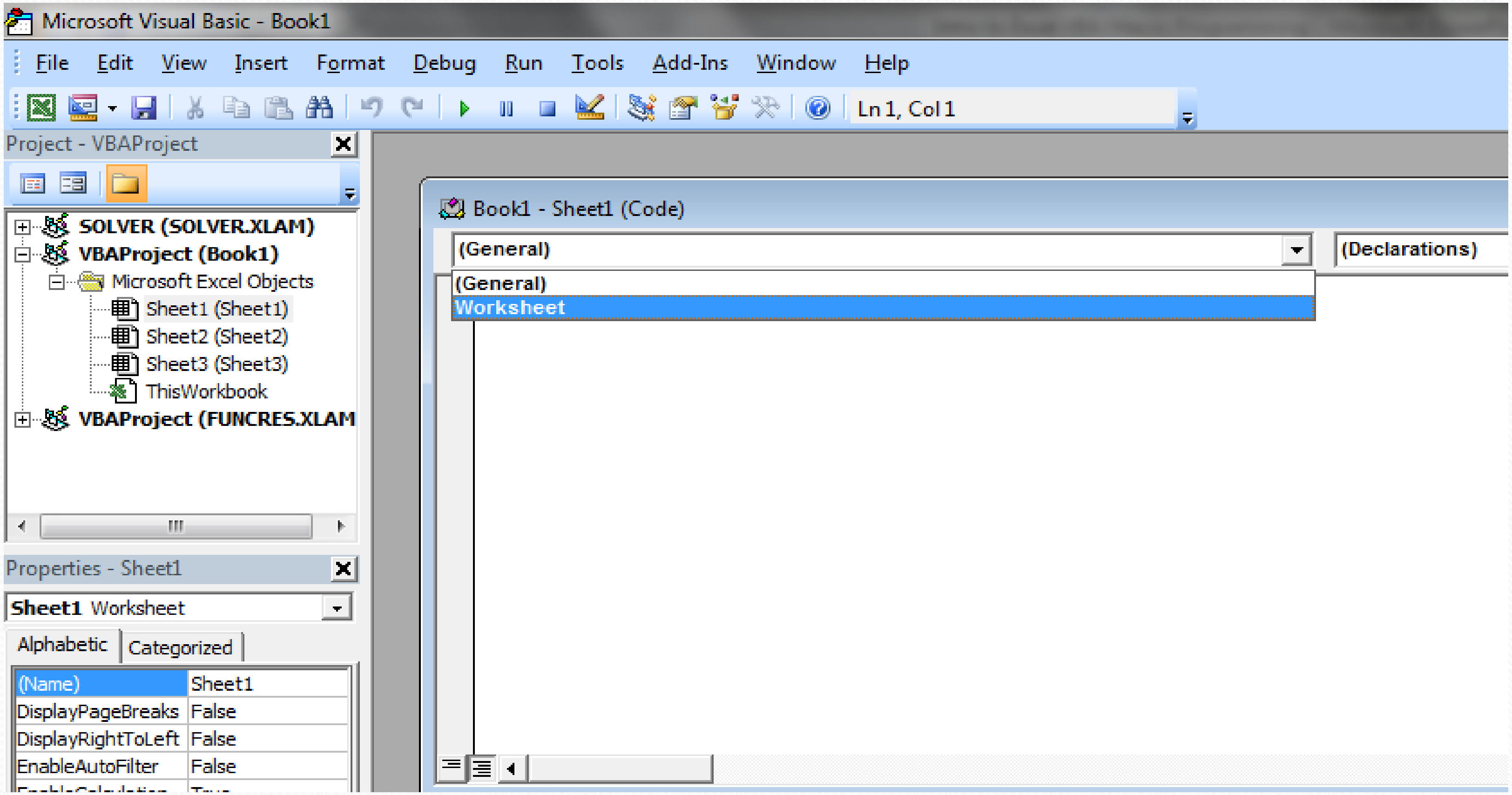
Hello World

- Recall, task: Display “Hello World” to the window when contents of “Sheet1” has been changed.
- Double-click on “Sheet1”



Hello World

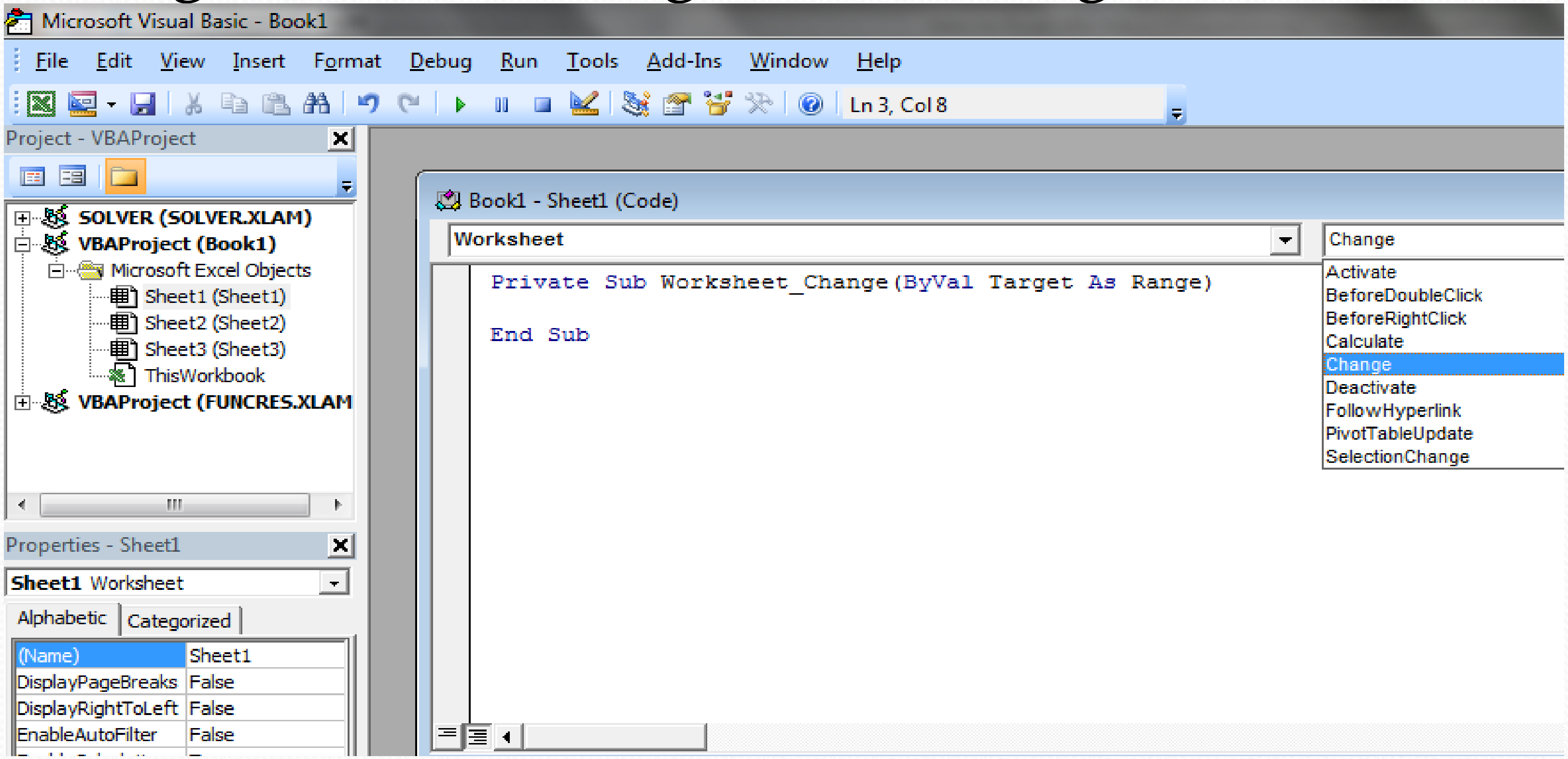
- Select “Worksheet”



(Name)	Sheet1
DisplayPageBreaks	False
DisplayRightToLeft	False
EnableAutoFilter	False

Hello World

- Since we want code to run when this worksheet is changed, select “Change” from the right menu



```
Private Sub Worksheet_Change(ByVal Target As Range)

End Sub
```

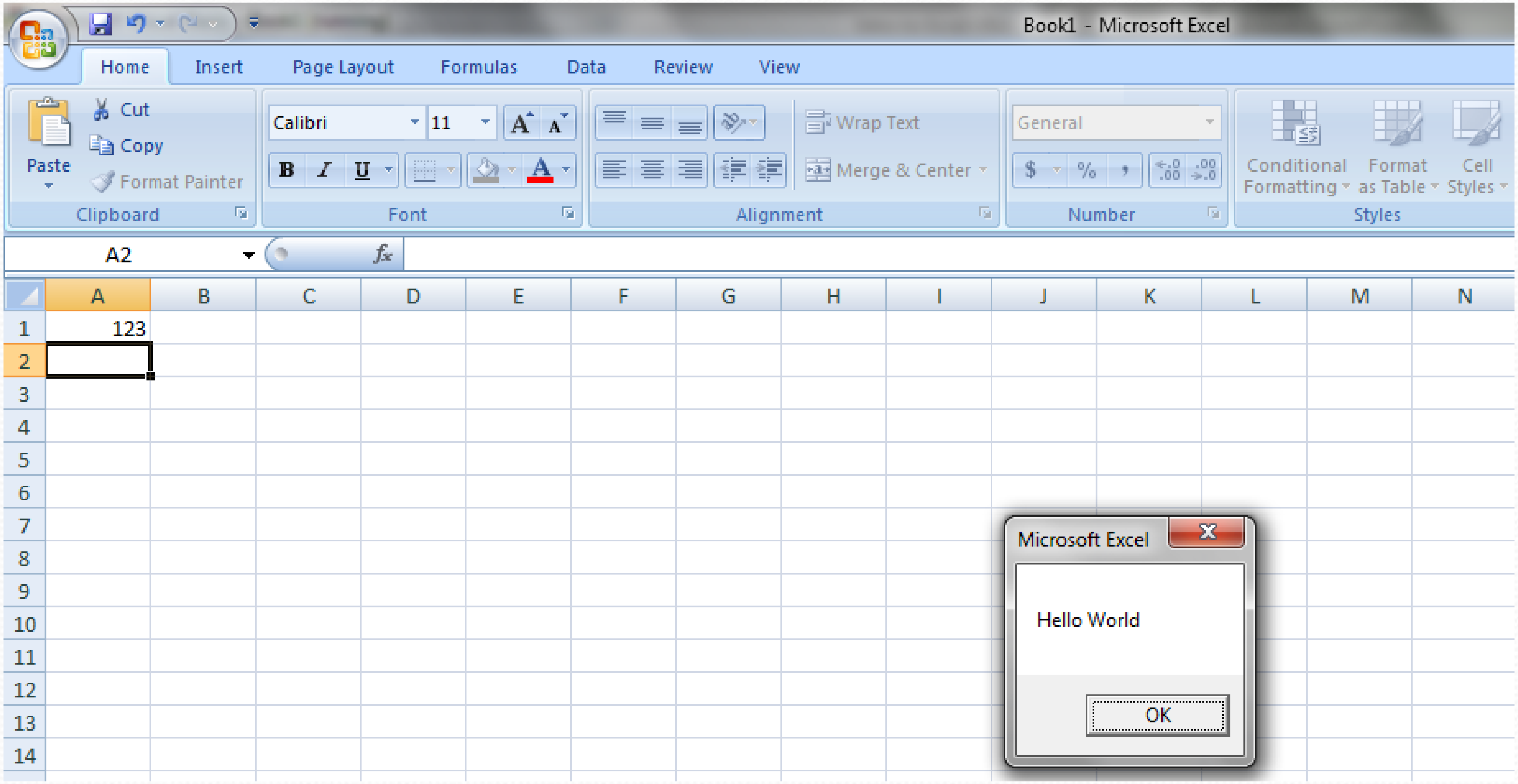
(Name)	Sheet1
DisplayPageBreaks	False
DisplayRightToLeft	False
EnableAutoFilter	False

Hello World

```
Private Sub Worksheet_Change(ByVal Target As Range)
MsgBox "Hello World"
End Sub
```

- “Hello World” is a string
- MsgBox “Hello World” displays a pop-up box with the words “Hello World”

Hello World



Book1 - Microsoft Excel

A2

	A
1	123
2	

Microsoft Excel

Hello World

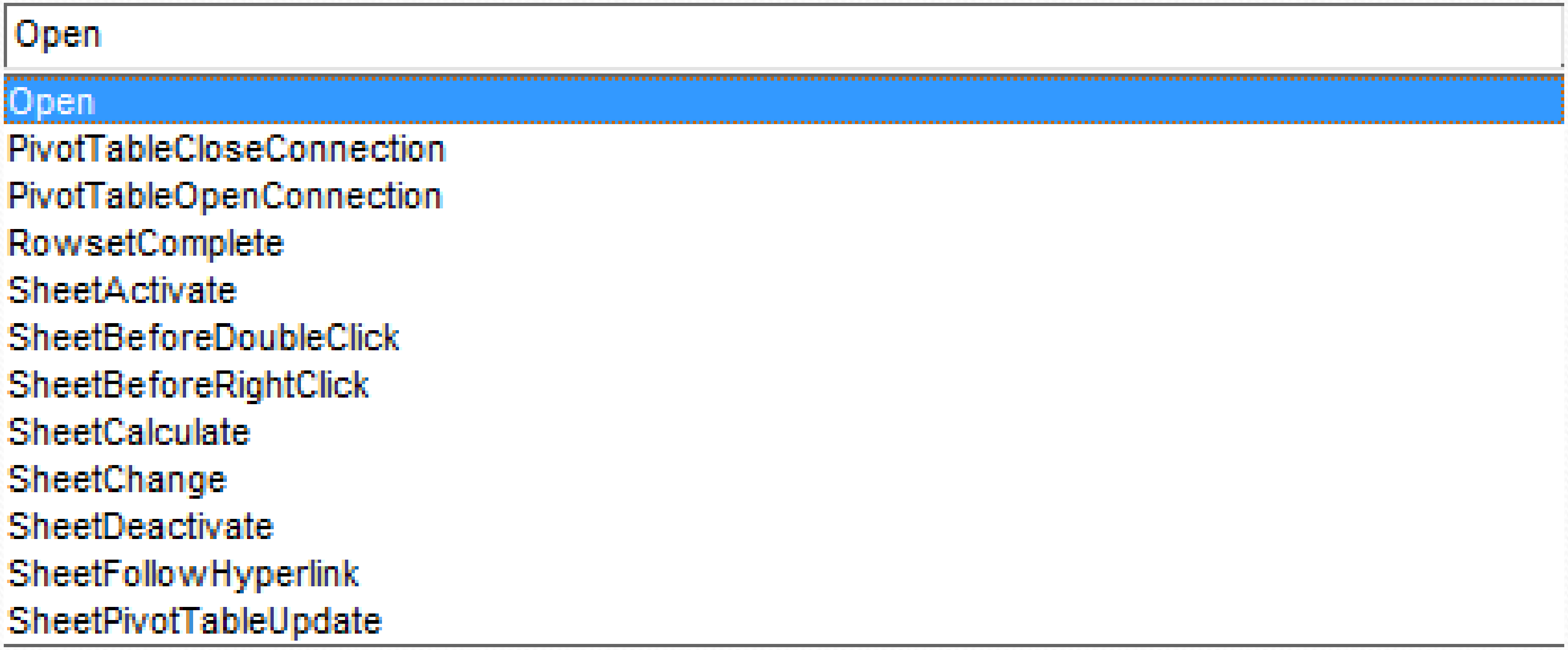
OK

Other Worksheet Actions

- Activate
- BeforeDoubleClick
- BeforeRightClick
- Calculate
- Change
- Deactivate
- FollowHyperlink
- PivotTableUpdate
- SelectionChange

Workbook Actions

- Open: codes to be run when the workbook opens



Open

Open
PivotTableCloseConnection
PivotTableOpenConnection
RowsetComplete
SheetActivate
SheetBeforeDoubleClick
SheetBeforeRightClick
SheetCalculate
SheetChange
SheetDeactivate
SheetFollowHyperlink
SheetPivotTableUpdate

Practice Exercise

- Suppose you want a pop-up that says “Welcome to the Database” when you open the workbook, and then want a pop-up that says “Are you super sure?” before any calculations are performed on Sheet1.

Practice Exercise

- ThisWorkBook > WorkBook > Open:

```
Private Sub Workbook_Open()
MsgBox "Welcome to the Database!"
End Sub
```

- Sheet1 > WorkSheet > Calculate:

```
Private Sub Worksheet_Calculate()
MsgBox "Are you sure?"
End Sub
```

Tutorial 2: Functions

A Word on Functions

- Excel has pre-defined function, including:
  - =SUM() returns the sum of an array of numbers
  - =AVERAGE() returns the average value of an array of number
  - Etc.
- We can write, define, and use our own functions
  - For example, a function that takes an array of numbers, and returns the product of the maximum value and the minimum value

Intro to Modules

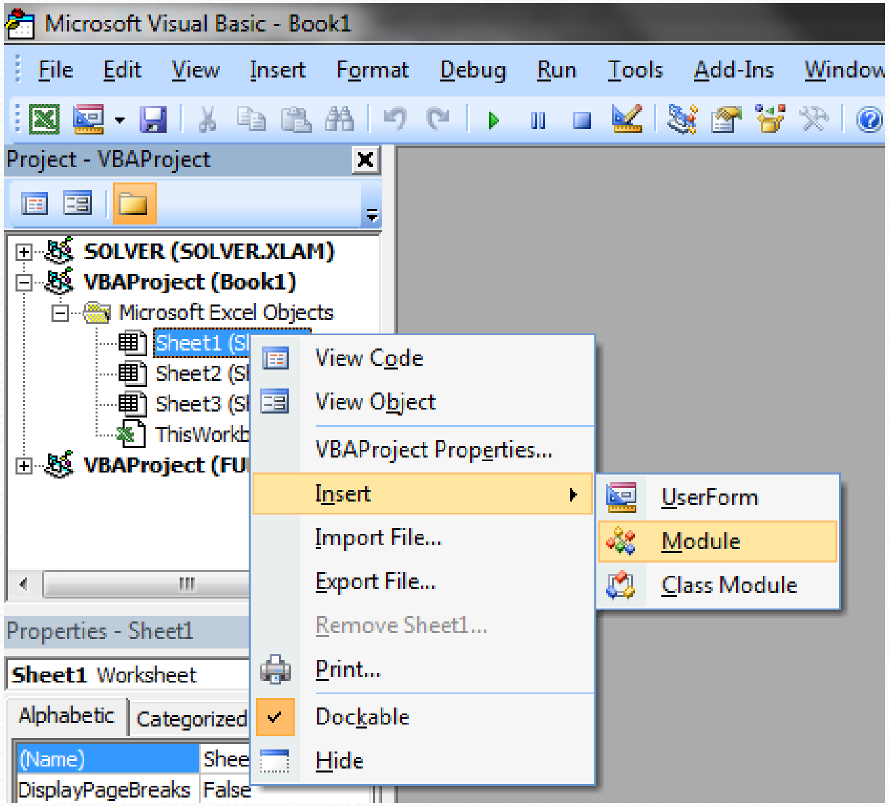
- Subroutines (Sub)
  - Piece of code that perform a set of actions or calculations or a combination of the two
  - Does not return a value
- Functions
  - Exactly like a subroutine except returning a value
  - Think of f(x, y, z)
  - Can have as many inputs, but returns one value

Function Example

- Suppose given three digits a, b, c, return the number abc
- If digits 4, 5, 6 are passed, the function returns 456
- Algorithm: 100*a+10*b+c
- Let’s call this function Concat

Module

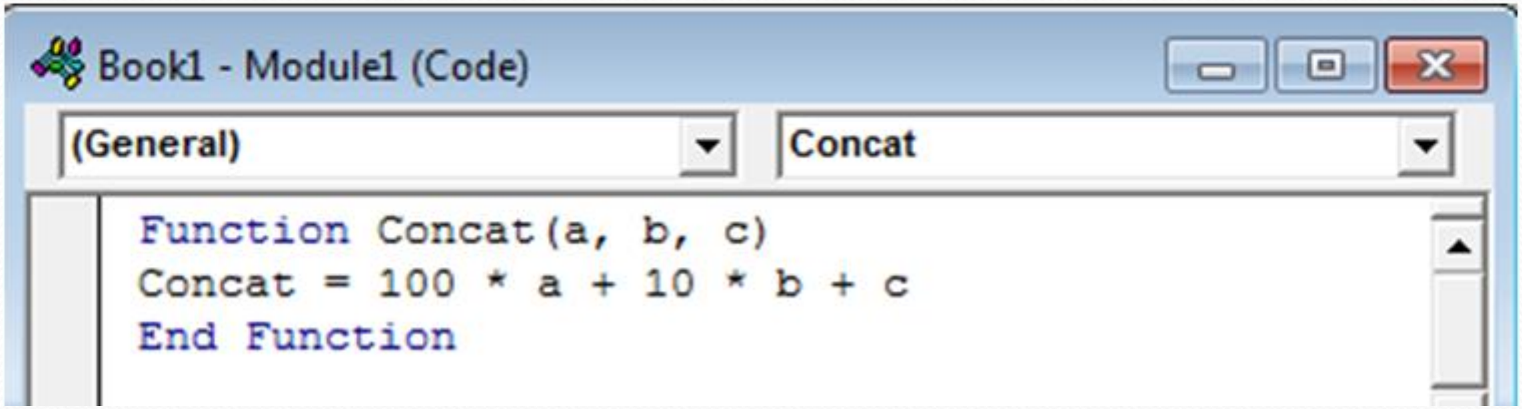
- Can no longer write codes under the worksheet objects
- Right-click Sheet > Insert > Module



Concat

```
Function Concat(a, b, c)
Concat = 100 * a + 10 * b + c
End Function
```

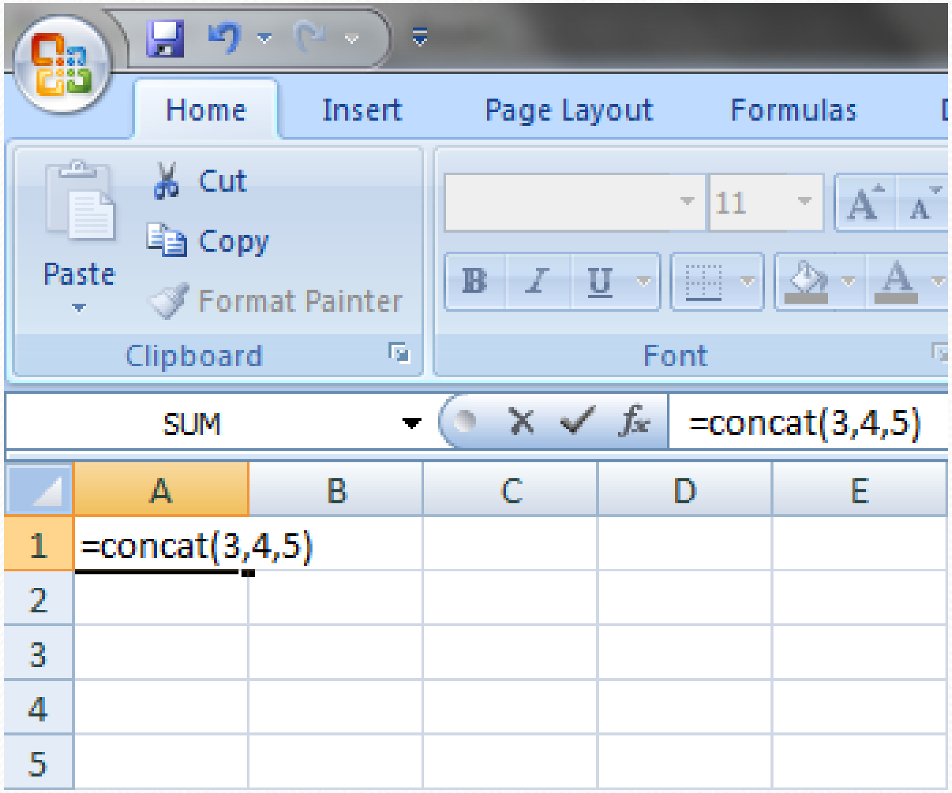
Function name equals the value to be returned



```
Function Concat(a, b, c)
Concat = 100 * a + 10 * b + c
End Function
```

Using Functions in Excel

- You can use this user-defined function in Excel cells, just like how you use =sum() or =average()

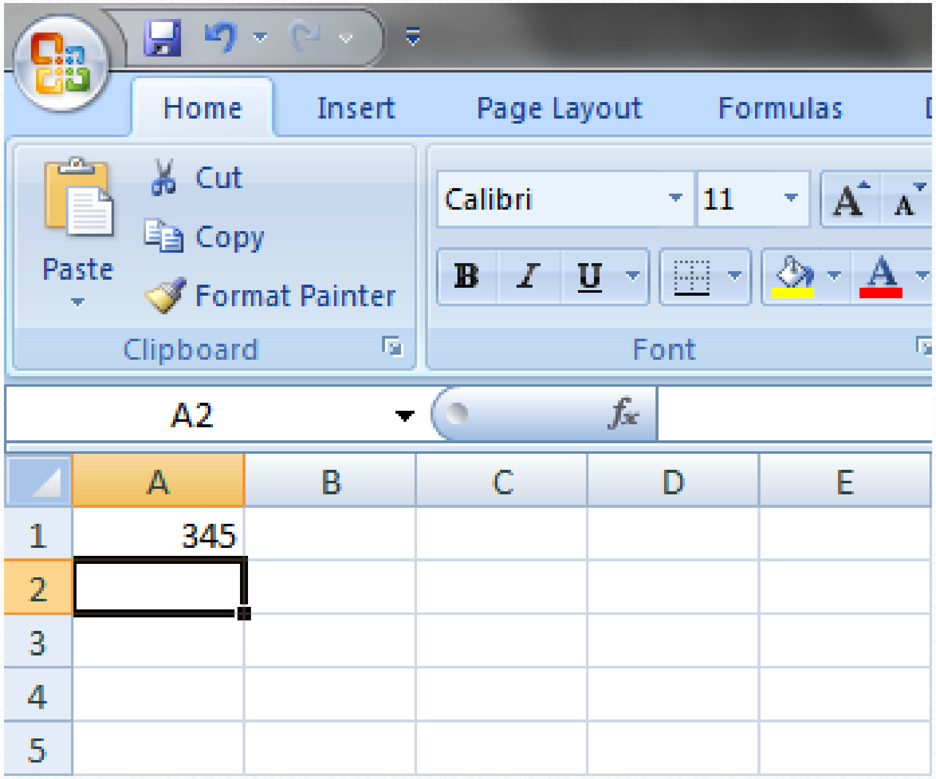


=concat(3,4,5)

=concat(3,4,5)

Concat(3, 4, 5)

- Should return 345
- And indeed it does



A2

	A	B	C	D	E
1	345				
2					
3					
4					
5					

Practice Exercise 1

- Without using pre-existing Excel functions, write your own function that does the following:
  - Take in 2 integers
  - Raise the first integer to the power of the second integer
  - Take that result and modulo by the second integer
  - Return this final result
- Recall: a modulo b is the remainder after a / b, performed in VBA via *a Mod b*

Practice Exercise 1

```
Function Special(a, b)
Special = (a ^ b) Mod b
End Function
```

- In A1 cell, can type =special(2,3)
- Upon hitting enter, 2 would show up in A1
- 2^3 = 8, and 8 mod 3 = 2

Practice Exercise 2

- Without using pre-existing Excel functions, write your own function that does the following:
  - Take in a string and an integer
  - Return true or false – as to whether the length of the string is equal to the integer
- Note: the function *Len(string_variable)* returns the length of string_variable as an integer

Practice Exercise 2

- Function takes in two variables:
  - A string
  - An integer
- Need to compute the length of the string
- Need to compare the length of the string, to the integer
- Return whether or not (true or false) the two values are equal

Practice Exercise 2

```
Function StrLength(a, b)
Length = Len(a)
StrLength = (Length = b)
End Function
```

- Note that (Length = b) is a boolean statement. It is either true or false.
- =strlength("abc",4) returns FALSE
- =strlength("abc",3) returns TRUE

Tutorial 3: Loops and Decisions

Loops and Decisions

- Loops:
  - For
  - While
  - Do Until
- Decisions:
  - If statements
    - If … Then … [ElseIf … Then … Else …] EndIf
    - Entries in [ ] is optional

Sample Exercise

- Print out 1, 2 … 1000 in column A of the first 1000 rows

```
Cells(1,1) = 1
Cells(2,1) = 2
Cells(3,1) = 3
…
Cells(1000,1) = 1000
```

For Loop Syntax

```
For Row = 1 To 1000
Cells(Row, 1) = Row
Next Row
```

```
For [variable name] = [start] To [end]
[...codes to be run for each iteration...]
Next [variable name]
```

Same Exercise: While Loop

```
Row = 1
While Row <= 1000
Cells(Row, 1) = Row
Row = Row + 1
Wend
```

```
While [conditional statement]
[…does to be run…]
Wend
```

Common Fatal Error: While Loop

```
Row = 1
While Row <= 1000
Cells(Row, 1) = Row
Row = Row + 1
Wend
```

- For loop forces us to increment our counter: “next Row”
- We have to do that on our own for While loop

Do Until

- Similar to While loop

```
Row = 1
Do Until Row=1000
Cells(Row, 1) = Row
Row = Row + 1
Loop
```

Word on Loops

- Usually interchangeable
- Choice of which loop to use usually coming down to personal preference
- For loop usually best when the number of iterations are known

If Statements

- Given 1, 2, 3 … 1000 printed in column A
- Display in column B whether each integer is divisible by 2, 3, both, or neither
- Recall: a number is divisible by 6 if it's divisible by both 2 and 3

Algorithm Approach

- Traverse through 1, 2 … 1000
  - If divisible by 6, note it
  - Otherwise … check if it’s divisible by 2 or 3 and note if so
- Important: If it’s divisible by 6 already, no need to check if it’s divisible by 2 or 3
- Recall: a Mod b gives the remainder of a / b
- In another words, a is divisible by b if a Mod b = 0

The Code

```
For Row = 1 To 1000
If Cells(Row, 1) Mod 6 = 0 Then
Cells(Row, 2) = "Divisible by 6"
ElseIf Cells(Row, 1) Mod 2 = 0 Then
Cells(Row, 2) = "Divisible by 2"
ElseIf Cells(Row, 1) Mod 3 = 0 Then
Cells(Row, 2) = "Divisible by 3"
End If
Next Row
```

If Ladder

- An If ladder begins with If … Then
  - Can include multiple ElseIf … Then
  - Ladder ends with EndIf
- Within a ladder, as long as the first satisfying condition is met, other conditions are ignored and skipped … even if they are true

Think About It

- What will happen after this switch or ordering?

```
For Row = 1 To 1000
If Cells(Row, 1) Mod 2 = 0 Then
Cells(Row, 2) = "Divisible by 2"
ElseIf Cells(Row, 1) Mod 3 = 0 Then
Cells(Row, 2) = "Divisible by 3"
ElseIf Cells(Row, 1) Mod 6 = 0 Then
Cells(Row, 2) = "Divisible by 6"
End If
Next Row
```

Think About it

- Take the number 12:
  - Since it 12 % 2 = 0, satisfying the first condition, will display “Divisible by 2” and exit the If ladder
  - Even if it’s divisible by 3 and 6 also…
- Instead, breaking this ladder into multiple different If statements works successfully

Hypothetical Revision

- Overriding each time … just inefficient

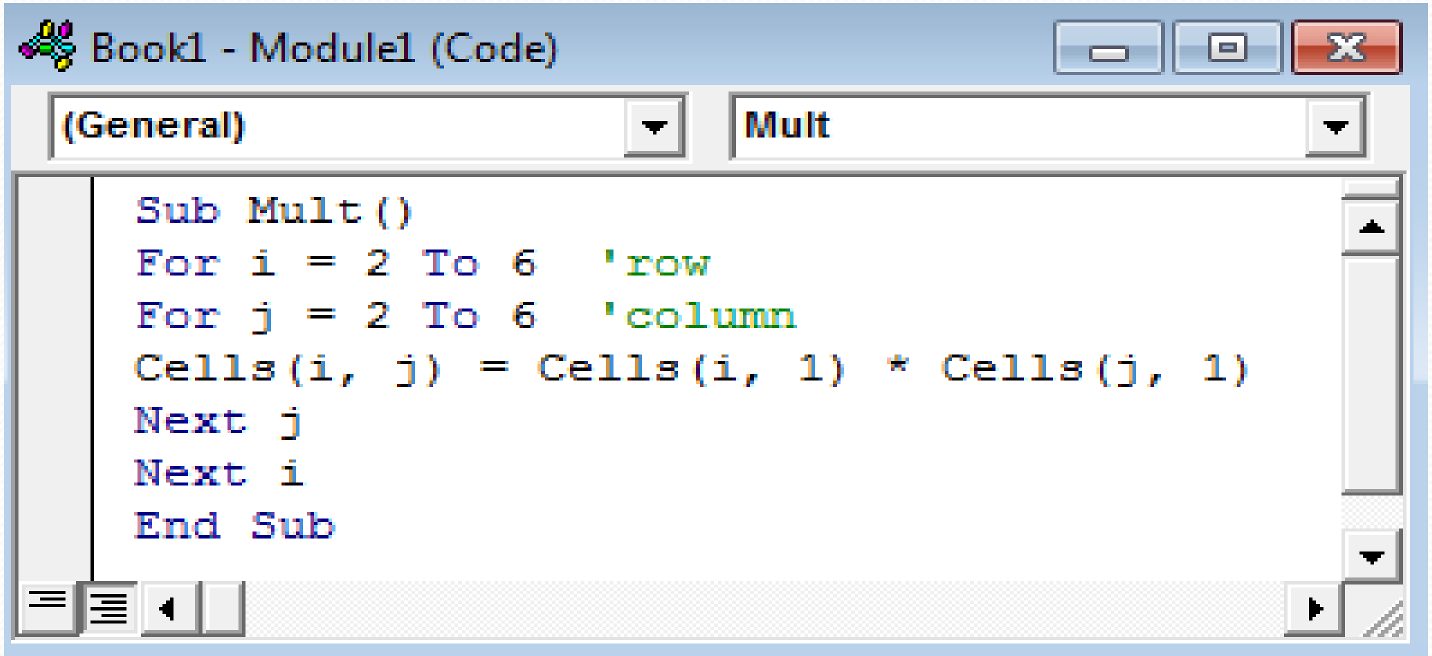
```
If Cells(Row, 1) Mod 2 = 0 Then
Cells(Row, 2) = "Divisible by 2"
End If

If Cells(Row, 1) Mod 3 = 0 Then
Cells(Row, 2) = "Divisible by 3"
End If

If Cells(Row, 1) Mod 6 = 0 Then
Cells(Row, 2) = "Divisible by 6"
End If
```

Remember This?

- Nested for loop
- (i, j) = (2,2), (2,3) … (2,6), (3,2), (3,3) … (3,6), (4,2) … (4,6), (5,2) … (5,6), (6,2) … (6,6)



```
Sub Mult()
For i = 2 To 6   'row
For j = 2 To 6   'column
Cells(i, j) = Cells(i, 1) * Cells(j, 1)
Next j
Next i
End Sub
```

Case 4: Gradebook Tallying

Task

- Suppose this grading scheme:
  - With the caveat that up to 2 assignments can be dropped
  - Compute the overall grade

Assignment	Weight	Points	Total
HW 1	10%		
Quiz 1	15%		
HW 2	10%		
Quiz 2	15%		
Test 1	20%		
HW 3	10%		
Test 2	20%		

Illustration

- For example, if Quiz 1 and Test 1 are dropped:
  - 35% of the assignments are dropped
  - HW 1 is now worth (10%) / (1 – 0.35) and so on…

Assignment	Weight	Points	Total
HW 1	10%		
Quiz 1	15%		
HW 2	10%		
Quiz 2	15%		
Test 1	20%		
HW 3	10%		
Test 2	20%		

Approach

- Take into consideration:
  - Varying total points for assignments
  - Unexpected position of empty slots

Assignment	Weight	Points	Total
HW 1	10%	18	20
Quiz 1	15%		10
HW 2	10%	43	50
Quiz 2	15%	23	24
Test 1	20%		50
HW 3	10%	9	20
Test 2	20%	49	50

Approach

- Calculate grade for each assignment
- If score cell is empty, keep track of the weight of the assignment
- Need Excel functions:
  - Determine if cell is empty
  - Return different depending on whether or not another cell is blank

Useful Excel Functions

- =IF() in Excel
  - =IF([boolean statement], [value if true], [value if false])
  - =IF(1=1, 10, 20) would return 10 since “1=1” is true
  - =IF(1=2, 10, 20) would return 20 since “1=2” is false
- =ISBLANK() in Excel
  - =ISBLANK([cell reference])
  - Returns TRUE or FALSE
  - If A1 is empty, =ISBLANK(A1) returns TRUE

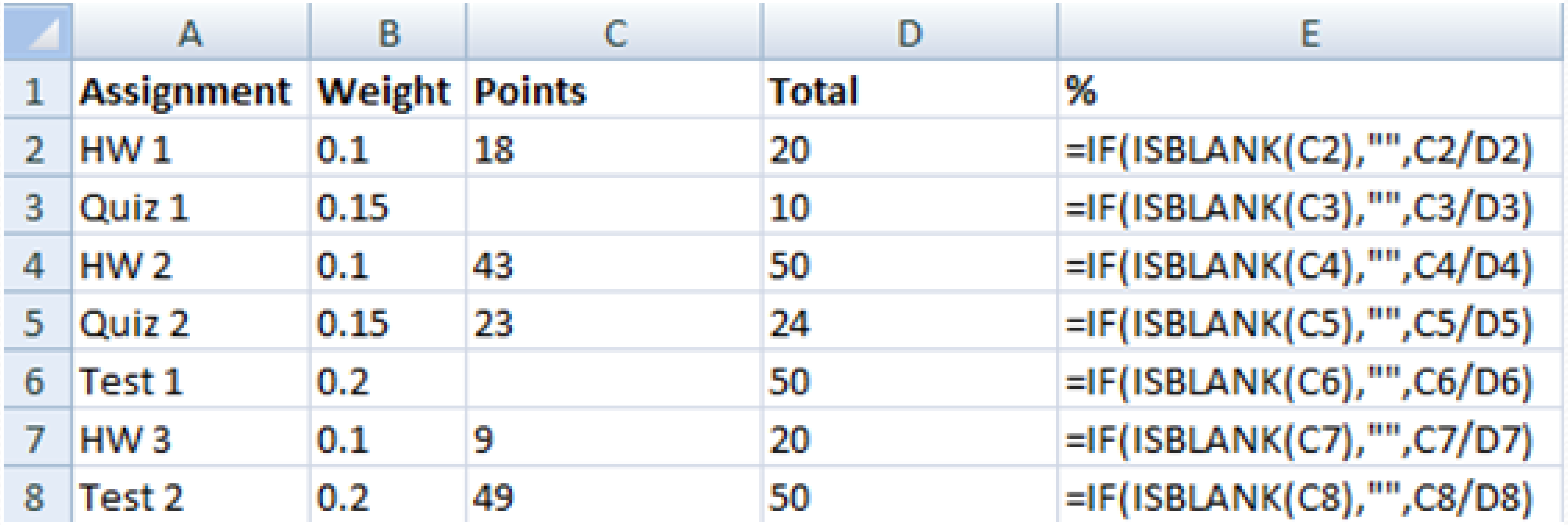
Approach

- If no points are listed in the POINTS column, a grade should not be calculated: not 0%!

Assignment	Weight	Points	Total	Grade %
HW 1	10%	18	20	90.00%
Quiz 1	15%		10	
HW 2	10%	43	50	86.00%
Quiz 2	15%	23	24	95.83%
Test 1	20%		50	
HW 3	10%	9	20	45.00%
Test 2	20%	49	50	98.00%

Approach

- If the “Points” column is blank, return blank in the “%” column. Otherwise, give the result of points / total



	A	B	C	D	E
1	**Assignment**	**Weight**	**Points**	**Total**	**%**
2	HW 1	0.1	18	20	=IF(ISBLANK(C2),"",C2/D2)
3	Quiz 1	0.15		10	=IF(ISBLANK(C3),"",C3/D3)
4	HW 2	0.1	43	50	=IF(ISBLANK(C4),"",C4/D4)
5	Quiz 2	0.15	23	24	=IF(ISBLANK(C5),"",C5/D5)
6	Test 1	0.2		50	=IF(ISBLANK(C6),"",C6/D6)
7	HW 3	0.1	9	20	=IF(ISBLANK(C7),"",C7/D7)
8	Test 2	0.2	49	50	=IF(ISBLANK(C8),"",C8/D8)

Approach

- Multiply each assignment grade by the assignment weight, sum the product
- Normalize the overall weight, to take into consideration dropped assignments

Assignment	Weight	Points	Total	%
HW 1	10%	18	20	90.00%
Quiz 1	15%		10	
HW 2	10%	43	50	86.00%
Quiz 2	15%	23	24	95.83%
Test 1	20%		50	
HW 3	10%	9	20	45.00%
Test 2	20%	49	50	98.00%

Sum:	56.08%
Normalized Sum:	86.27%

Calculation

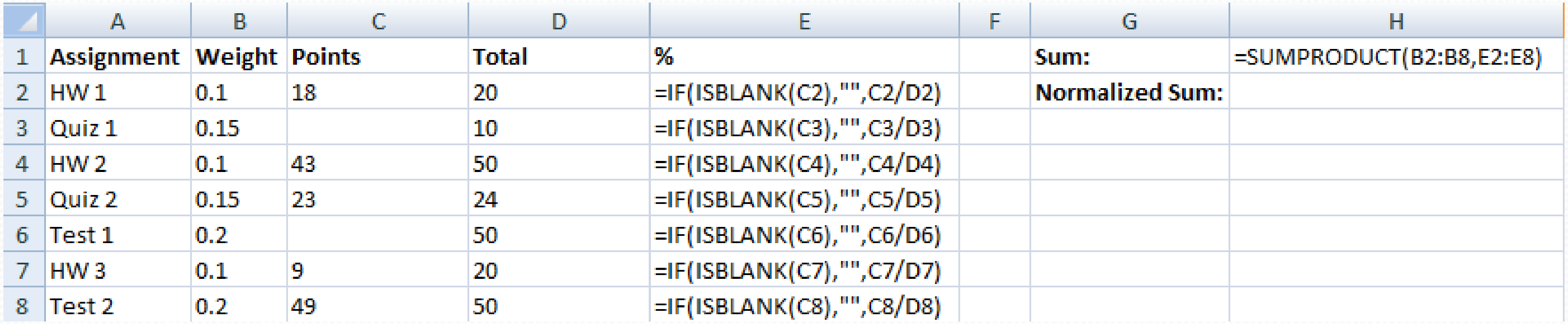
- 10% * 90% + 10% * 86% + 15% * 95.83% + 10% * 45% + 20% * 98% = 56.08%
- 56.08% / (100% - 15% - 20%) = 86.27%

Assignment	Weight	Points	Total	%
HW 1	10%	18	20	90.00%
Quiz 1	15%		10	
HW 2	10%	43	50	86.00%
Quiz 2	15%	23	24	95.83%
Test 1	20%		50	
HW 3	10%	9	20	45.00%
Test 2	20%	49	50	98.00%

Sum:	56.08%
Normalized Sum:	86.27%

=SUMPRODUCT(): cross product

- 10% * 90% + 10% * 86% + 15% * 95.83% + 10% * 45% + 20% * 98% = 56.08%
- Numbers in red represent the weights, multiplied by its corresponding grade %
- We multiply each weight by its corresponding %, and tally them: cross product of vectors in calculus



	A	B	C	D	E	F	G	H
1	**Assignment**	**Weight**	**Points**	**Total**	**%**		**Sum:**	=SUMPRODUCT(B2:B8,E2:E8)
2	HW 1	0.1	18	20	=IF(ISBLANK(C2),"",C2/D2)		**Normalized Sum:**	
3	Quiz 1	0.15		10	=IF(ISBLANK(C3),"",C3/D3)			
4	HW 2	0.1	43	50	=IF(ISBLANK(C4),"",C4/D4)			
5	Quiz 2	0.15	23	24	=IF(ISBLANK(C5),"",C5/D5)			
6	Test 1	0.2		50	=IF(ISBLANK(C6),"",C6/D6)			
7	HW 3	0.1	9	20	=IF(ISBLANK(C7),"",C7/D7)			
8	Test 2	0.2	49	50	=IF(ISBLANK(C8),"",C8/D8)			

Normalized Sum

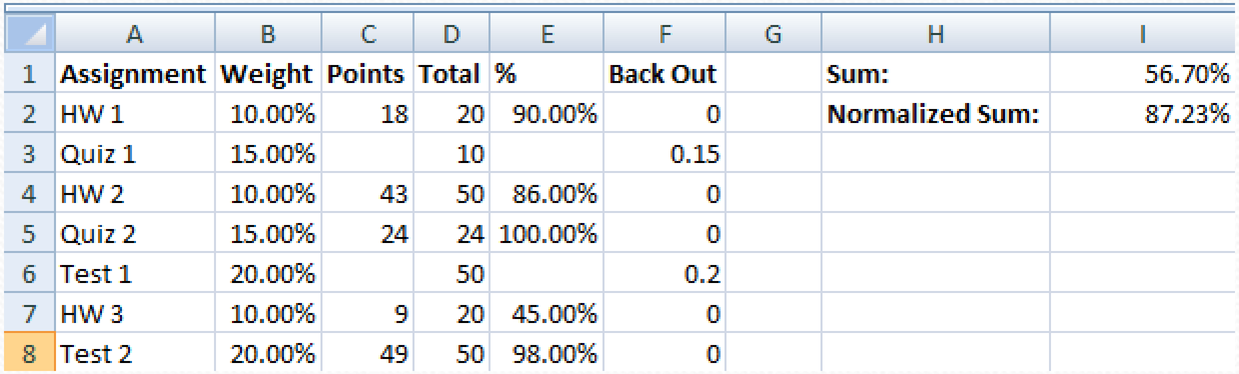
- 56.08% / (100% - 15% - 20%) = 86.27%
- If “Points” column is empty, take the “Weight” value of that row … tally this sum, to be subtracted later

Assignment	Weight	Points	Total	%
HW 1	10%	18	20	90.00%
Quiz 1	15%		10	
HW 2	10%	43	50	86.00%
Quiz 2	15%	23	24	95.83%
Test 1	20%		50	
HW 3	10%	9	20	45.00%
Test 2	20%	49	50	98.00%

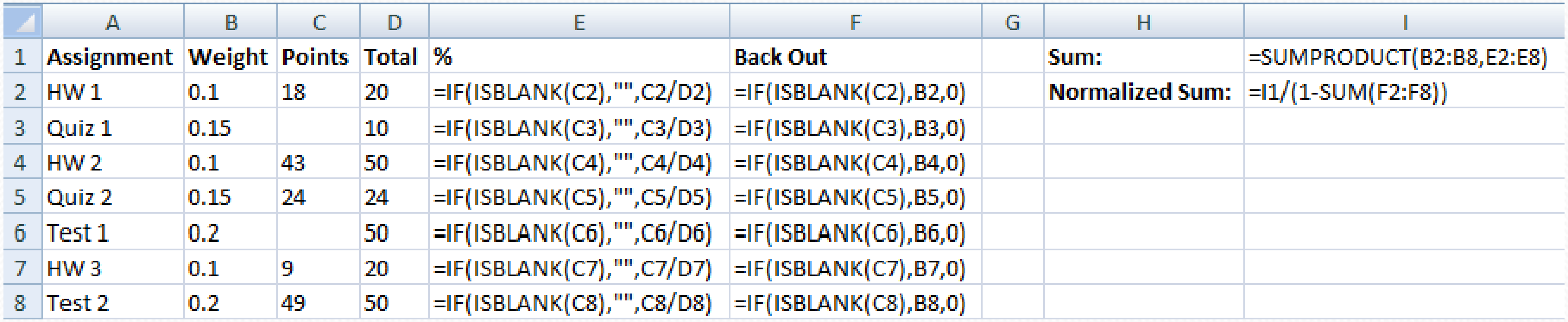
Sum:	56.08%
Normalized Sum:	86.27%

Normalized Sum

- Added Column F to display the weight of the assignment, if there is no grade for that



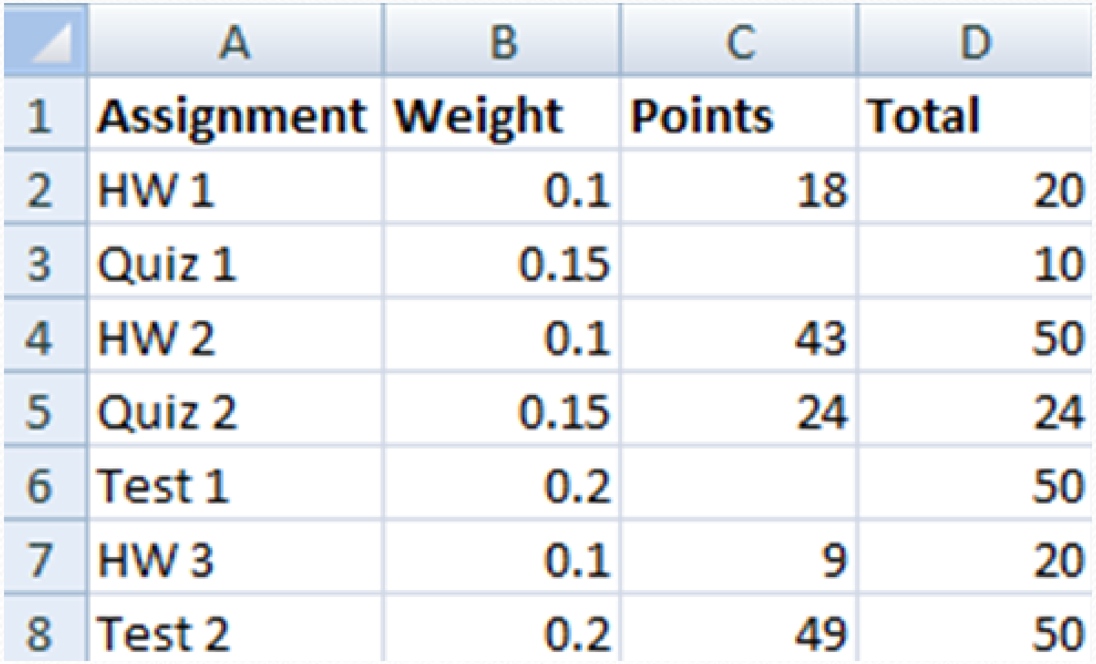
	A	B	C	D	E	F	G	H	I
1	Assignment	Weight	Points	Total	%	Back Out		Sum:	56.70%
2	HW 1	10.00%	18	20	90.00%	0		Normalized Sum:	87.23%
3	Quiz 1	15.00%		10		0.15			
4	HW 2	10.00%	43	50	86.00%	0			
5	Quiz 2	15.00%	24	24	100.00%	0			
6	Test 1	20.00%		50		0.2			
7	HW 3	10.00%	9	20	45.00%	0			
8	Test 2	20.00%	49	50	98.00%	0			



	A	B	C	D	E	F	G	H	I
1	Assignment	Weight	Points	Total	%	Back Out		Sum:	=SUMPRODUCT(B2:B8,E2:E8)
2	HW 1	0.1	18	20	=IF(ISBLANK(C2),"",C2/D2)	=IF(ISBLANK(C2),B2,0)		Normalized Sum:	=I1/(1-SUM(F2:F8))
3	Quiz 1	0.15		10	=IF(ISBLANK(C3),"",C3/D3)	=IF(ISBLANK(C3),B3,0)			
4	HW 2	0.1	43	50	=IF(ISBLANK(C4),"",C4/D4)	=IF(ISBLANK(C4),B4,0)			
5	Quiz 2	0.15	24	24	=IF(ISBLANK(C5),"",C5/D5)	=IF(ISBLANK(C5),B5,0)			
6	Test 1	0.2		50	=IF(ISBLANK(C6),"",C6/D6)	=IF(ISBLANK(C6),B6,0)			
7	HW 3	0.1	9	20	=IF(ISBLANK(C7),"",C7/D7)	=IF(ISBLANK(C7),B7,0)			
8	Test 2	0.2	49	50	=IF(ISBLANK(C8),"",C8/D8)	=IF(ISBLANK(C8),B8,0)			

Programming Approach

- We already have the algorithms down
- Just need to translate to VBA codes
- Given the following structure to begin with:



	A	B	C	D
1	**Assignment**	**Weight**	**Points**	**Total**
2	HW 1	0.1	18	20
3	Quiz 1	0.15		10
4	HW 2	0.1	43	50
5	Quiz 2	0.15	24	24
6	Test 1	0.2		50
7	HW 3	0.1	9	20
8	Test 2	0.2	49	50

Psuedocode

- Start from row 2
- Traverse down the rows, as long as there is an assignment
    - For each row, compute % if there is a score
    - Otherwise, keep track of the weight, to be backed out later
- Sum-product "Weights" and "%" for the sum
- Divide Sum by (100% - backed out weights) for normalized sum

The Code

```
Sub Tally()
Row = 2
BackedOut = 0
Sum = 0

While Cells(Row, 1) <> ""
If Cells(Row, 3) <> "" Then
Sum = Sum + Cells(Row, 2) * Cells(Row, 3) / Cells(Row, 4)
Else
BackedOut = BackedOut + Cells(Row, 2)
End If
Row = Row + 1
Wend

Cells(2, 9) = Sum / (1 - BackedOut)
End Sub
```

Set-Up

```
Sub Tally()
Row = 2
BackedOut = 0
Sum = 0
```

- Start from row 2
- Sum and weights to back out begin with 0

Loop Running Condition

While Cells(Row, 1) <> ""

- Cells(2,1) returns the value of row 2, column 1 … also known as cellA2
- <> means "not equal to"
- <> "" means not equal to blank cell … in another words, the cell contains something

Normal Entries

```
If Cells(Row, 3) <> "" Then
Sum = Sum + Cells(Row, 2) * Cells(Row, 3) / Cells(Row, 4)
```

- Recall: column 3 (column C) is the “Points” column
- If content of column C for each row is empty, then:
- Sum is the previous sum + (weight of assignment) * (points earned on assignment)/ (total number of points)

Empty Entries

```
Else
BackedOut = BackedOut + Cells(Row, 2)
End If
```

- Otherwise, the weight of that assignment entry (column 2) is tallied
- EndIf to denote the end of an If statement block

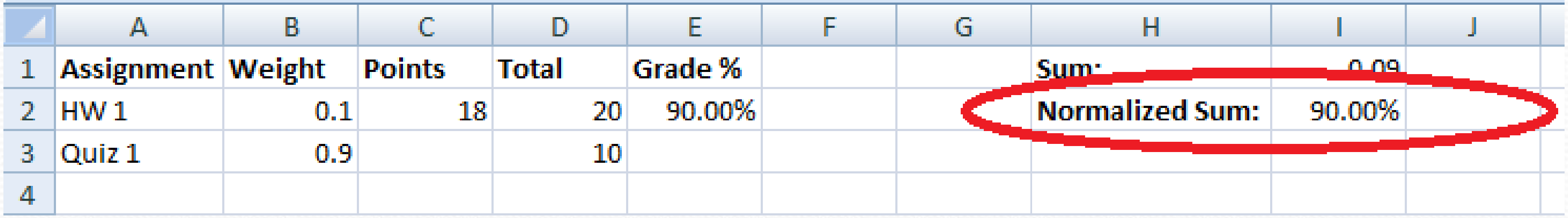
Traversing

```
Row = Row + 1
Wend
Cells(2, 9) = Sum / (1 - BackedOut)
Cells(2, 9).NumberFormat = "0.00%"
End Sub
```

- Move onto the next row
- Wend to denote the end of a While loop
- End Sub to denote the end of the Subroutine

Remarks

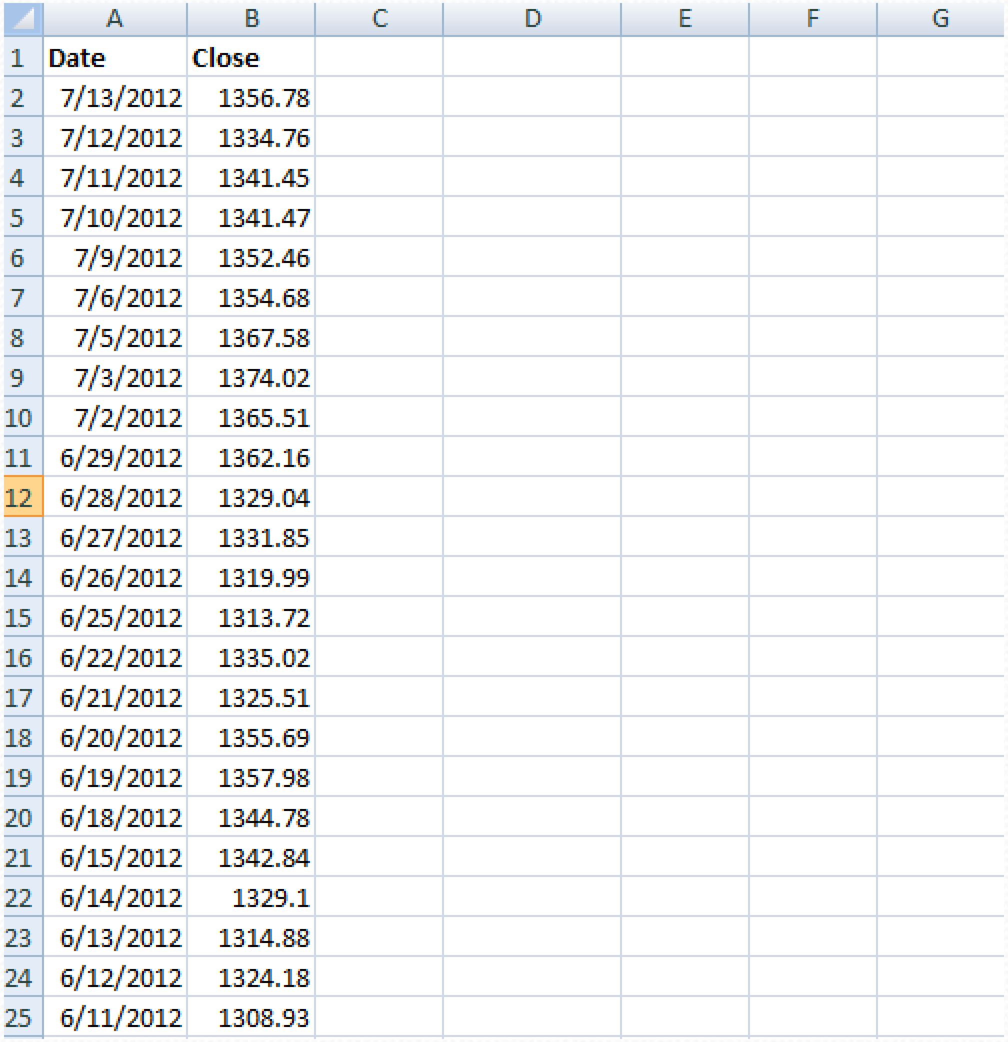
- This code can handle any number of assignment entries in this format
- Takes one click to run the program



	A	B	C	D	E	F	G	H	I	J
1	Assignment	Weight	Points	Total	Grade %			Sum:	0.09	
2	HW 1	0.1	18	20	90.00%			Normalized Sum:	90.00%	
3	Quiz 1	0.9		10						
4										

Case 5: Loop through Data

Refer to Excel Data



	A	B	C	D	E	F	G
1	Date	Close					
2	7/13/2012	1356.78					
3	7/12/2012	1334.76					
4	7/11/2012	1341.45					
5	7/10/2012	1341.47					
6	7/9/2012	1352.46					
7	7/6/2012	1354.68					
8	7/5/2012	1367.58					
9	7/3/2012	1374.02					
10	7/2/2012	1365.51					
11	6/29/2012	1362.16					
12	6/28/2012	1329.04					
13	6/27/2012	1331.85					
14	6/26/2012	1319.99					
15	6/25/2012	1313.72					
16	6/22/2012	1335.02					
17	6/21/2012	1325.51					
18	6/20/2012	1355.69					
19	6/19/2012	1357.98					
20	6/18/2012	1344.78					
21	6/15/2012	1342.84					
22	6/14/2012	1329.1					
23	6/13/2012	1314.88					
24	6/12/2012	1324.18					
25	6/11/2012	1308.93					

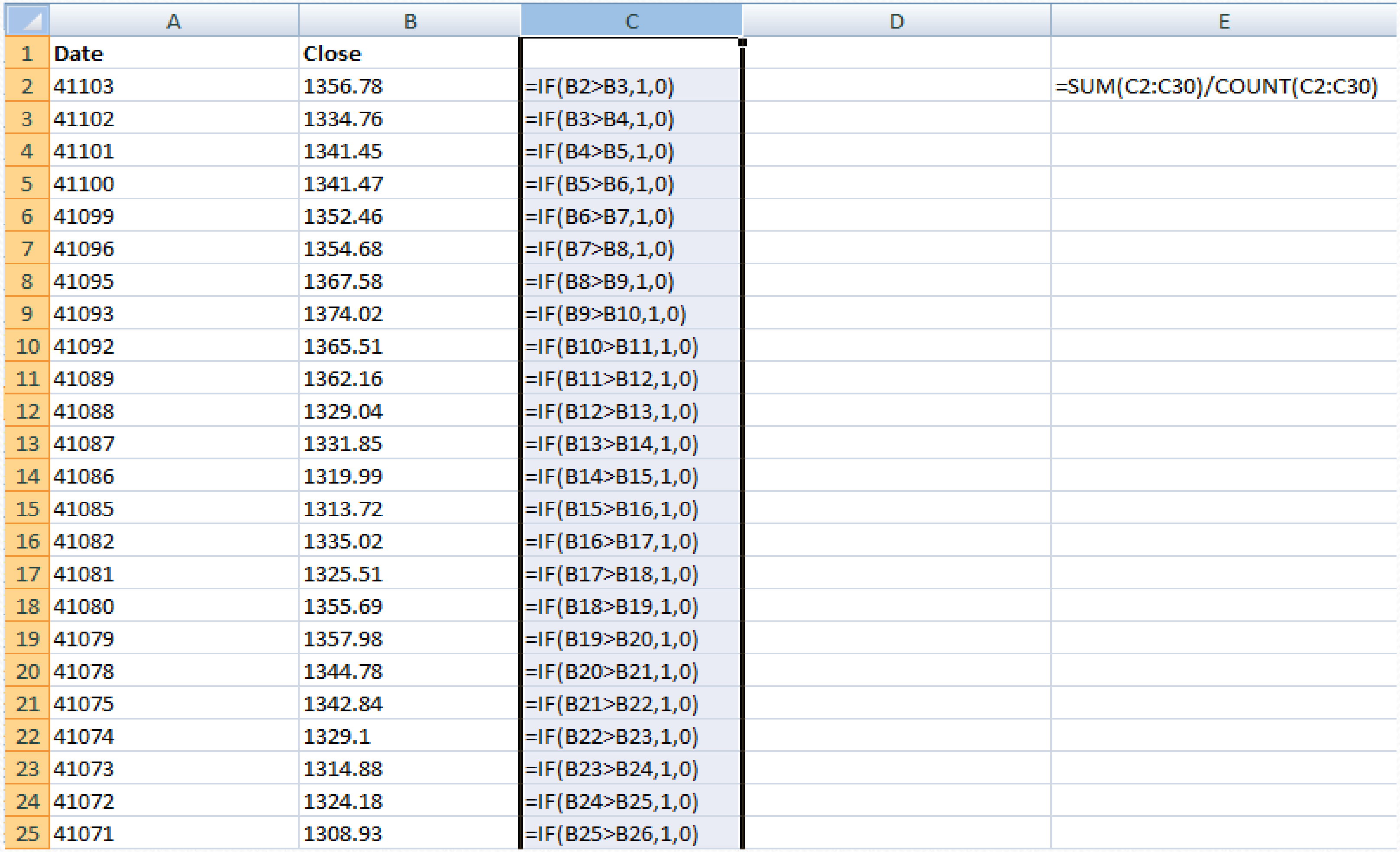
Task

- Give the percentage of days in which the stock index (S&P 500) closed above the previous day
- The last date on the table wouldn’t be factored
  - If there are 19 dates of raw data, there’ll only be 18 entries for this exercise of computing daily changes
- Accommodate the algorithm for future entries to be added below
- Perform the task without programming, and then with programming

Algorithm Reasoning

- In column C, return 1 if the value in same row's column B is greater than previous row's column B
- Otherwise, return 0
- Drag the formula down to the penultimate entry
- Sum up column C, divided by the number of entries in column C

No Programming



	A	B	C	D	E
1	**Date**	**Close**			
2	41103	1356.78	=IF(B2>B3,1,0)		=SUM(C2:C30)/COUNT(C2:C30)
3	41102	1334.76	=IF(B3>B4,1,0)		
4	41101	1341.45	=IF(B4>B5,1,0)		
5	41100	1341.47	=IF(B5>B6,1,0)		
6	41099	1352.46	=IF(B6>B7,1,0)		
7	41096	1354.68	=IF(B7>B8,1,0)		
8	41095	1367.58	=IF(B8>B9,1,0)		
9	41093	1374.02	=IF(B9>B10,1,0)		
10	41092	1365.51	=IF(B10>B11,1,0)		
11	41089	1362.16	=IF(B11>B12,1,0)		
12	41088	1329.04	=IF(B12>B13,1,0)		
13	41087	1331.85	=IF(B13>B14,1,0)		
14	41086	1319.99	=IF(B14>B15,1,0)		
15	41085	1313.72	=IF(B15>B16,1,0)		
16	41082	1335.02	=IF(B16>B17,1,0)		
17	41081	1325.51	=IF(B17>B18,1,0)		
18	41080	1355.69	=IF(B18>B19,1,0)		
19	41079	1357.98	=IF(B19>B20,1,0)		
20	41078	1344.78	=IF(B20>B21,1,0)		
21	41075	1342.84	=IF(B21>B22,1,0)		
22	41074	1329.1	=IF(B22>B23,1,0)		
23	41073	1314.88	=IF(B23>B24,1,0)		
24	41072	1324.18	=IF(B24>B25,1,0)		
25	41071	1308.93	=IF(B25>B26,1,0)		

Programming Reasoning

- For loop would not be appropriate, since the number of iterations isn’t known at first
- While loop or Do Until loop
- Similar reasoning as the non-programming approach

While Loop

```
Row = 2
Sum = 0
Total = 0
While Cells(Row,1) <> ""
If Cells(Row,2) > Cells(Row+1,2) Then
Sum = Sum + 1
End If
Total = Total + 1
Row = Row + 1
Wend
Cells(Row-1,3) = ""
Cells(2,5) = (Sum-1) / (Total-1)  ‘to discount the last entry
```

ActiveCell

- ActiveCell, as the name suggests, refers to the currently selected cell
- ActiveCell.Value returns the value of the current cell
- Say cell B4 is to be selected:
  - Cells(4,2).Select
  - By the way, Range("B4") and Cells(2,4) refers to the same
  - ActiveCell.Value now returns the value of cell B4

Offset

- Very important function used both in programming and functions
- [Original cell].Offset([rows], [columns])
  - Cells(2,2).Offset(1,1) returns Cells(3,3)
  - Cells(2,2).Offset(3,0) returns Cells(5,2)
  - Cells(2,2).Offset(-1,1) returns Cells(1,3)
- ActiveCell.Offset(1,0).Select
  - Select the cell one row beneath, in the same column
  - Potential for loops?

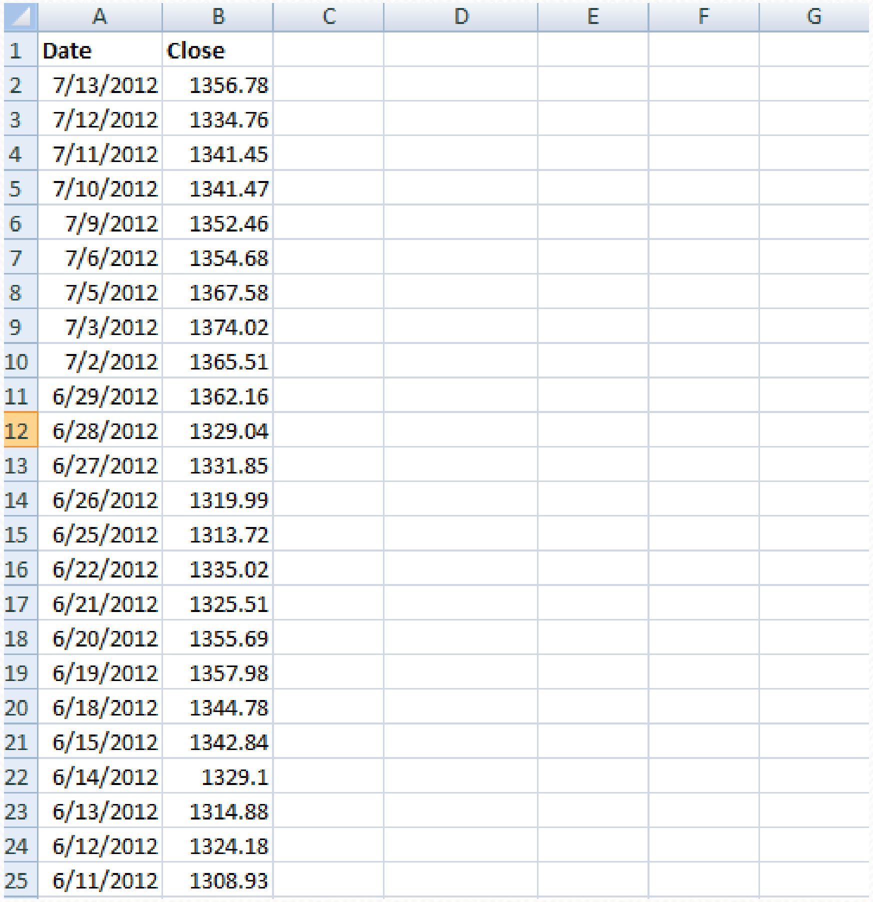
Offset Looping

- Given rows of entries, begin with cell A1 and loop down the rows, until there are no more entries
- No more the need to keep track of a “Row” variable

```
Range("A1").Select
Do Until IsEmpty(ActiveCell.Value)
‘[Can do something here for each row]
ActiveCell.Offset(1,0).Select
Loop
```

Try With This

- Use offset function in program



	A	B	C	D	E	F	G
1	Date	Close					
2	7/13/2012	1356.78					
3	7/12/2012	1334.76					
4	7/11/2012	1341.45					
5	7/10/2012	1341.47					
6	7/9/2012	1352.46					
7	7/6/2012	1354.68					
8	7/5/2012	1367.58					
9	7/3/2012	1374.02					
10	7/2/2012	1365.51					
11	6/29/2012	1362.16					
12	6/28/2012	1329.04					
13	6/27/2012	1331.85					
14	6/26/2012	1319.99					
15	6/25/2012	1313.72					
16	6/22/2012	1335.02					
17	6/21/2012	1325.51					
18	6/20/2012	1355.69					
19	6/19/2012	1357.98					
20	6/18/2012	1344.78					
21	6/15/2012	1342.84					
22	6/14/2012	1329.1					
23	6/13/2012	1314.88					
24	6/12/2012	1324.18					
25	6/11/2012	1308.93					

Programming

```
Range("B2").Select
Sum = 0
Total = 0
Do Until IsEmpty(ActiveCell.Value)
Total = Total + 1
If ActiveCell.Value > ActiveCell.Offset(1,0).Value Then
Sum = Sum + 1
EndIf
ActiveCell.Offset(1,0).Select
Loop
Range("E2").Value = (Sum – 1) / (Total – 1)
```

Tutorial 4: Recording Macro

Unlike Other Programming Lang.

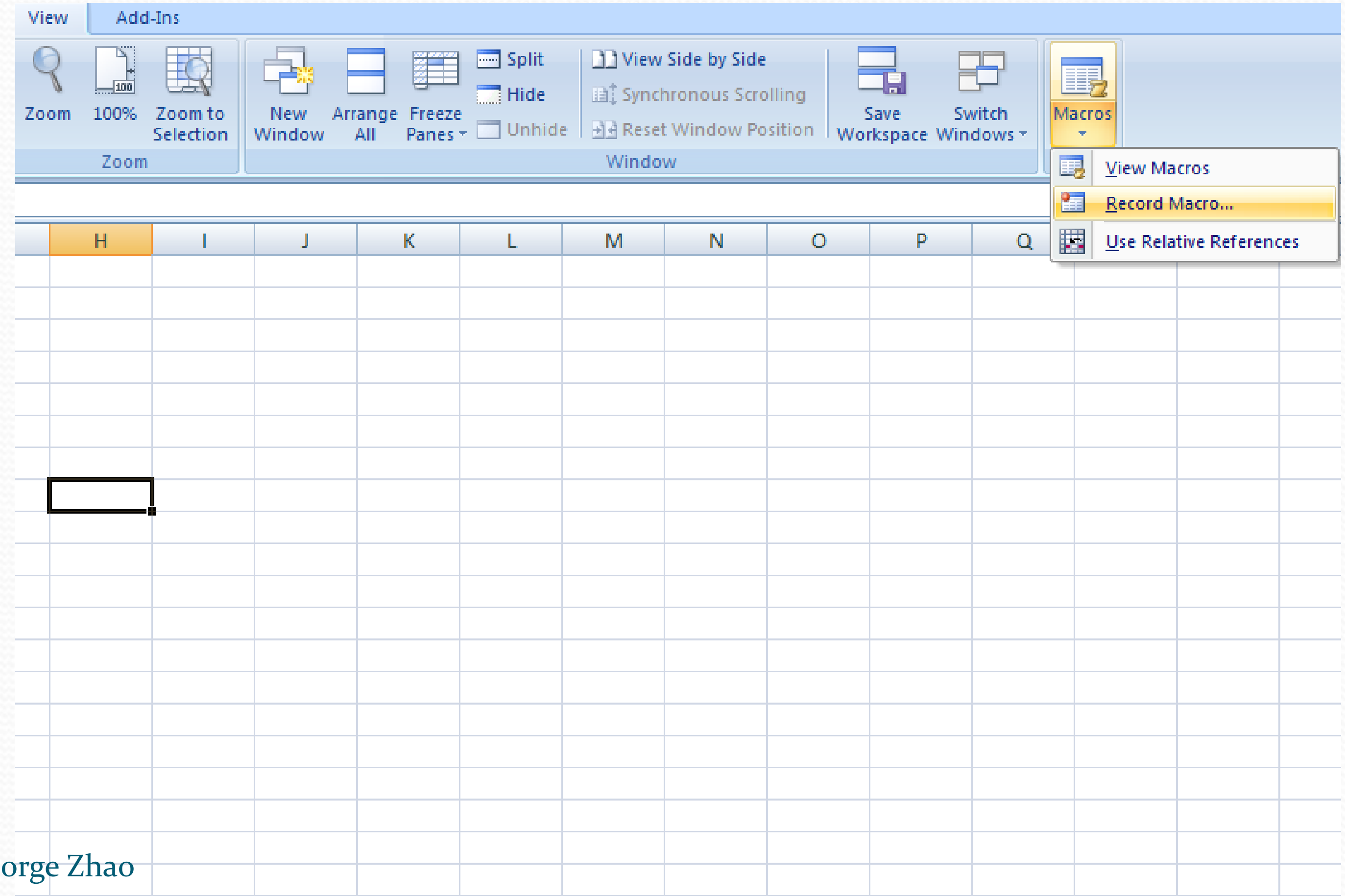
- Don’t need to know how to write all types of codes
  - Asking “why learn it then?”
- Excel can “record” actions performed by the user and give the code (macro) that, once upon run, would perform the same actions
  - But what about tweaking a mini detail?
- Challenge then becomes deciphering the codes, to adapt to similar but different scenarios
  - Therefore, still important to learn VBA

Sorting

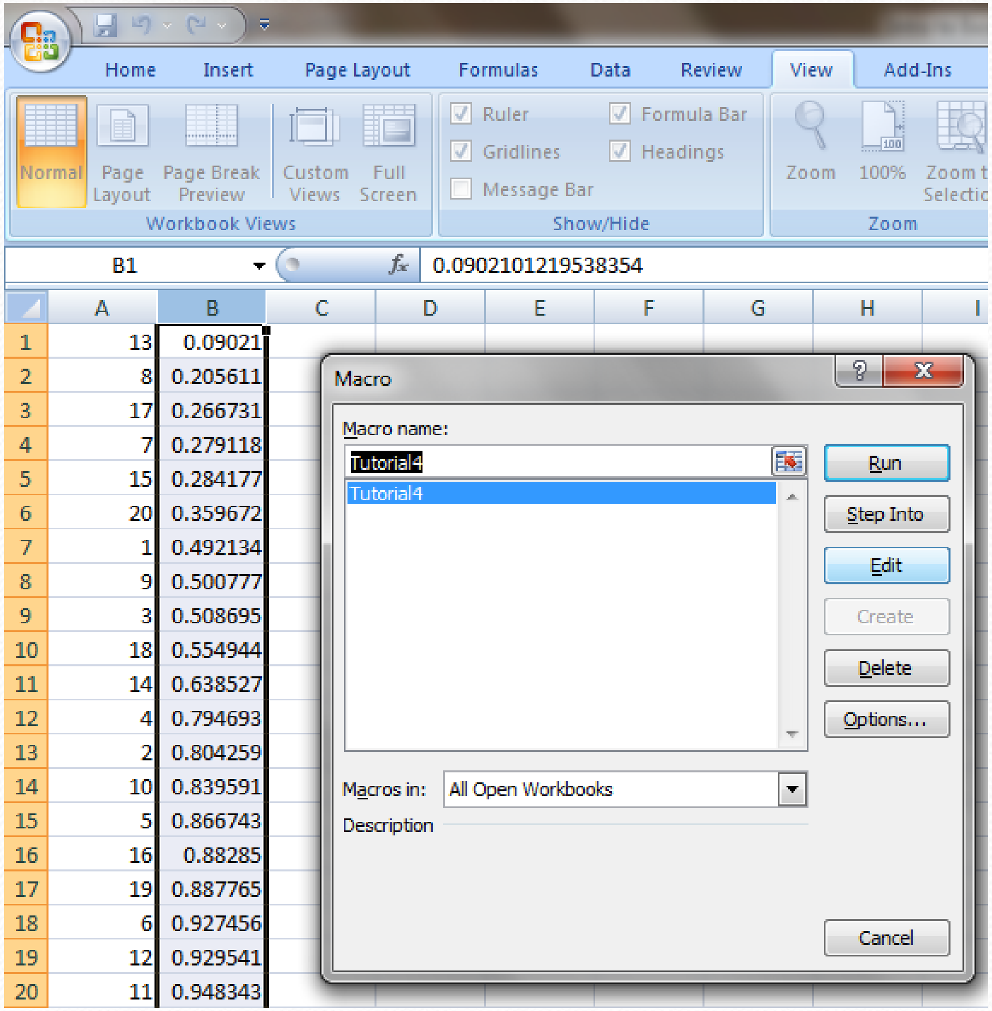
- Refer to Excel file
- Wish to sort by values in column B, smallest to largest, expanding the selection
- Easy to perform manually
- Record macro to see the syntax to automate the action
  - Record macro
  - Perform action manually
  - Stop macro
  - Read macro code

Record Macro

- Essentially subroutines



View Macro Codes



B1 f_x 0.0902101219538354

	A	B
1	13	0.09021
2	8	0.205611
3	17	0.266731
4	7	0.279118
5	15	0.284177
6	20	0.359672
7	1	0.492134
8	9	0.500777
9	3	0.508695
10	18	0.554944
11	14	0.638527
12	4	0.794693
13	2	0.804259
14	10	0.839591
15	5	0.866743
16	16	0.88285
17	19	0.887765
18	6	0.927456
19	12	0.929541
20	11	0.948343

Macro

Macro name: Tutorial4

Tutorial4

Macros in: All Open Workbooks

Description

Look at the Codes

```
Columns("B:B").Select
    ActiveWorkbook.Worksheets("Tutorial 4").Sort.SortFields.Clear
    ActiveWorkbook.Worksheets("Tutorial 4").Sort.SortFields.Add
     Key:=Range("B1") _
        , SortOn:=xlSortOnValues, Order:=xlAscending,
     DataOption:=xlSortNormal
    With ActiveWorkbook.Worksheets("Tutorial 4").Sort
        .SetRange Range("A1:B20")
        .Header = xlNo
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With
```

Pick Out Characteristics

```
Columns("B:B").Select
    ActiveWorkbook.Worksheets("Tutorial 4").Sort.SortFields.Clear
    ActiveWorkbook.Worksheets("Tutorial 4").Sort.SortFields.Add
     Key:=Range("B1") _
        , SortOn:=xlSortOnValues, Order:=xlAscending,
     DataOption:=xlSortNormal
    With ActiveWorkbook.Worksheets("Tutorial 4").Sort
        .SetRange Range("A1:B20")
        .Header = xlNo
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With
```

Expanded Exercise

- Suppose in a table of data, spanning from columns A to E, rows 1 to 100, sort by values in column A
- Replace references of column B with column A
- Range("A1:B20") becomes Range(“A1:E100”)

Tasks Recording Macro Can Teach

- Inserting row
- Hiding a sheet
- Formatting cell text to be italic, red, and value displayed as percentage
- Copying, cutting, pasting
- Moving selection to the top row of a table entry
- Defining a formula in a cell
  - RC format just like offset

RC Reference: Row / Column

- Let cell C3 be value of A1 squared
- In the worksheet, formula of C3 would be: =A1^2
  - This involves "hard-coding" the A1 in the formula
  - However, functions without treating A1 as hard-coded.
  - Select the 3x3 table of A1:C3 and move it anywhere … the formula updates accordingly
  - It's as if the formula of the bottom-right cell (while in C3) is more like: =OFFSET(C3,-2,-2)^2
  - Now no reference of A1 necessary
  - Both works fine

RC Reference: Row / Column

- In VBA, only the equivalent of =OFFSET() works
- Select cell A1 and record this macro:
  - Type 3 in A1
  - Type “=A1^2” in C3
  - Stop macro

```
 ActiveCell.FormulaR1C1 = "3"
 Range("C3").Select
ActiveCell.FormulaR1C1 = "=R[-2]C[-2]^2"
Range("C4").Select
```

Defining Function in VBA

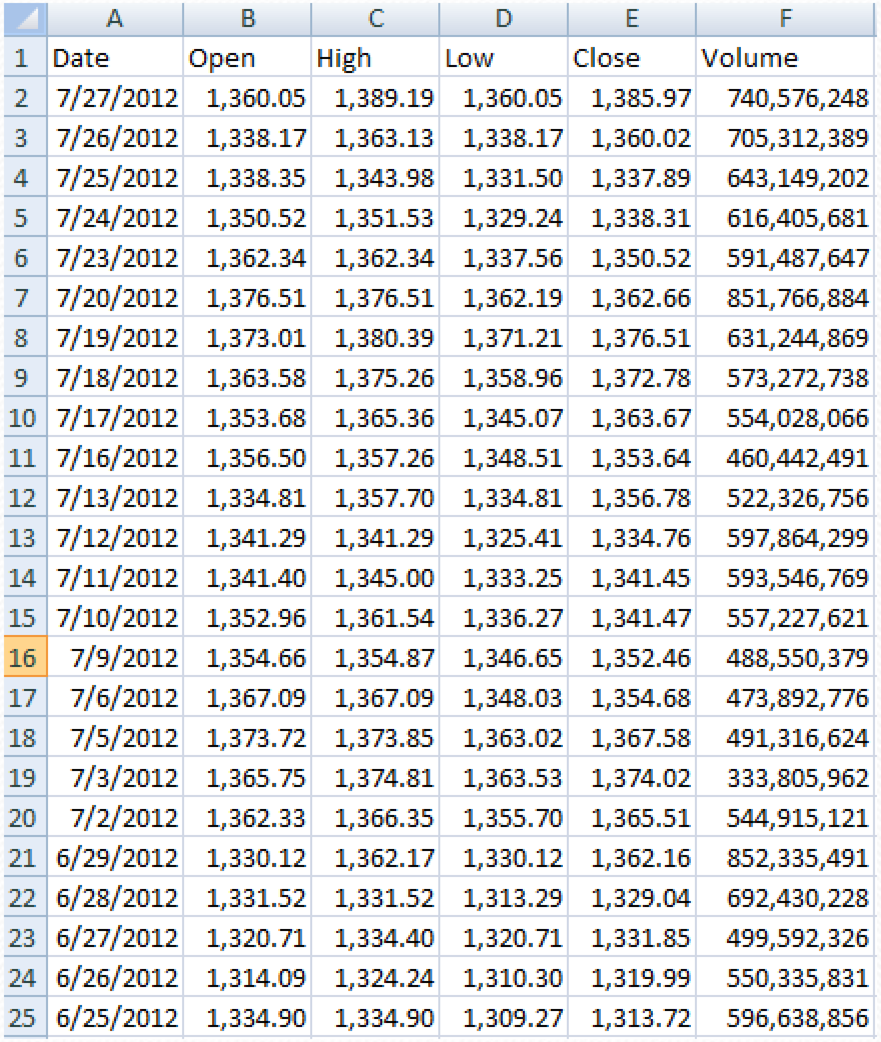
- Look at this line of code:
  - *ActiveCell.FormulaR1C1 = "=R[-2]C[-2]^2"*
- ActiveCell can be substituted for other equivalents:
  - Cells(2,3)
  - ActiveCell.Offset(1,2)
  - Range("F5:G9")
  - Range("F5:G9").Offset(10, 10)
- “=R[-2]C[-2]^2”
  - Offset from current cell, 2 rows above and 2 columns left

Note on Recording Macro

- Should not be substituted for learning code syntax
- Useful tool to learn the syntax
- Need to understand the algorithm of the codes, in order to know how and where to change the recorded codes to fit in the specific task
- This workshop will not be able to cover all types of scenarios in Excel, but knowing how to approach new problems is the most crucial component of problem solving

Case 6: Select, Pull, Display

Google Finance Data



	A	B	C	D	E	F
1	Date	Open	High	Low	Close	Volume
2	7/27/2012	1,360.05	1,389.19	1,360.05	1,385.97	740,576,248
3	7/26/2012	1,338.17	1,363.13	1,338.17	1,360.02	705,312,389
4	7/25/2012	1,338.35	1,343.98	1,331.50	1,337.89	643,149,202
5	7/24/2012	1,350.52	1,351.53	1,329.24	1,338.31	616,405,681
6	7/23/2012	1,362.34	1,362.34	1,337.56	1,350.52	591,487,647
7	7/20/2012	1,376.51	1,376.51	1,362.19	1,362.66	851,766,884
8	7/19/2012	1,373.01	1,380.39	1,371.21	1,376.51	631,244,869
9	7/18/2012	1,363.58	1,375.26	1,358.96	1,372.78	573,272,738
10	7/17/2012	1,353.68	1,365.36	1,345.07	1,363.67	554,028,066
11	7/16/2012	1,356.50	1,357.26	1,348.51	1,353.64	460,442,491
12	7/13/2012	1,334.81	1,357.70	1,334.81	1,356.78	522,326,756
13	7/12/2012	1,341.29	1,341.29	1,325.41	1,334.76	597,864,299
14	7/11/2012	1,341.40	1,345.00	1,333.25	1,341.45	593,546,769
15	7/10/2012	1,352.96	1,361.54	1,336.27	1,341.47	557,227,621
16	7/9/2012	1,354.66	1,354.87	1,346.65	1,352.46	488,550,379
17	7/6/2012	1,367.09	1,367.09	1,348.03	1,354.68	473,892,776
18	7/5/2012	1,373.72	1,373.85	1,363.02	1,367.58	491,316,624
19	7/3/2012	1,365.75	1,374.81	1,363.53	1,374.02	333,805,962
20	7/2/2012	1,362.33	1,366.35	1,355.70	1,365.51	544,915,121
21	6/29/2012	1,330.12	1,362.17	1,330.12	1,362.16	852,335,491
22	6/28/2012	1,331.52	1,331.52	1,313.29	1,329.04	692,430,228
23	6/27/2012	1,320.71	1,334.40	1,320.71	1,331.85	499,592,326
24	6/26/2012	1,314.09	1,324.24	1,310.30	1,319.99	550,335,831
25	6/25/2012	1,334.90	1,334.90	1,309.27	1,313.72	596,638,856

Task

- Create a customized data, whereby the user can enter and change 5 selected dates
- For those 5 dates, the customized data would pull all of the data to display from the original table
- For those 5 dates, make a bar graph, with the Opening and Closing values displayed on top of another for each date
- When the user changes the dates, the customized table and graphs should update automatically

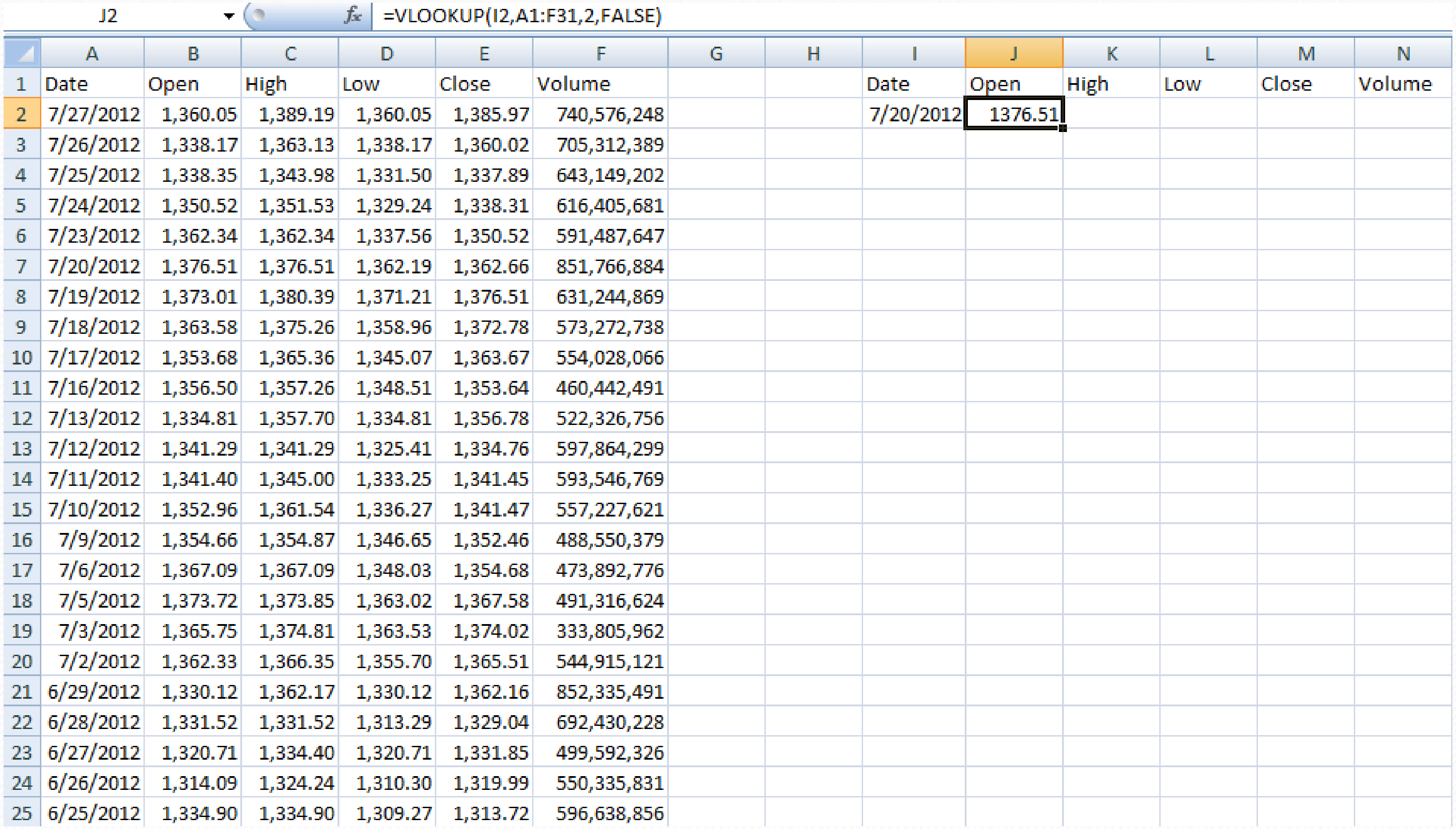
Algorithm

- Need a function that would take a value (user-inputted date) and search for that value within a bigger table
- =VLOOKUP() very useful when data is presented row-by-row
  - For example, if the look-up value is “7/20/2012”
  - Looks through the master table, looking for the row entry with “7/20/2012” in the first column
  - Function can return specific column entry
  - =HLOOKUP() not used as often, for when each data entry presented column-by-column

=VLOOKUP()

- User inputs 7/20/2012 … for Opening value:
  - =VLOOKUP(I2,A1:F31,2,FALSE)
  - I2 is the look-up value
  - A1:F31 is the master table to search from
  - 2 signifies return the 2nd column value from master table
  - FALSE means return only exact results (TRUE would yield approximate result when no exact result is found)

=VLOOKUP()



=VLOOKUP(I2,A1:F31,2,FALSE)

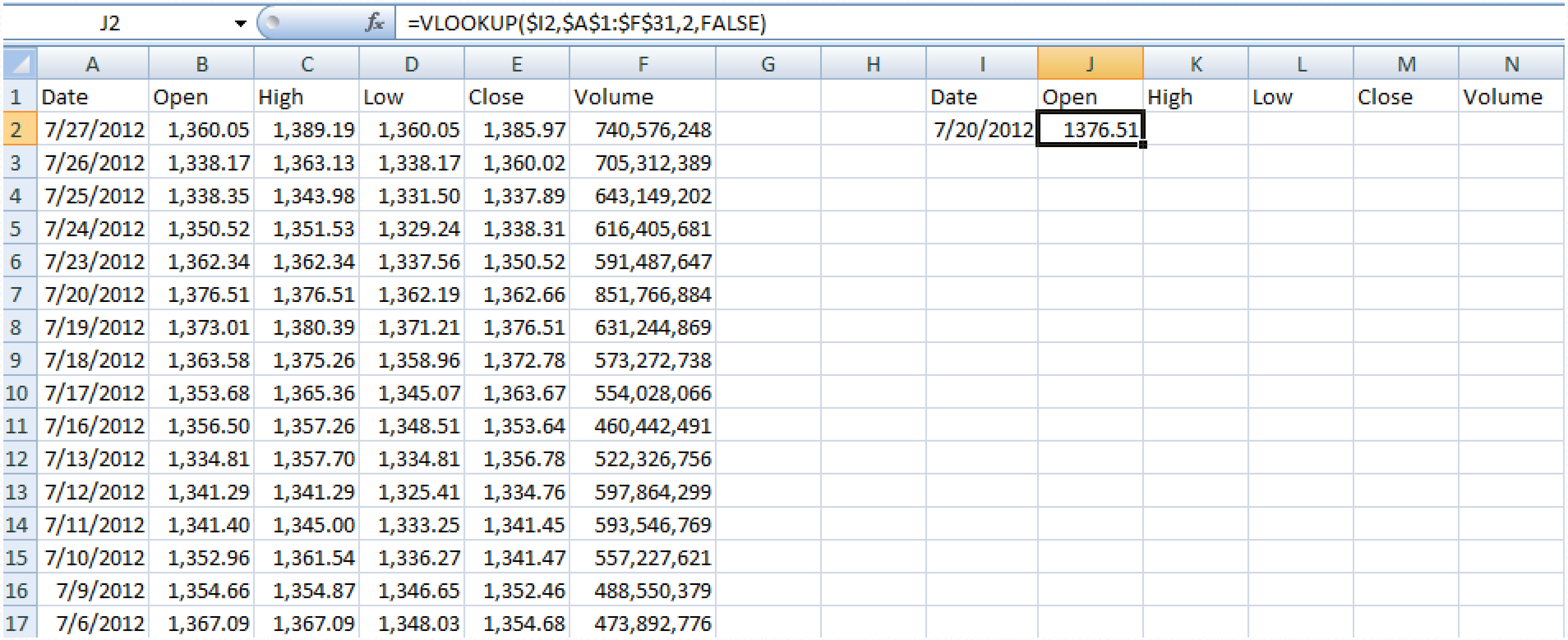
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Date	Open	High	Low	Close	Volume			Date	Open	High	Low	Close	Volume
2	7/27/2012	1,360.05	1,389.19	1,360.05	1,385.97	740,576,248			7/20/2012	1376.51				
3	7/26/2012	1,338.17	1,363.13	1,338.17	1,360.02	705,312,389								
4	7/25/2012	1,338.35	1,343.98	1,331.50	1,337.89	643,149,202								
5	7/24/2012	1,350.52	1,351.53	1,329.24	1,338.31	616,405,681								
6	7/23/2012	1,362.34	1,362.34	1,337.56	1,350.52	591,487,647								
7	7/20/2012	1,376.51	1,376.51	1,362.19	1,362.66	851,766,884								
8	7/19/2012	1,373.01	1,380.39	1,371.21	1,376.51	631,244,869								
9	7/18/2012	1,363.58	1,375.26	1,358.96	1,372.78	573,272,738								
10	7/17/2012	1,353.68	1,365.36	1,345.07	1,363.67	554,028,066								
11	7/16/2012	1,356.50	1,357.26	1,348.51	1,353.64	460,442,491								
12	7/13/2012	1,334.81	1,357.70	1,334.81	1,356.78	522,326,756								
13	7/12/2012	1,341.29	1,341.29	1,325.41	1,334.76	597,864,299								
14	7/11/2012	1,341.40	1,345.00	1,333.25	1,341.45	593,546,769								
15	7/10/2012	1,352.96	1,361.54	1,336.27	1,341.47	557,227,621								
16	7/9/2012	1,354.66	1,354.87	1,346.65	1,352.46	488,550,379								
17	7/6/2012	1,367.09	1,367.09	1,348.03	1,354.68	473,892,776								
18	7/5/2012	1,373.72	1,373.85	1,363.02	1,367.58	491,316,624								
19	7/3/2012	1,365.75	1,374.81	1,363.53	1,374.02	333,805,962								
20	7/2/2012	1,362.33	1,366.35	1,355.70	1,365.51	544,915,121								
21	6/29/2012	1,330.12	1,362.17	1,330.12	1,362.16	852,335,491								
22	6/28/2012	1,331.52	1,331.52	1,313.29	1,329.04	692,430,228								
23	6/27/2012	1,320.71	1,334.40	1,320.71	1,331.85	499,592,326								
24	6/26/2012	1,314.09	1,324.24	1,310.30	1,319.99	550,335,831								
25	6/25/2012	1,334.90	1,334.90	1,309.27	1,313.72	596,638,856								

Fixing Some References

- Here's what would happen if the formula is dragged down to the following row:
  - A1:F31 reference gets shifted
    - We don't want this, instead want fixed for all searches, across columns and rows
    - Fix both the row and column: $A$1:$F$31
  - Dragging across column, the look-up value reference gets shifted
    - We want to only fixed the column reference, not the row reference
    - Fix just the column: $I2

Refined Formula

- =VLOOKUP($I2,$A$1:$F$31,2,FALSE)

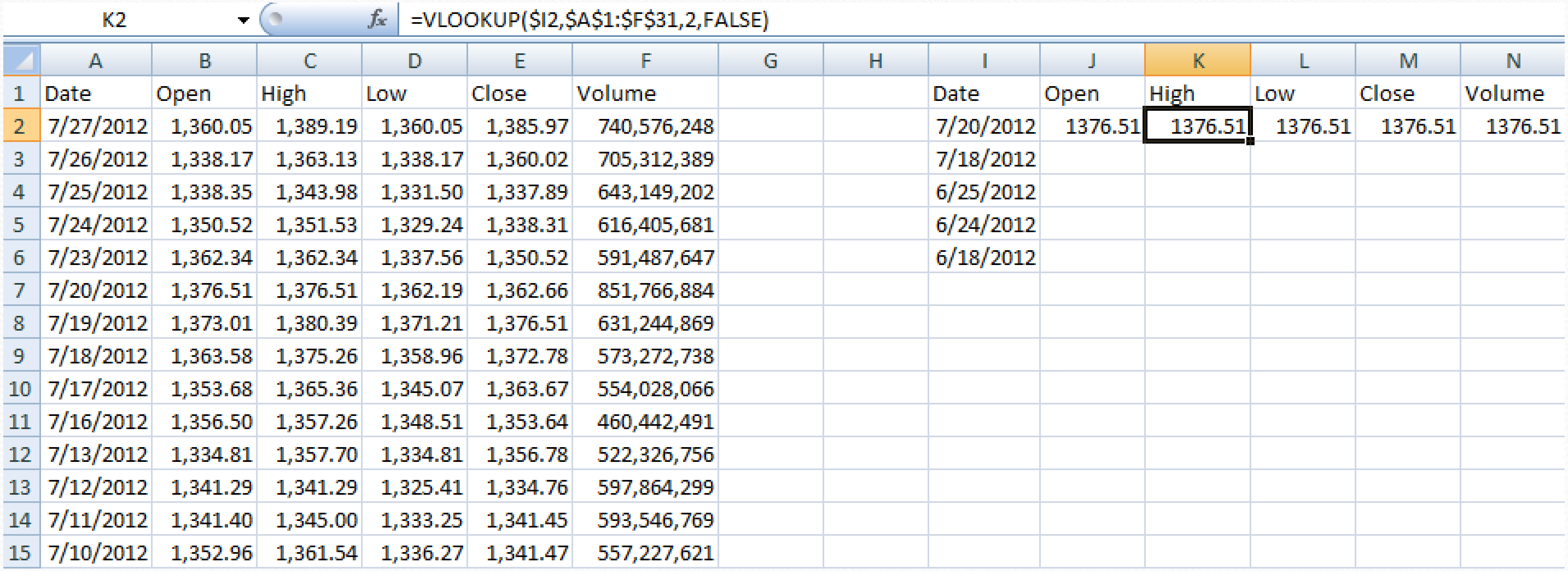


J2 =VLOOKUP($I2,$A$1:$F$31,2,FALSE)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Date	Open	High	Low	Close	Volume			Date	Open	High	Low	Close	Volume
2	7/27/2012	1,360.05	1,389.19	1,360.05	1,385.97	740,576,248			7/20/2012	1376.51				
3	7/26/2012	1,338.17	1,363.13	1,338.17	1,360.02	705,312,389								
4	7/25/2012	1,338.35	1,343.98	1,331.50	1,337.89	643,149,202								
5	7/24/2012	1,350.52	1,351.53	1,329.24	1,338.31	616,405,681								
6	7/23/2012	1,362.34	1,362.34	1,337.56	1,350.52	591,487,647								
7	7/20/2012	1,376.51	1,376.51	1,362.19	1,362.66	851,766,884								
8	7/19/2012	1,373.01	1,380.39	1,371.21	1,376.51	631,244,869								
9	7/18/2012	1,363.58	1,375.26	1,358.96	1,372.78	573,272,738								
10	7/17/2012	1,353.68	1,365.36	1,345.07	1,363.67	554,028,066								
11	7/16/2012	1,356.50	1,357.26	1,348.51	1,353.64	460,442,491								
12	7/13/2012	1,334.81	1,357.70	1,334.81	1,356.78	522,326,756								
13	7/12/2012	1,341.29	1,341.29	1,325.41	1,334.76	597,864,299								
14	7/11/2012	1,341.40	1,345.00	1,333.25	1,341.45	593,546,769								
15	7/10/2012	1,352.96	1,361.54	1,336.27	1,341.47	557,227,621								
16	7/9/2012	1,354.66	1,354.87	1,346.65	1,352.46	488,550,379								
17	7/6/2012	1,367.09	1,367.09	1,348.03	1,354.68	473,892,776								

Problem

- Upon dragging the formula to the right, nothing changes
- Need to change the 2 in the formula to 2, 3, etc.



K2 =VLOOKUP($I2,$A$1:$F$31,2,FALSE)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Date	Open	High	Low	Close	Volume			Date	Open	High	Low	Close	Volume
2	7/27/2012	1,360.05	1,389.19	1,360.05	1,385.97	740,576,248			7/20/2012	1376.51	1376.51	1376.51	1376.51	1376.51
3	7/26/2012	1,338.17	1,363.13	1,338.17	1,360.02	705,312,389			7/18/2012					
4	7/25/2012	1,338.35	1,343.98	1,331.50	1,337.89	643,149,202			6/25/2012					
5	7/24/2012	1,350.52	1,351.53	1,329.24	1,338.31	616,405,681			6/24/2012					
6	7/23/2012	1,362.34	1,362.34	1,337.56	1,350.52	591,487,647			6/18/2012					
7	7/20/2012	1,376.51	1,376.51	1,362.19	1,362.66	851,766,884								
8	7/19/2012	1,373.01	1,380.39	1,371.21	1,376.51	631,244,869								
9	7/18/2012	1,363.58	1,375.26	1,358.96	1,372.78	573,272,738								
10	7/17/2012	1,353.68	1,365.36	1,345.07	1,363.67	554,028,066								
11	7/16/2012	1,356.50	1,357.26	1,348.51	1,353.64	460,442,491								
12	7/13/2012	1,334.81	1,357.70	1,334.81	1,356.78	522,326,756								
13	7/12/2012	1,341.29	1,341.29	1,325.41	1,334.76	597,864,299								
14	7/11/2012	1,341.40	1,345.00	1,333.25	1,341.45	593,546,769								
15	7/10/2012	1,352.96	1,361.54	1,336.27	1,341.47	557,227,621								

Possible Solutions

- Manually change
- If the customized table will not be moved, can use a function that determines the current column value, and rearrange accordingly
- =COLUMN() returns the column number of the cell
  - Column A returns 1
  - Column B returns 2
  - Etc.

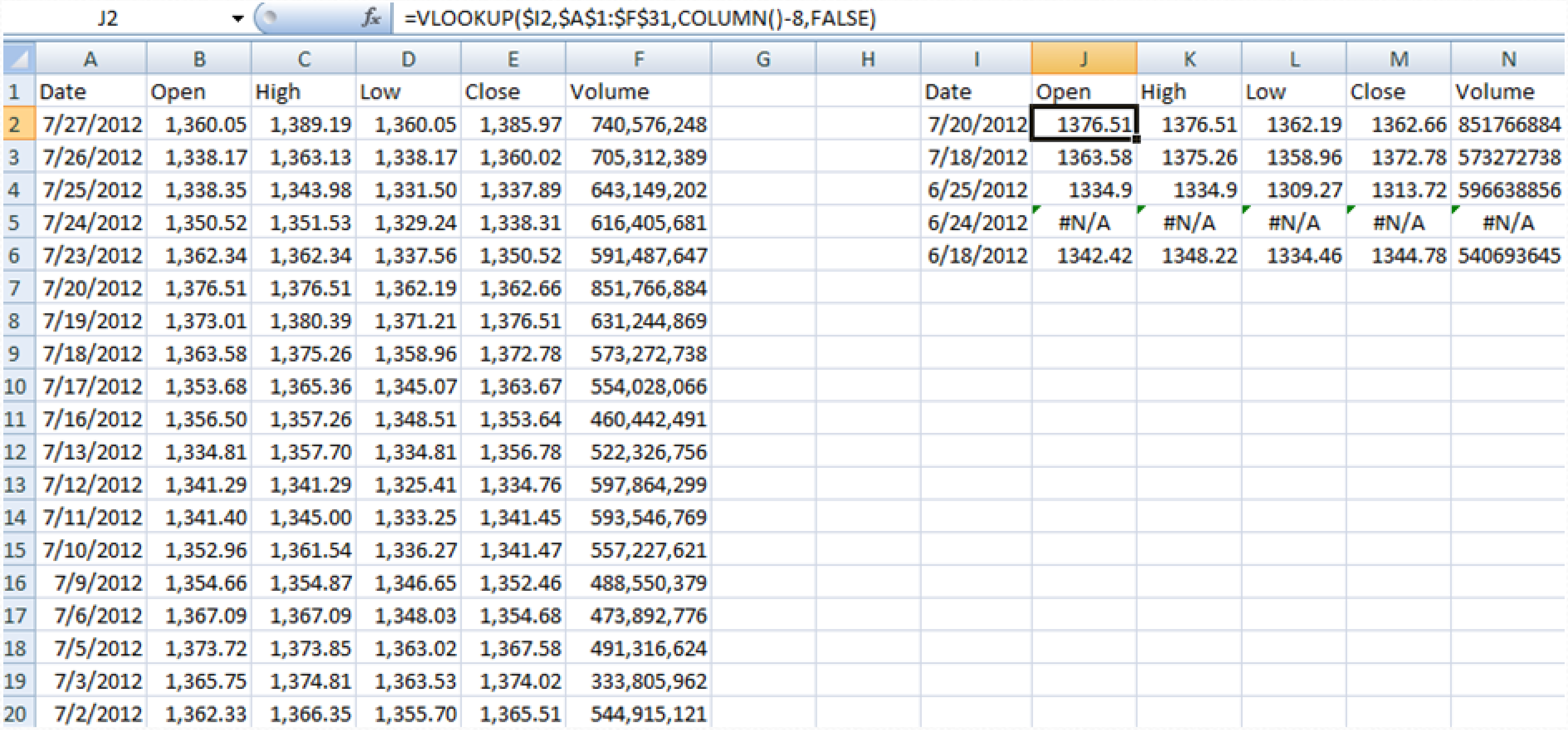
As It Stands

- Formula in J2 would’ve been:
  - =VLOOKUP($I2,$A$1:$F$31,**2**,FALSE)
- Formula in K2 would’ve been:
  - =VLOOKUP($I2,$A$1:$F$31,**3**,FALSE)
- Formula in L2 would’ve been:
  - =VLOOKUP($I2,$A$1:$F$31,**4**,FALSE)
- Formula in M2 would’ve been:
  - =VLOOKUP($I2,$A$1:$F$31,**5**,FALSE)
- Formula in N2 would’ve been:
  - =VLOOKUP($I2,$A$1:$F$31,**6**,FALSE)

Use =COLUMN()

- Column J is really column 10
- Column K is really column 11
- Column L is really column 12
- Column M is really column 13
- Column N is really column 14
- In all cases, COLUMN()-8 would give the appropriate number to incorporate into the formula

Refined Formula



J2 =VLOOKUP($I2,$A$1:$F$31,COLUMN()-8,FALSE)

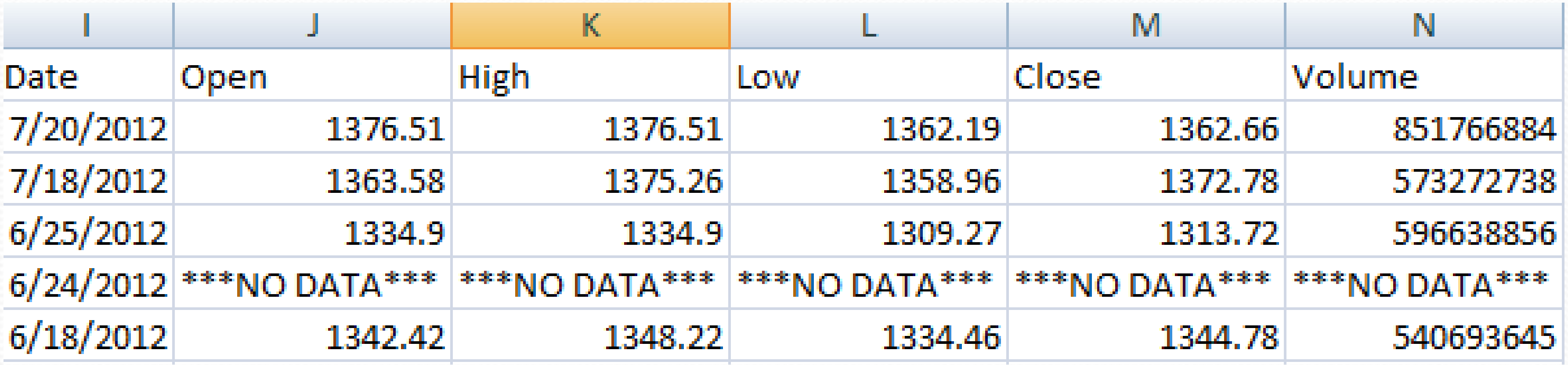
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Date	Open	High	Low	Close	Volume			Date	Open	High	Low	Close	Volume
2	7/27/2012	1,360.05	1,389.19	1,360.05	1,385.97	740,576,248			7/20/2012	1376.51	1376.51	1362.19	1362.66	851766884
3	7/26/2012	1,338.17	1,363.13	1,338.17	1,360.02	705,312,389			7/18/2012	1363.58	1375.26	1358.96	1372.78	573272738
4	7/25/2012	1,338.35	1,343.98	1,331.50	1,337.89	643,149,202			6/25/2012	1334.9	1334.9	1309.27	1313.72	596638856
5	7/24/2012	1,350.52	1,351.53	1,329.24	1,338.31	616,405,681			6/24/2012	#N/A	#N/A	#N/A	#N/A	#N/A
6	7/23/2012	1,362.34	1,362.34	1,337.56	1,350.52	591,487,647			6/18/2012	1342.42	1348.22	1334.46	1344.78	540693645
7	7/20/2012	1,376.51	1,376.51	1,362.19	1,362.66	851,766,884								
8	7/19/2012	1,373.01	1,380.39	1,371.21	1,376.51	631,244,869								
9	7/18/2012	1,363.58	1,375.26	1,358.96	1,372.78	573,272,738								
10	7/17/2012	1,353.68	1,365.36	1,345.07	1,363.67	554,028,066								
11	7/16/2012	1,356.50	1,357.26	1,348.51	1,353.64	460,442,491								
12	7/13/2012	1,334.81	1,357.70	1,334.81	1,356.78	522,326,756								
13	7/12/2012	1,341.29	1,341.29	1,325.41	1,334.76	597,864,299								
14	7/11/2012	1,341.40	1,345.00	1,333.25	1,341.45	593,546,769								
15	7/10/2012	1,352.96	1,361.54	1,336.27	1,341.47	557,227,621								
16	7/9/2012	1,354.66	1,354.87	1,346.65	1,352.46	488,550,379								
17	7/6/2012	1,367.09	1,367.09	1,348.03	1,354.68	473,892,776								
18	7/5/2012	1,373.72	1,373.85	1,363.02	1,367.58	491,316,624								
19	7/3/2012	1,365.75	1,374.81	1,363.53	1,374.02	333,805,962								
20	7/2/2012	1,362.33	1,366.35	1,355.70	1,365.51	544,915,121								

Improvement

- =VLOOKUP() returns #N/A error is the look-up value is not found in the original table
- We can error trap using =IFERROR() function
- =IFERROR([value if no error], [value if error])
- Suppose we want to display “***NO DATA***” if there is no data
- In cell J2, to be dragged in both directions:
  - =IFERROR(VLOOKUP($I2,$A$1:$F$31,COLUMN()-8,FALSE),"***NO DATA***")

Refined Table

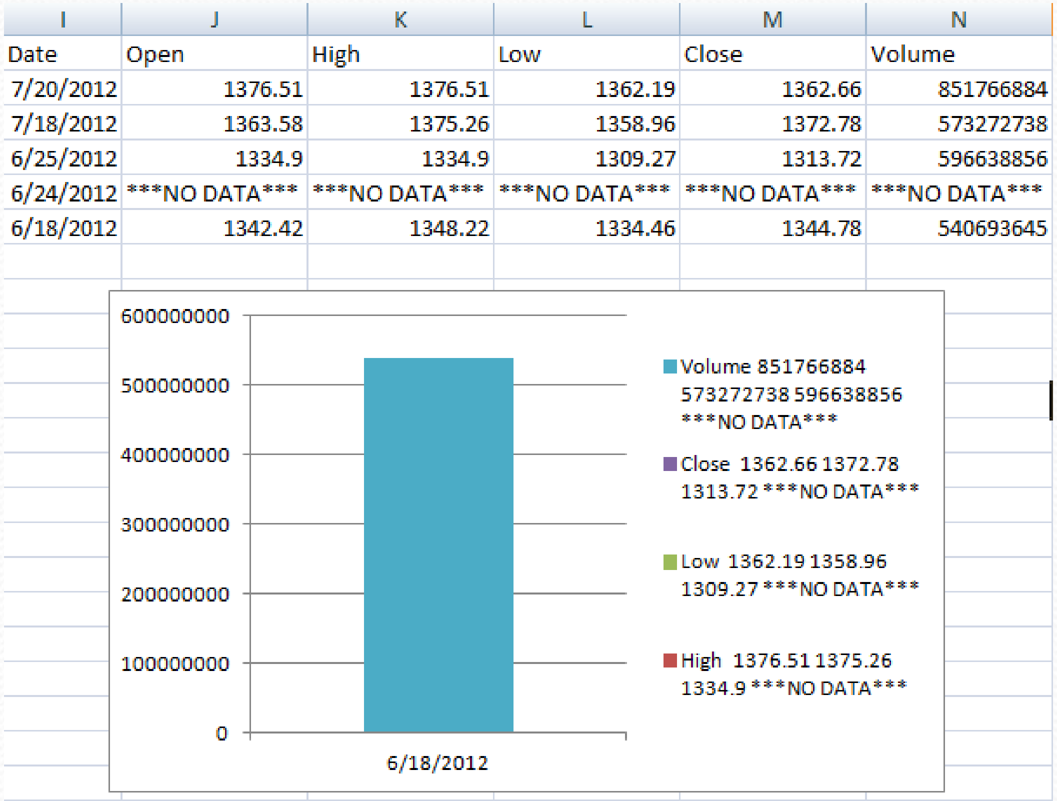
- Manually efficient
- Dynamic



I	J	K	L	M	N
Date	Open	High	Low	Close	Volume
7/20/2012	1376.51	1376.51	1362.19	1362.66	851766884
7/18/2012	1363.58	1375.26	1358.96	1372.78	573272738
6/25/2012	1334.9	1334.9	1309.27	1313.72	596638856
6/24/2012	***NO DATA***	***NO DATA***	***NO DATA***	***NO DATA***	***NO DATA***
6/18/2012	1342.42	1348.22	1334.46	1344.78	540693645

Bar Chart

- Select the customized table, and choose bar graph
- Horrendous output, nothing like what we're looking for

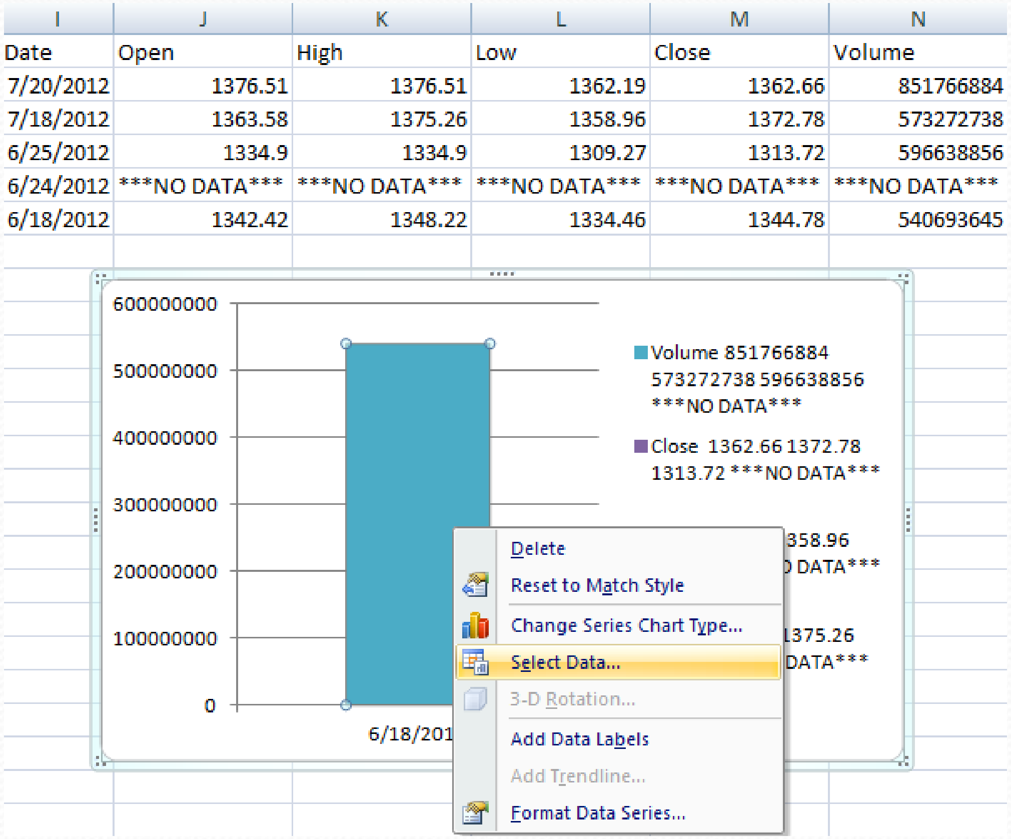


I	J	K	L	M	N
Date	Open	High	Low	Close	Volume
7/20/2012	1376.51	1376.51	1362.19	1362.66	851766884
7/18/2012	1363.58	1375.26	1358.96	1372.78	573272738
6/25/2012	1334.9	1334.9	1309.27	1313.72	596638856
6/24/2012	***NO DATA***	***NO DATA***	***NO DATA***	***NO DATA***	***NO DATA***
6/18/2012	1342.42	1348.22	1334.46	1344.78	540693645

600000000
500000000
400000000
300000000
200000000
100000000
0
6/18/2012
🟦 Volume 851766884 573272738 596638856 ***NO DATA***
🟪 Close 1362.66 1372.78 1313.72 ***NO DATA***
🟩 Low 1362.19 1358.96 1309.27 ***NO DATA***
🟥 High 1376.51 1375.26 1334.9 ***NO DATA***

Don’t Worry

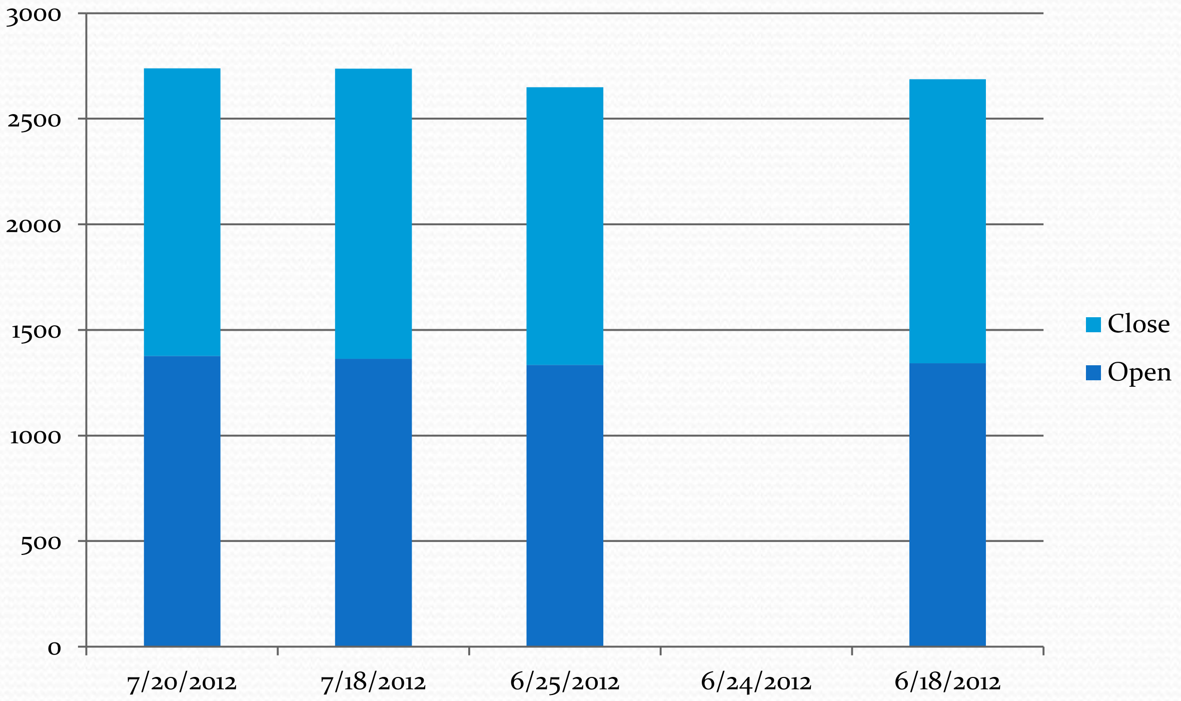
- We can manipulate a lot about the chart
- Right click > Select Data



I	J	K	L	M	N
Date	Open	High	Low	Close	Volume
7/20/2012	1376.51	1376.51	1362.19	1362.66	851766884
7/18/2012	1363.58	1375.26	1358.96	1372.78	573272738
6/25/2012	1334.9	1334.9	1309.27	1313.72	596638856
6/24/2012	***NO DATA***	***NO DATA***	***NO DATA***	***NO DATA***	***NO DATA***
6/18/2012	1342.42	1348.22	1334.46	1344.78	540693645

End Result

- Series
- Categories



Series and Categories

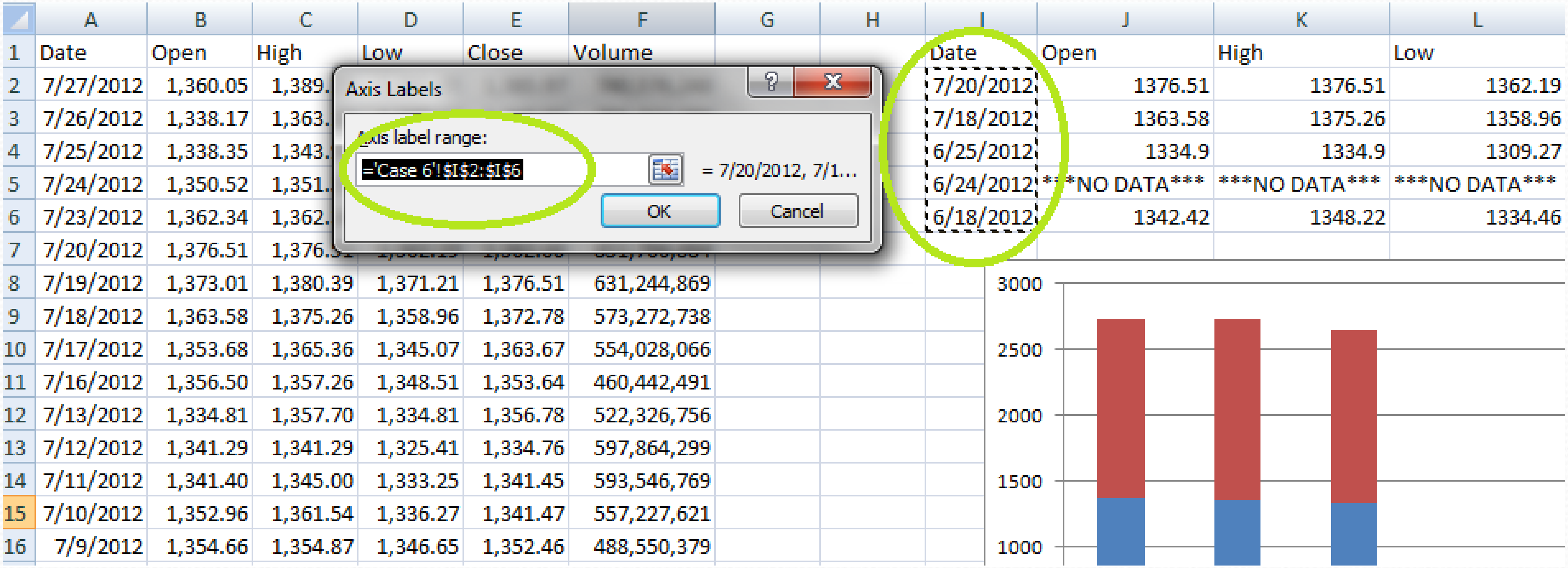
- Categories: different entries of similar types
  - Date 1, date 2, etc.
  - Product 1, product 2, etc.
  - Each of the 5 dates selected in the customized table
- Series: different types of values for each entry
  - High temp and low temp for each day
  - Number of hits and number of homeruns for each baseball player
  - Opening Price and Closing Price for each of the 5 days

Series

- Series 1 name: cell containing "Open"
- Series 1 value: cell range containing opening prices
- Series 2 name: cell containing "Closing"
- Series 2 value: cell range containing closing prices

Category

- Select the 5 dates



	A	B	C	D	E	F
1	Date	Open	High	Low	Close	Volume
2	7/27/2012	1,360.05	1,389.			
3	7/26/2012	1,338.17	1,363.			
4	7/25/2012	1,338.35	1,343.			
5	7/24/2012	1,350.52	1,351.			
6	7/23/2012	1,362.34	1,362.			
7	7/20/2012	1,376.51	1,376.5			
8	7/19/2012	1,373.01	1,380.39	1,371.21	1,376.51	631,244,869
9	7/18/2012	1,363.58	1,375.26	1,358.96	1,372.78	573,272,738
10	7/17/2012	1,353.68	1,365.36	1,345.07	1,363.67	554,028,066
11	7/16/2012	1,356.50	1,357.26	1,348.51	1,353.64	460,442,491
12	7/13/2012	1,334.81	1,357.70	1,334.81	1,356.78	522,326,756
13	7/12/2012	1,341.29	1,341.29	1,325.41	1,334.76	597,864,299
14	7/11/2012	1,341.40	1,345.00	1,333.25	1,341.45	593,546,769
15	7/10/2012	1,352.96	1,361.54	1,336.27	1,341.47	557,227,621
16	7/9/2012	1,354.66	1,354.87	1,346.65	1,352.46	488,550,379

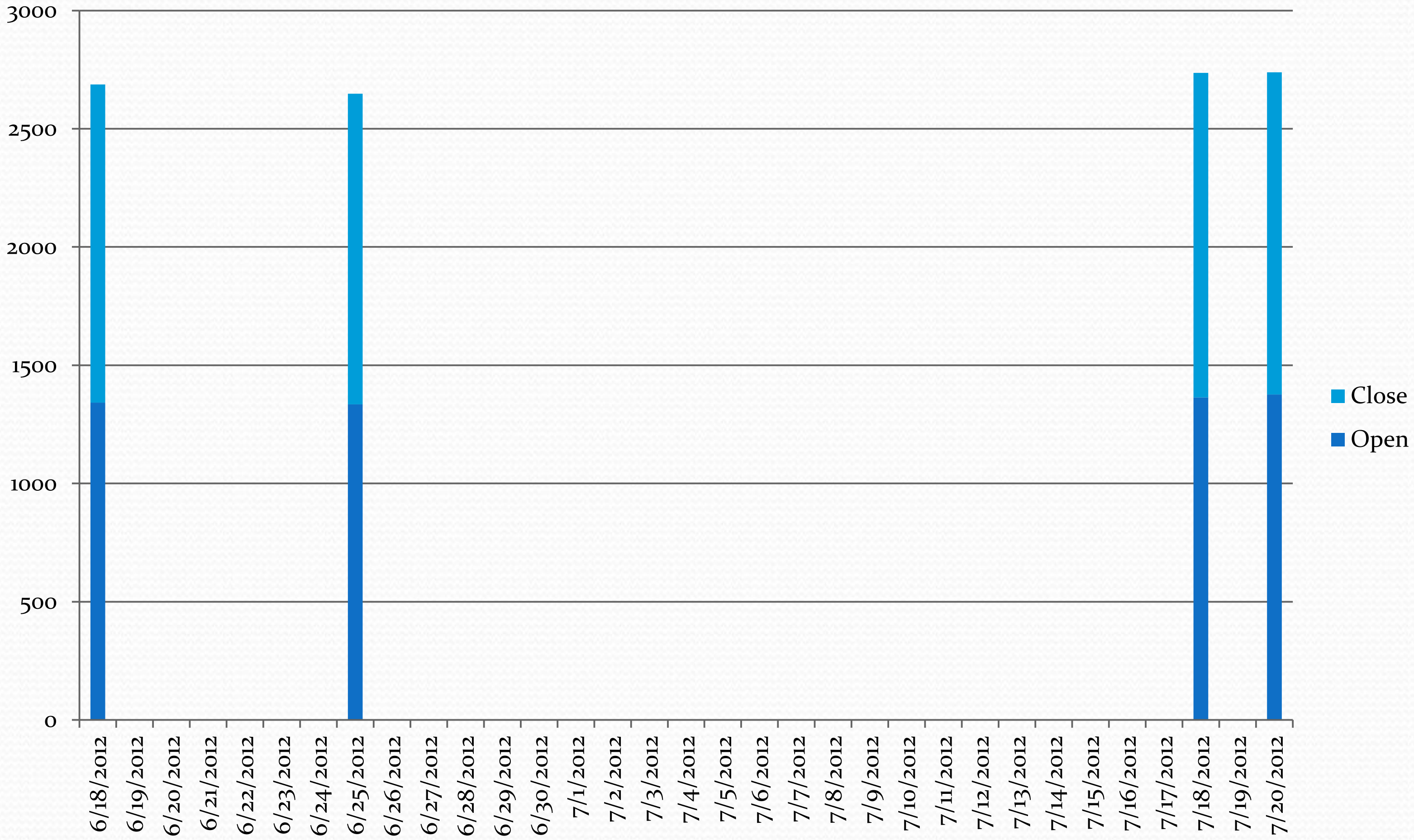
	I	J	K	L
1	Date	Open	High	Low
2	7/20/2012	1376.51	1376.51	1362.19
3	7/18/2012	1363.58	1375.26	1358.96
4	6/25/2012	1334.9	1334.9	1309.27
5	6/24/2012	***NO DATA***	***NO DATA***	***NO DATA***
6	6/18/2012	1342.42	1348.22	1334.46

Axis Labels

Axis label range:

='Case 6'!$I$2:$I$6 = 7/20/2012, 7/1...

Probably See This Initially

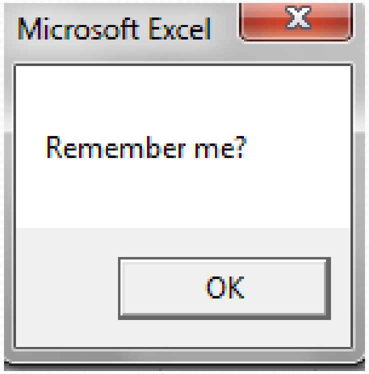


The Problem

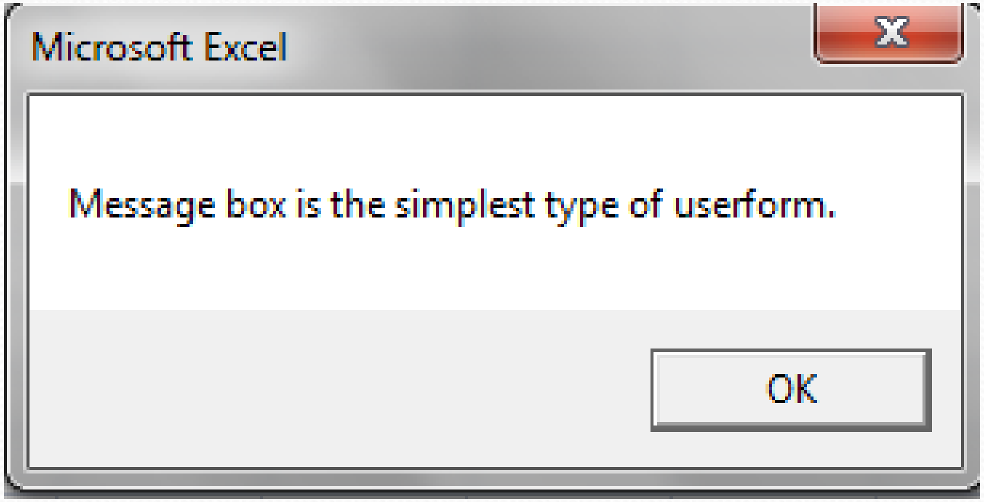
- Axis is arranged numerically, treating the dates on a continuous spectrum
- We want discrete spectrum, treating the dates not as numerical, but as text
- Luckily, there's feature for that
  - Right click axis > Format Axis > Axis Type: Text Axis

Tutorial 5: Userform

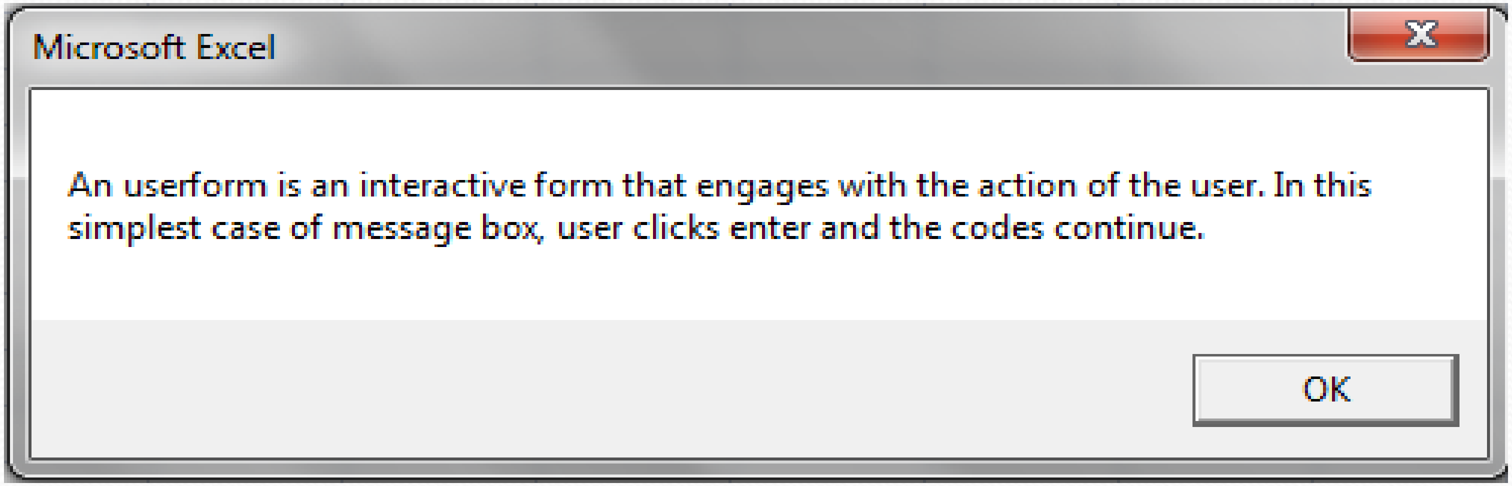
You’re Seen Some Already



Remember me?



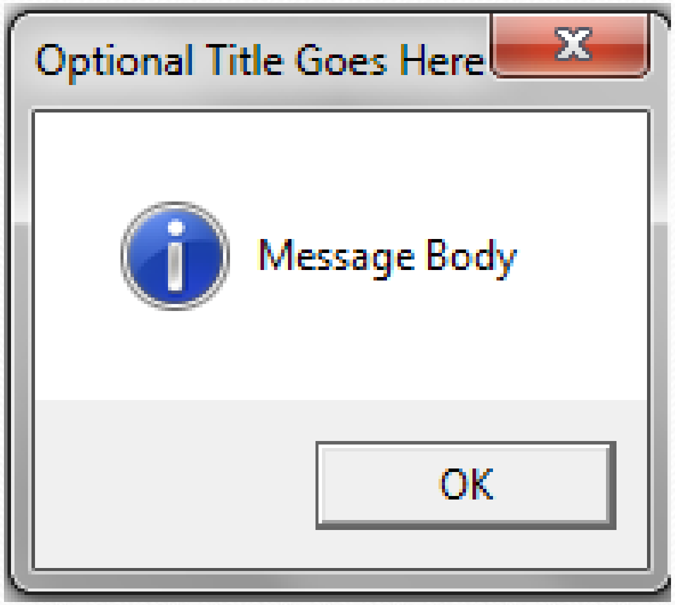
Message box is the simplest type of userform.



An userform is an interactive form that engages with the action of the user. In this simplest case of message box, user clicks enter and the codes continue.

Message Box

- So far, we've worked with only the simplest type of message box: user can only click "Okay"
- *MsgBox "Message Body", vbInformation, "Optional Title Goes Here"*

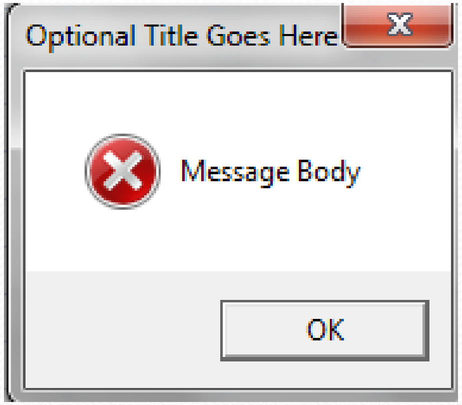


Optional Title Goes Here

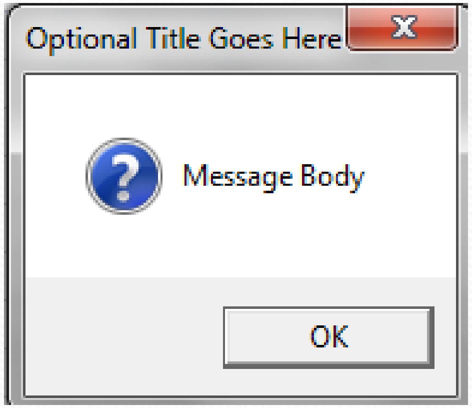
Message Body

Message Box Icon Displays

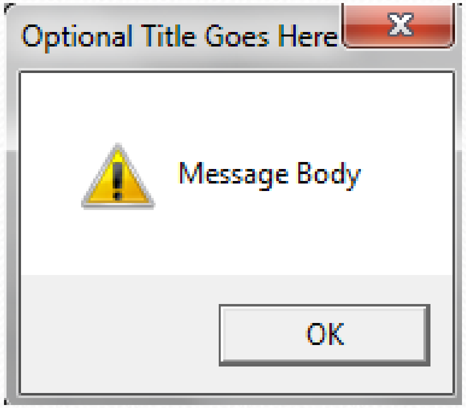
- *MsgBox "Message Body", **vbCritical**, "Optional Title Goes Here“*
- *MsgBox "Message Body", **vbQuestion**, "Optional Title Goes Here“*
- *MsgBox "Message Body", **vbExclamation**, "Optional Title Goes Here"*


Optional Title Goes Here

Message Body


Optional Title Goes Here

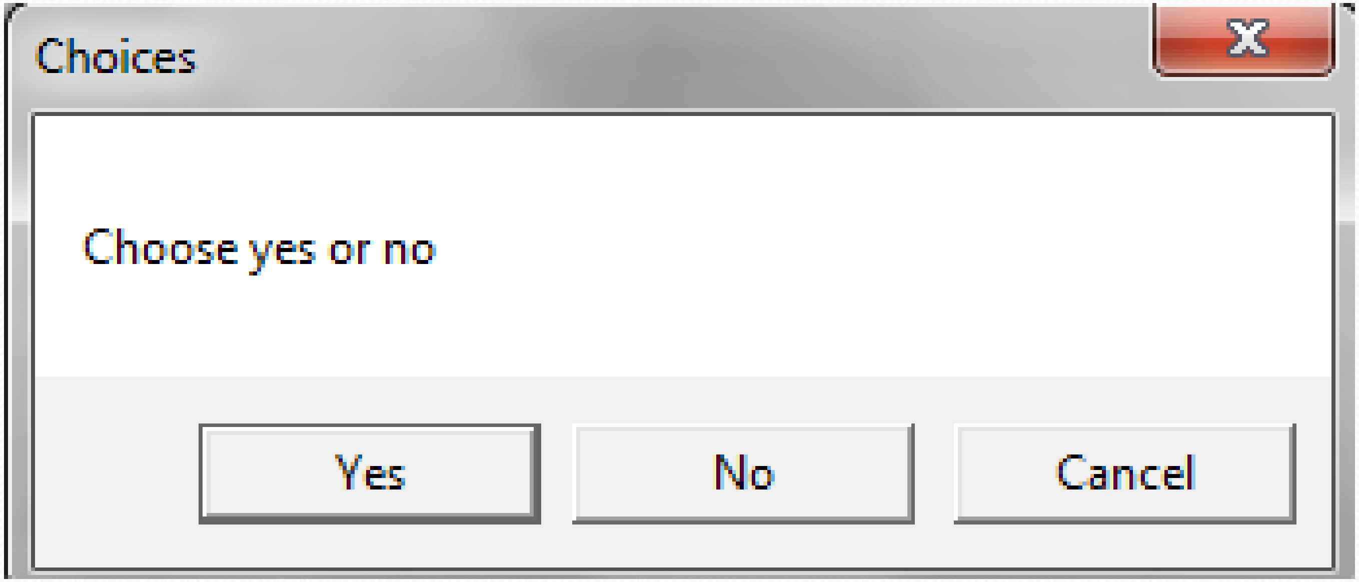
Message Body


Optional Title Goes Here

Message Body

Multi-Option Message Box

- What if we want some other button than just "Okay" ?
- In this case, we want to "capture" the clicking response
    - Will then use If-Else ladders to determine the course of action



Choices

Choose yes or no

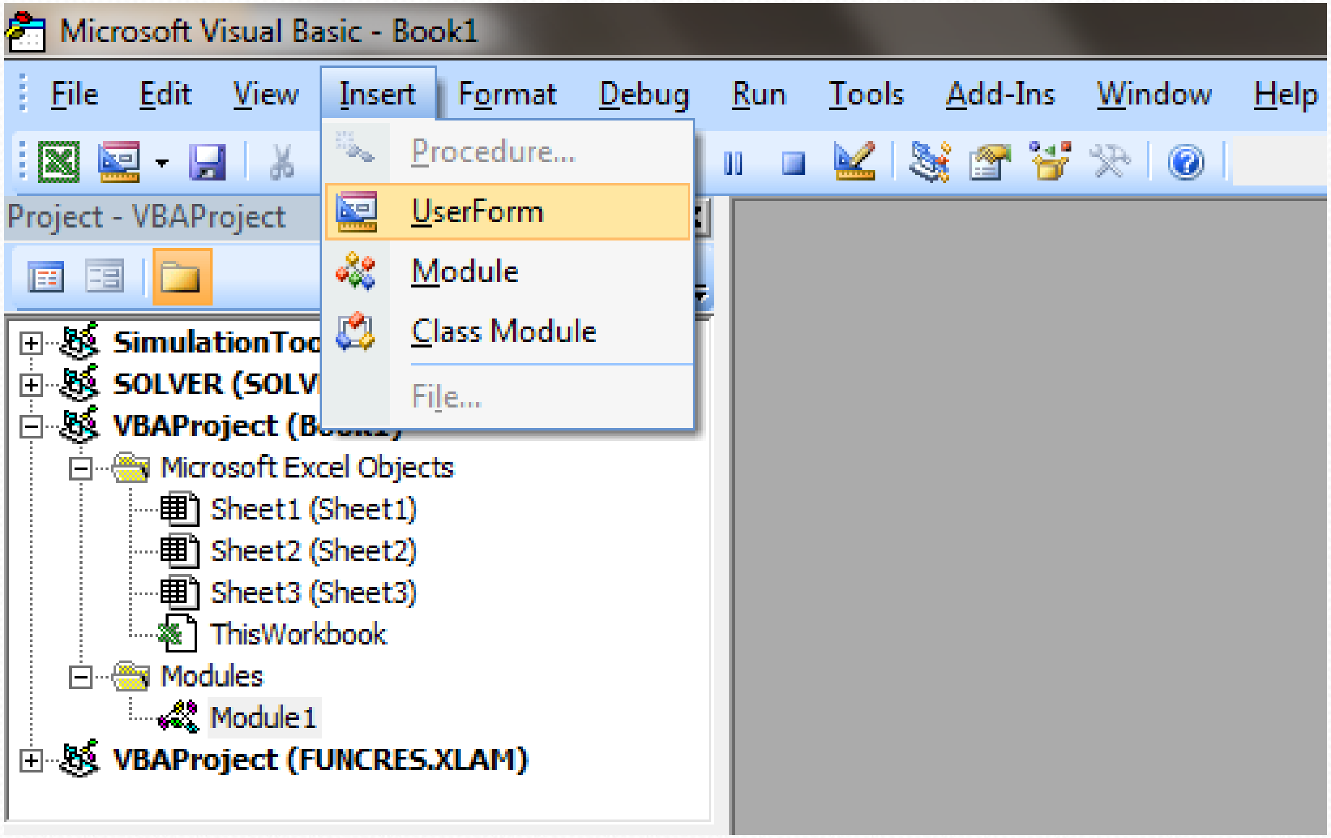
Yes or No

- *Capture the user response onto the variable “response”*

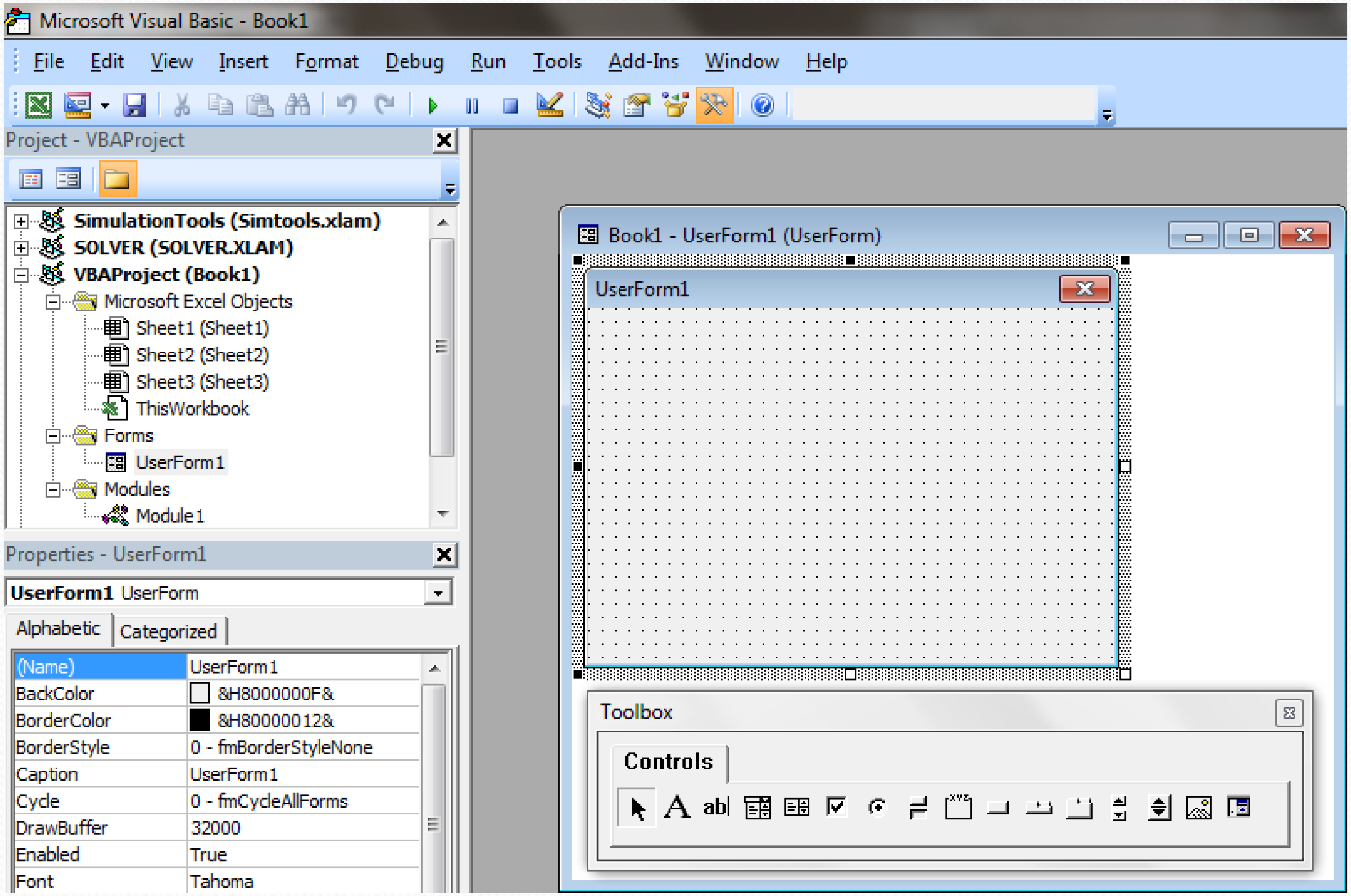
```
response = MsgBox("Choose yes or no", vbYesNoCancel, "Choices")
If response = vbYes Then
MsgBox "You clicked yes"
ElseIf response = vbNo Then
MsgBox "You clicked no"
Else
MsgBox "Why can't you follow directions?"
End If
```

Customized Userform

- Begin by inserting a blank userform

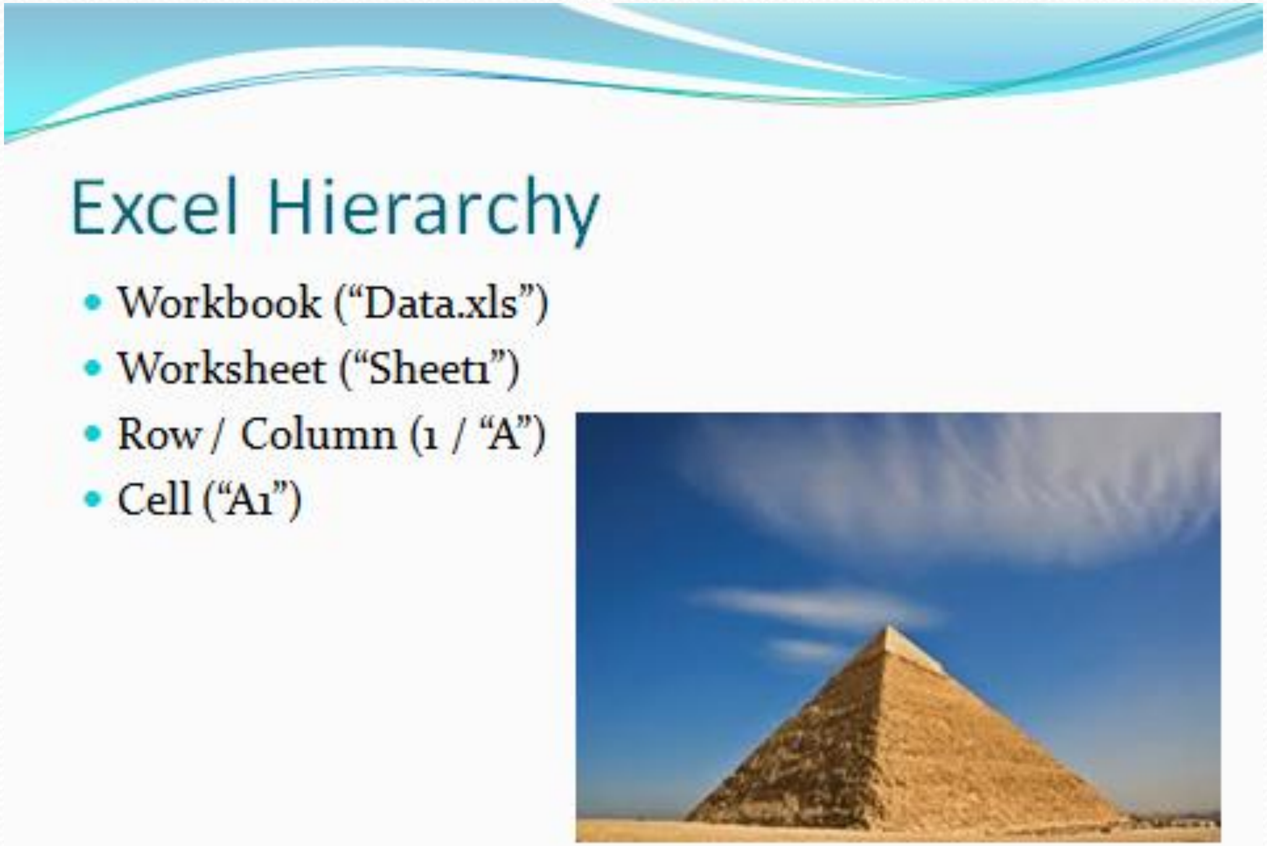


Blank Userform



(Name)	UserForm1
BackColor	□ &H8000000F&
BorderColor	■ &H80000012&
BorderStyle	0 - fmBorderStyleNone
Caption	UserForm1
Cycle	0 - fmCycleAllForms
DrawBuffer	32000
Enabled	True
Font	Tahoma

Remember this Slide?

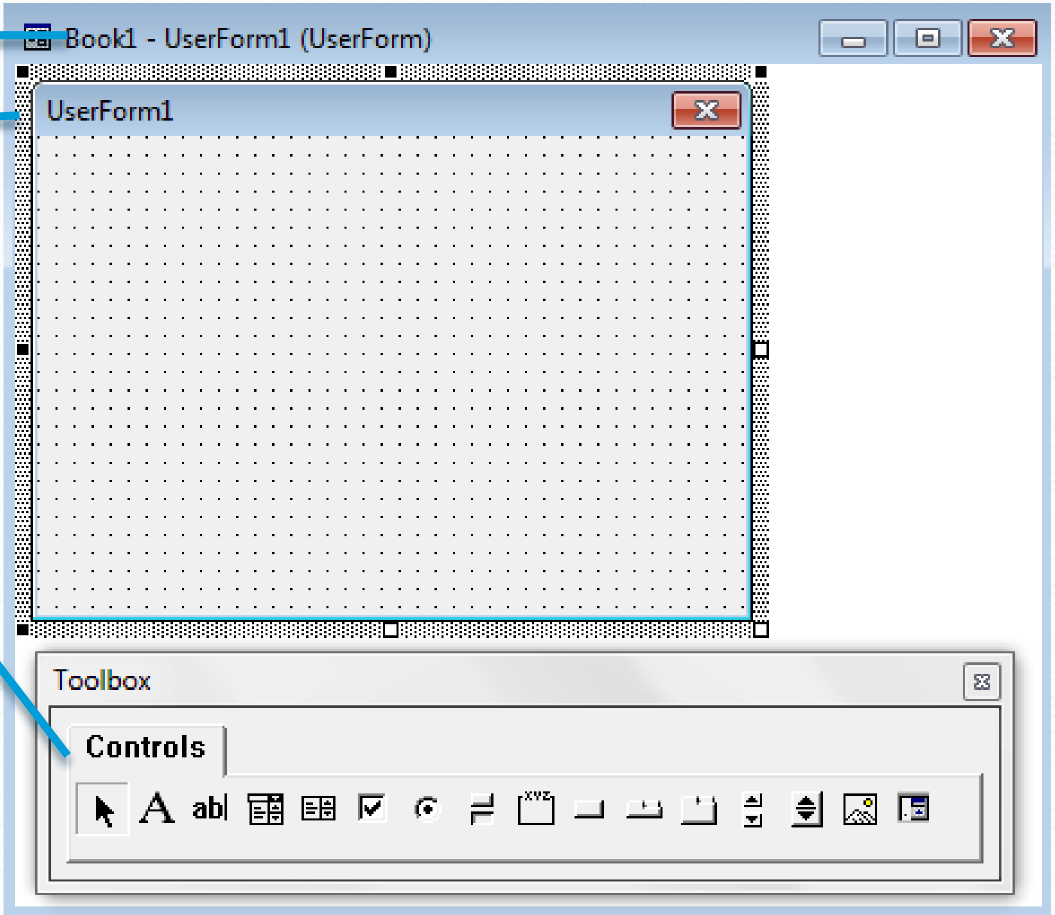


Excel Hierarchy

- Workbook ("Data.xls")
- Worksheet ("Sheet1")
- Row / Column (1 / "A")
- Cell ("A1")

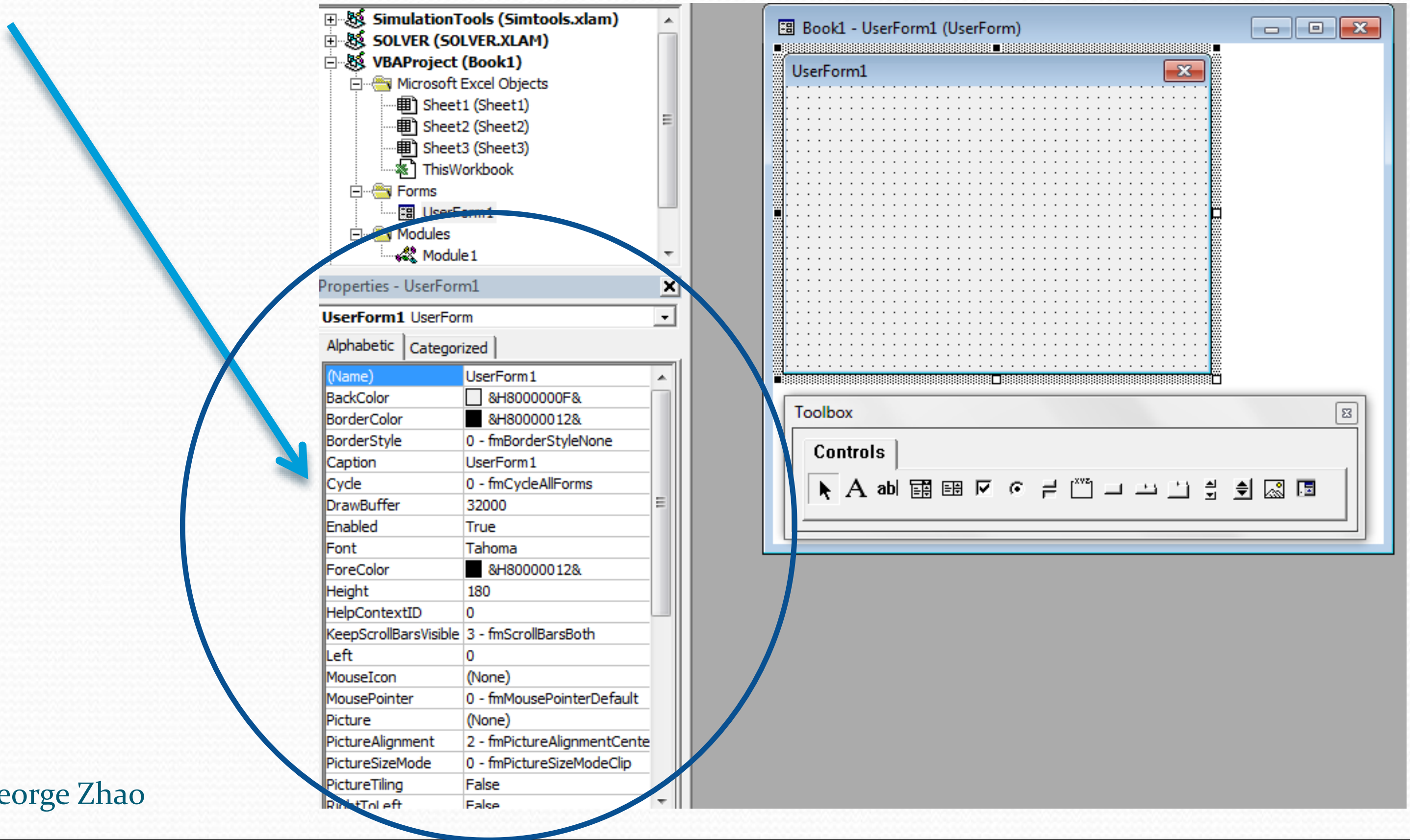
Userform Hierarchy

- Book1 (.xls file)
  - Userform1
    - Lablel1
    - ComboBox1
    - TextBox1



Properties Tab

- Important to keep track of for what the information is relating to: Userform1? Label1?



SimulationTools (Simtools.xlam)
SOLVER (SOLVER.XLAM)
VBAProject (Book1)
Microsoft Excel Objects
Sheet1 (Sheet1)
Sheet2 (Sheet2)
Sheet3 (Sheet3)
ThisWorkbook
Forms
UserForm1
Modules
Module1

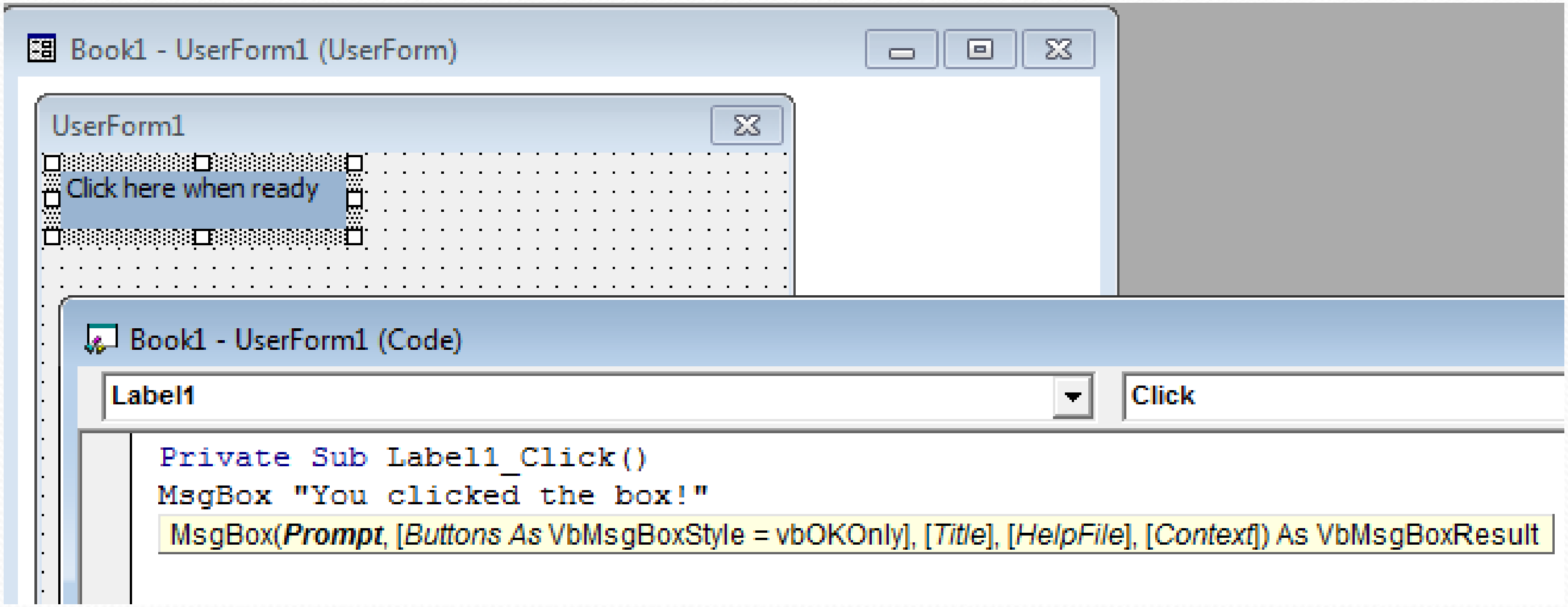
Properties - UserForm1

UserForm1 UserForm

(Name)	UserForm1
BackColor	□ &H8000000F&
BorderColor	■ &H80000012&
BorderStyle	0 - fmBorderStyleNone
Caption	UserForm1
Cycle	0 - fmCycleAllForms
DrawBuffer	32000
Enabled	True
Font	Tahoma
ForeColor	■ &H80000012&
Height	180
HelpContextID	0
KeepScrollBarsVisible	3 - fmScrollBarsBoth
Left	0
MouseIcon	(None)
MousePointer	0 - fmMousePointerDefault
Picture	(None)
PictureAlignment	2 - fmPictureAlignmentCente
PictureSizeMode	0 - fmPictureSizeModeClip
PictureTiling	False
RightToLeft	False

Label

- Create event-driven programs (much like worksheet programs, running when the Excel book opens, etc)



```
Private Sub Label1_Click()
MsgBox "You clicked the box!"
```

MsgBox(***Prompt***, [*Buttons As* VbMsgBoxStyle = vbOKOnly], [*Title*], [*HelpFile*], [*Context*]) As VbMsgBoxResult

Useful Controls in the Toolbox

- Label
  - Clicking it can enable some codes of action
- TextBox
  - Users can enter text into the box, that can be read into codes
- ComboBox
  - Users choose one of several determined options, and the selection can be read into codes

Continued

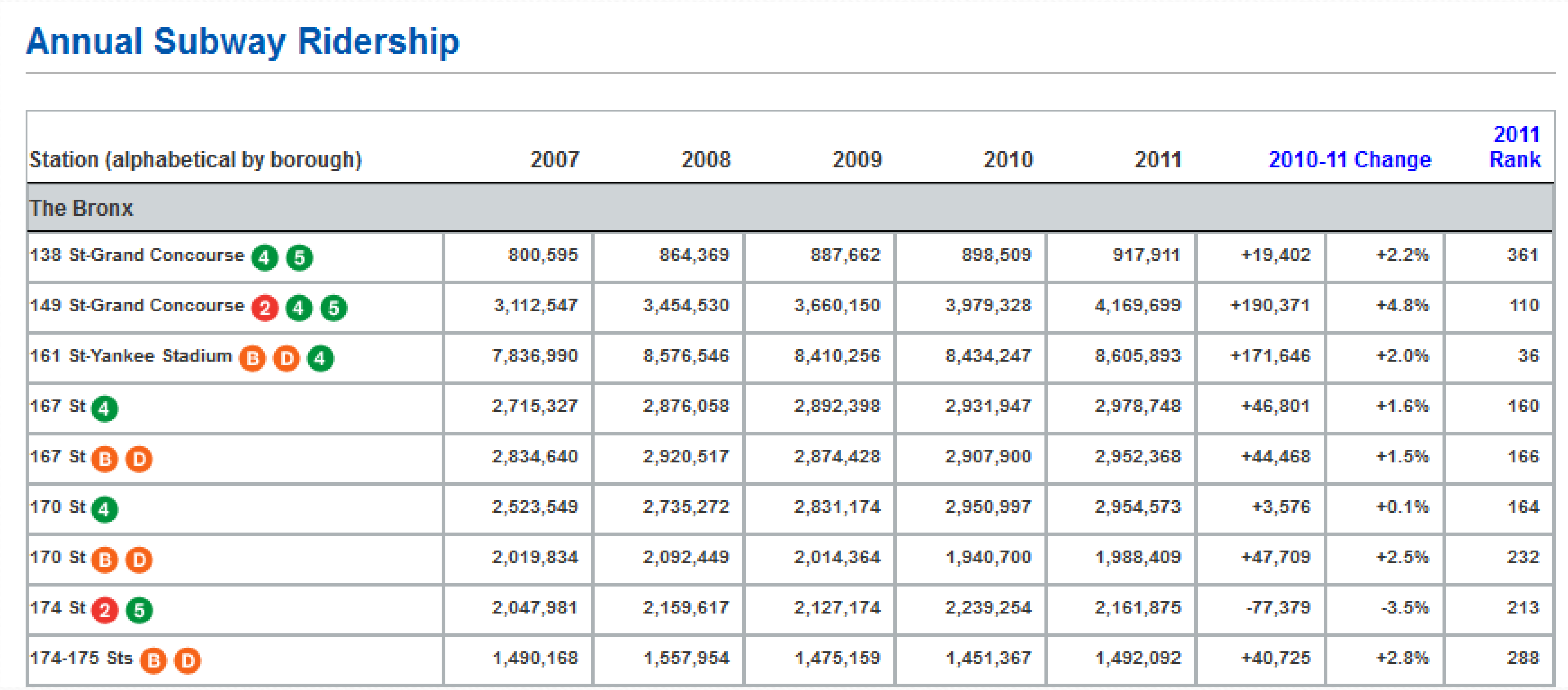
- ListBox
  - Similar to ComboBox, but multiple columns allowed
- CheckBox
  - Useful for boolean (true / false) conditions
- OptionBox
  - Can only choose one option, given multiple

- Rather than demonstrating the use for each, why not delve right into a case example to see how it works

Case 7: Subway Data All-Around Analysis

MTA Subway Ridership Data

- http://www.mta.info/nyct/facts/ridership/ridership_sub_annual.htm



Annual Subway Ridership

Station (alphabetical by borough)	2007	2008	2009	2010	2011	2010-11 Change		2011 Rank
The Bronx								
138 St-Grand Concourse 4 5	800,595	864,369	887,662	898,509	917,911	+19,402	+2.2%	361
149 St-Grand Concourse 2 4 5	3,112,547	3,454,530	3,660,150	3,979,328	4,169,699	+190,371	+4.8%	110
161 St-Yankee Stadium B D 4	7,836,990	8,576,546	8,410,256	8,434,247	8,605,893	+171,646	+2.0%	36
167 St 4	2,715,327	2,876,058	2,892,398	2,931,947	2,978,748	+46,801	+1.6%	160
167 St B D	2,834,640	2,920,517	2,874,428	2,907,900	2,952,368	+44,468	+1.5%	166
170 St 4	2,523,549	2,735,272	2,831,174	2,950,997	2,954,573	+3,576	+0.1%	164
170 St B D	2,019,834	2,092,449	2,014,364	1,940,700	1,988,409	+47,709	+2.5%	232
174 St 2 5	2,047,981	2,159,617	2,127,174	2,239,254	2,161,875	-77,379	-3.5%	213
174-175 Sts B D	1,490,168	1,557,954	1,475,159	1,451,367	1,492,092	+40,725	+2.8%	288

Pasting Data into Excel

- Directly copying, and pasting into Excel may or may not function, depending on the version of the software
- If dysfunctional:
  - Copy from website
  - Paste into Notepad
  - Copy from Notepad
  - Paste into Excel

Task Bundle 1: Clean the Data

- The station name and the numerical data are off-aligned by one row (besides the first entry)
  - Fix this
- Add an identifier to each row designating which borough the station is from
  - Can do this manually
- Clean out all of the "train icon" from the station names
  - Current format: "161 St-Yankee Stadium B train icon D train icon 4 train icon"
  - New format: "161 St-Yankee Stadium: [B, D, 4]"

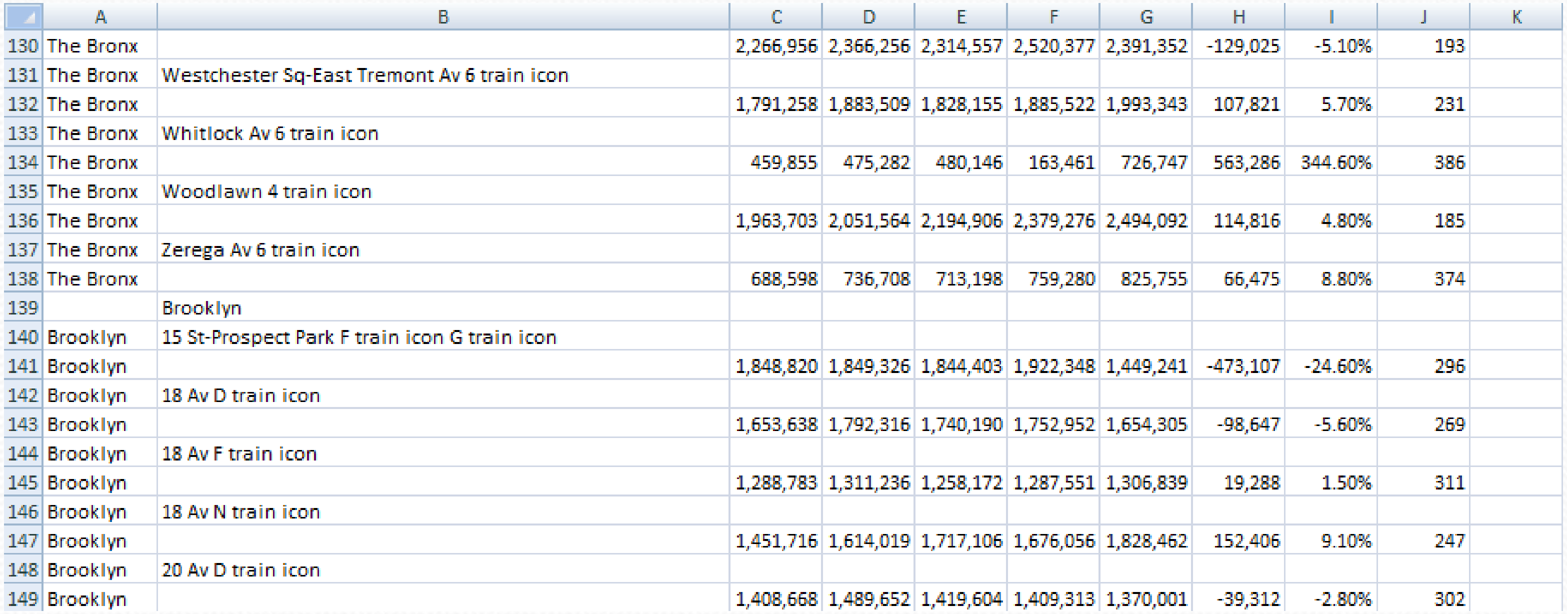
Adding Identifier

- Insert blank column to the left of column A
- Type the borough name, and drag it down until the end of the list for that borough
- Since there are only 4 boroughs (Staten Island not part of the Subway system), it's faster to do this one-time step completely manually
    - Need to recognize when to perform tasks manually vs. investing in the time to write functions or programs

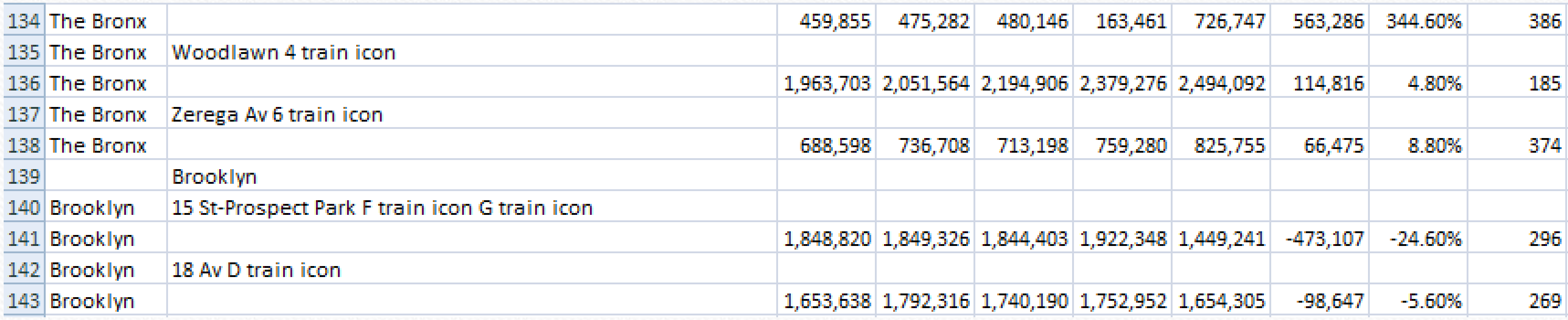
Off-Align

- For each station:
    - Take the numerical data, and move them up one row
    - Delete the row where the data used to reside in
- Do this for all stations in the borough
- Do this for the other boroughs
- This process needs to be done for all of the 400+ stations, so definitely an automated process is better.

Excel Sheet At the Moment



	A	B	C	D	E	F	G	H	I	J	K
130	The Bronx		2,266,956	2,366,256	2,314,557	2,520,377	2,391,352	-129,025	-5.10%	193	
131	The Bronx	Westchester Sq-East Tremont Av 6 train icon									
132	The Bronx		1,791,258	1,883,509	1,828,155	1,885,522	1,993,343	107,821	5.70%	231	
133	The Bronx	Whitlock Av 6 train icon									
134	The Bronx		459,855	475,282	480,146	163,461	726,747	563,286	344.60%	386	
135	The Bronx	Woodlawn 4 train icon									
136	The Bronx		1,963,703	2,051,564	2,194,906	2,379,276	2,494,092	114,816	4.80%	185	
137	The Bronx	Zerega Av 6 train icon									
138	The Bronx		688,598	736,708	713,198	759,280	825,755	66,475	8.80%	374	
139		Brooklyn									
140	Brooklyn	15 St-Prospect Park F train icon G train icon									
141	Brooklyn		1,848,820	1,849,326	1,844,403	1,922,348	1,449,241	-473,107	-24.60%	296	
142	Brooklyn	18 Av D train icon									
143	Brooklyn		1,653,638	1,792,316	1,740,190	1,752,952	1,654,305	-98,647	-5.60%	269	
144	Brooklyn	18 Av F train icon									
145	Brooklyn		1,288,783	1,311,236	1,258,172	1,287,551	1,306,839	19,288	1.50%	311	
146	Brooklyn	18 Av N train icon									
147	Brooklyn		1,451,716	1,614,019	1,717,106	1,676,056	1,828,462	152,406	9.10%	247	
148	Brooklyn	20 Av D train icon									
149	Brooklyn		1,408,668	1,489,652	1,419,604	1,409,313	1,370,001	-39,312	-2.80%	302	



134	The Bronx		459,855	475,282	480,146	163,461	726,747	563,286	344.60%	386
135	The Bronx	Woodlawn 4 train icon								
136	The Bronx		1,963,703	2,051,564	2,194,906	2,379,276	2,494,092	114,816	4.80%	185
137	The Bronx	Zerega Av 6 train icon								
138	The Bronx		688,598	736,708	713,198	759,280	825,755	66,475	8.80%	374
139		Brooklyn								
140	Brooklyn	15 St-Prospect Park F train icon G train icon								
141	Brooklyn		1,848,820	1,849,326	1,844,403	1,922,348	1,449,241	-473,107	-24.60%	296
142	Brooklyn	18 Av D train icon								
143	Brooklyn		1,653,638	1,792,316	1,740,190	1,752,952	1,654,305	-98,647	-5.60%	269

PsuedoCode

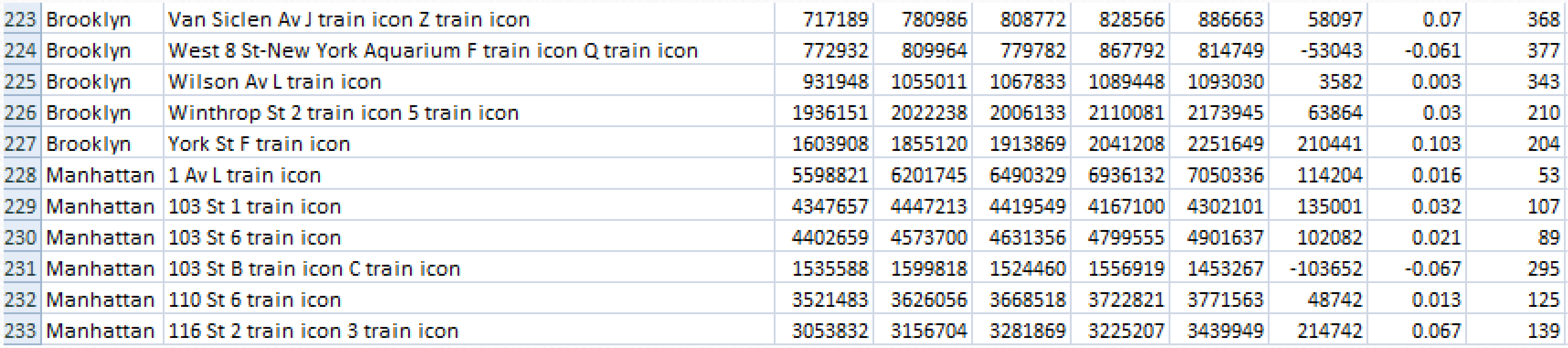
- Think from a “top-down” overview approach:
- Have a variable that tracks the number of boroughs processed
- Run the program in a loop until all 4 boroughs are traversed through:
    - For each station, copy the numerical data, and then delete the row with that data
    - End of a section of boroughs is reached when value in column A is empty
    - Delete that row and continue on

The Codes

```
Sub Align()
Range("A5").Select
boroughsTraversedThrough = 0
Do Until boroughsTraversedThrough = 4
For i = 2 To 9
Range(ActiveCell.Offset(0, i), ActiveCell.Offset(0, i)) = Range(ActiveCell.Offset(1, i), ActiveCell.Offset(1, i))
Next i
Rows(ActiveCell.Row + 1).Delete Shift:=xlUp
ActiveCell.Offset(1, 0).Select
If IsEmpty(ActiveCell.Value) Then
Rows(ActiveCell.Row).Delete Shift:=xlUp
boroughsTraversedThrough = boroughsTraversedThrough + 1
End If
Loop
End Sub
```

Processed Data

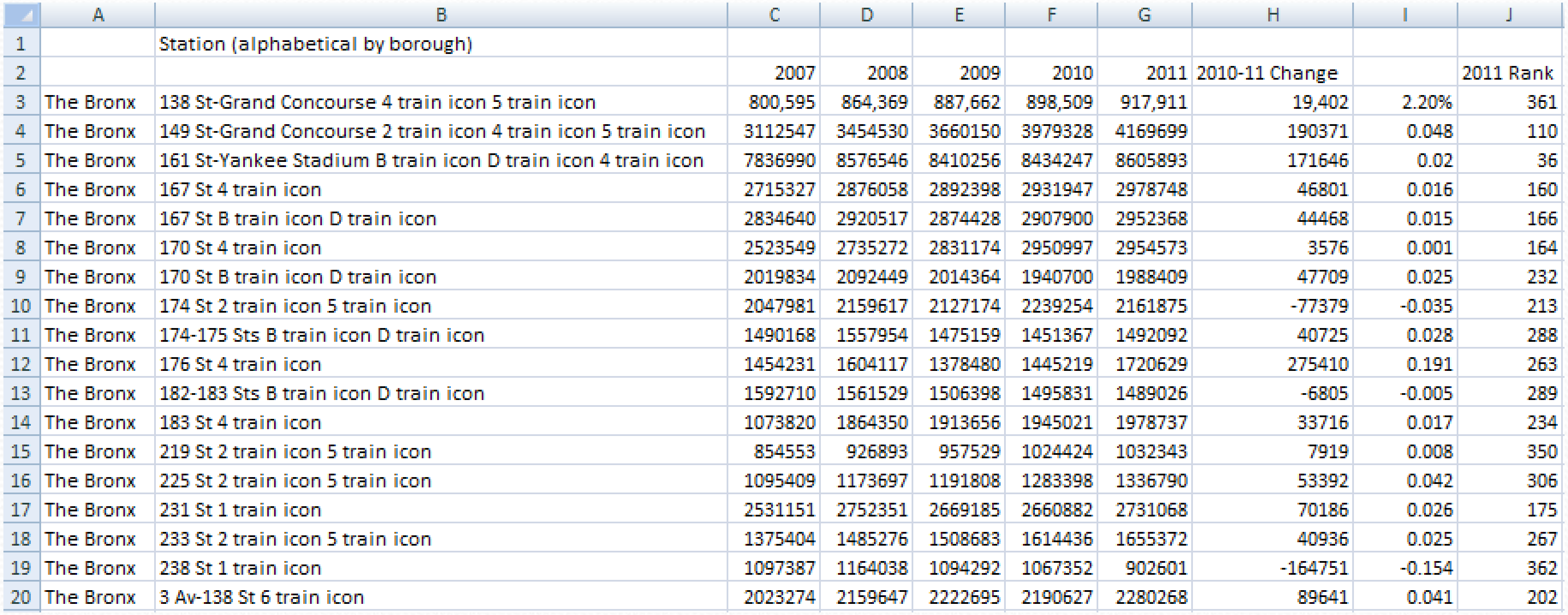
- All entries on one row, with the borough identified
- Clean transition from borough to borough



223	Brooklyn	Van Siclen Av J train icon Z train icon	717189	780986	808772	828566	886663	58097	0.07	368
224	Brooklyn	West 8 St-New York Aquarium F train icon Q train icon	772932	809964	779782	867792	814749	-53043	-0.061	377
225	Brooklyn	Wilson Av L train icon	931948	1055011	1067833	1089448	1093030	3582	0.003	343
226	Brooklyn	Winthrop St 2 train icon 5 train icon	1936151	2022238	2006133	2110081	2173945	63864	0.03	210
227	Brooklyn	York St F train icon	1603908	1855120	1913869	2041208	2251649	210441	0.103	204
228	Manhattan	1 Av L train icon	5598821	6201745	6490329	6936132	7050336	114204	0.016	53
229	Manhattan	103 St 1 train icon	4347657	4447213	4419549	4167100	4302101	135001	0.032	107
230	Manhattan	103 St 6 train icon	4402659	4573700	4631356	4799555	4901637	102082	0.021	89
231	Manhattan	103 St B train icon C train icon	1535588	1599818	1524460	1556919	1453267	-103652	-0.067	295
232	Manhattan	110 St 6 train icon	3521483	3626056	3668518	3722821	3771563	48742	0.013	125
233	Manhattan	116 St 2 train icon 3 train icon	3053832	3156704	3281869	3225207	3439949	214742	0.067	139

Small Manual Cleanup

- Move “2011 Rank” to column J
- Delete the original row 3



	A	B	C	D	E	F	G	H	I	J
1		Station (alphabetical by borough)								
2			2007	2008	2009	2010	2011	2010-11 Change		2011 Rank
3	The Bronx	138 St-Grand Concourse 4 train icon 5 train icon	800,595	864,369	887,662	898,509	917,911	19,402	2.20%	361
4	The Bronx	149 St-Grand Concourse 2 train icon 4 train icon 5 train icon	3112547	3454530	3660150	3979328	4169699	190371	0.048	110
5	The Bronx	161 St-Yankee Stadium B train icon D train icon 4 train icon	7836990	8576546	8410256	8434247	8605893	171646	0.02	36
6	The Bronx	167 St 4 train icon	2715327	2876058	2892398	2931947	2978748	46801	0.016	160
7	The Bronx	167 St B train icon D train icon	2834640	2920517	2874428	2907900	2952368	44468	0.015	166
8	The Bronx	170 St 4 train icon	2523549	2735272	2831174	2950997	2954573	3576	0.001	164
9	The Bronx	170 St B train icon D train icon	2019834	2092449	2014364	1940700	1988409	47709	0.025	232
10	The Bronx	174 St 2 train icon 5 train icon	2047981	2159617	2127174	2239254	2161875	-77379	-0.035	213
11	The Bronx	174-175 Sts B train icon D train icon	1490168	1557954	1475159	1451367	1492092	40725	0.028	288
12	The Bronx	176 St 4 train icon	1454231	1604117	1378480	1445219	1720629	275410	0.191	263
13	The Bronx	182-183 Sts B train icon D train icon	1592710	1561529	1506398	1495831	1489026	-6805	-0.005	289
14	The Bronx	183 St 4 train icon	1073820	1864350	1913656	1945021	1978737	33716	0.017	234
15	The Bronx	219 St 2 train icon 5 train icon	854553	926893	957529	1024424	1032343	7919	0.008	350
16	The Bronx	225 St 2 train icon 5 train icon	1095409	1173697	1191808	1283398	1336790	53392	0.042	306
17	The Bronx	231 St 1 train icon	2531151	2752351	2669185	2660882	2731068	70186	0.026	175
18	The Bronx	233 St 2 train icon 5 train icon	1375404	1485276	1508683	1614436	1655372	40936	0.025	267
19	The Bronx	238 St 1 train icon	1097387	1164038	1094292	1067352	902601	-164751	-0.154	362
20	The Bronx	3 Av-138 St 6 train icon	2023274	2159647	2222695	2190627	2280268	89641	0.041	202

Station Name Processing

- Change “161 St-Yankee Stadium B train icon D train icon 4 train icon” to “161 St-Yankee Stadium [B, D, 4]
- Useful ideas:
  - “train icon” designation
  - Whether B or 4, the identifier is 1 character
- Loop through column B and do this processing for all entries

InStr Function

- We need a function that finds a string (“train icon”) within a larger string
- InStr([start], string, substring, [compare])
  - *InStr("161 St-Yankee Stadium B train icon D train icon 4 train icon", "train icon")* returns 25 (first instance)
  - *InStr(**26**, "161 St-Yankee Stadium B train icon D train icon 4 train icon", "train icon")* returns 38 (second instance)
  - *InStr(**52**, "161 St-Yankee Stadium B train icon D train icon 4 train icon", "train icon")* returns 0 (none found anymore)

Mid Function

- Similar to substring() function in Java
- Mid("abcdefg", 2, 3) returns “bcd”
  - 2 is the start position (starts counting from 1, not 0)
  - 3 is the length to extract
- Mid("abcdefg", 2) returns “bcdefg”
  - If the second number is not specified, assumes rest of the string

Psuedocode

- For all entries (station names) in column B:
  - Start with position 1, and run Instr() function search in for “train icon”
  - Loop through the text until Instr() function returns 0
  - Record the letter / number (B, 4, etc), which is always two spaces to the left of the Instr() value given
  - Search again, except starting from the Instr() value + 1, so we can search for subsequent instances of “train icon”
  - Retain the station name, and add the appropriate brackets syntax

The Code

```
Sub NameProcess()

Range("B3").Select
Do Until IsEmpty(ActiveCell.Value)
fullText = ActiveCell.Value
startPosition = 1
bracketedText = "["
```

The Code

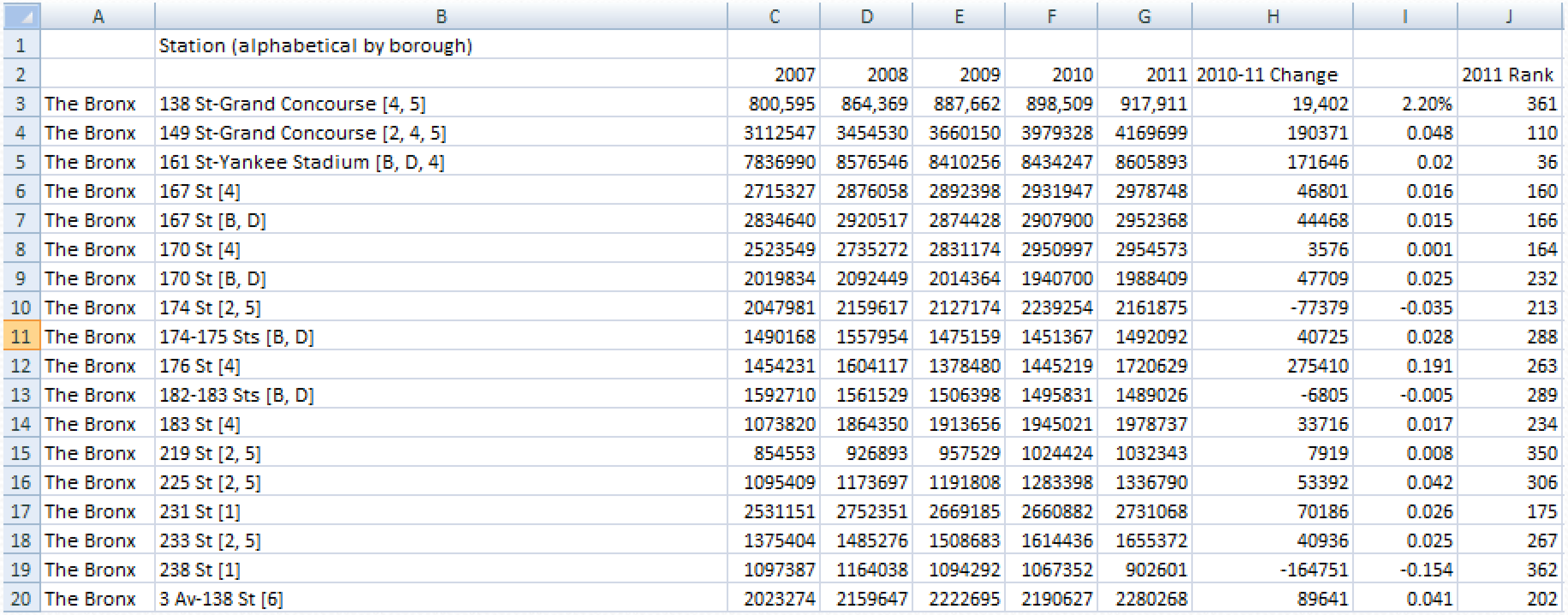
```
While InStr(startPosition, fullText, "train icon") <> 0
result = InStr(startPosition, fullText, "train icon")
bracketedText = bracketedText & Mid(fullText, result - 2, 1) & ", "
startPosition = result + 1
Wend
```

The Code

```
stationName = Left(fullText, InStr(fullText, "train icon") - 3) & bracketedText
stationName = Left(stationName, Len(stationName) - 2) & "]"
ActiveCell.Value = stationName
ActiveCell.Offset(1, 0).Select
Loop
End Sub
```

Looks Much Better

- Task bundle 1 complete!



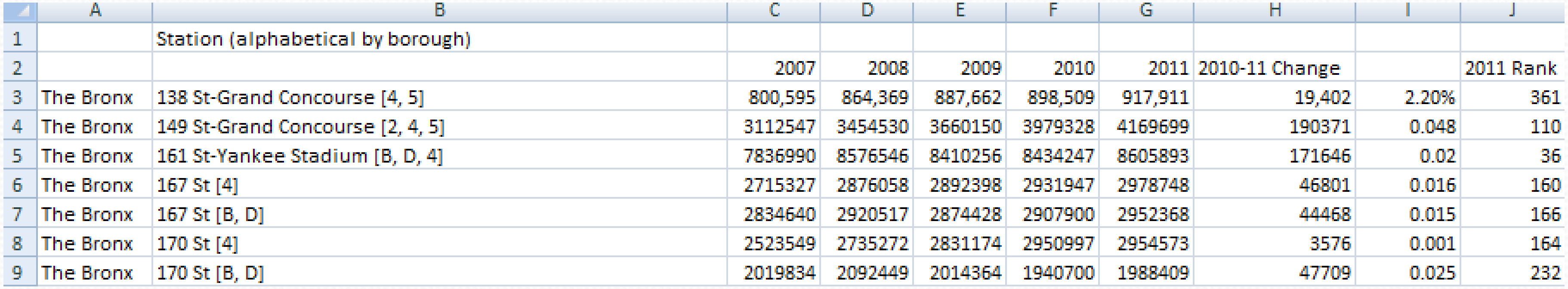
	A	B	C	D	E	F	G	H	I	J
1		Station (alphabetical by borough)								
2			2007	2008	2009	2010	2011	2010-11 Change		2011 Rank
3	The Bronx	138 St-Grand Concourse [4, 5]	800,595	864,369	887,662	898,509	917,911	19,402	2.20%	361
4	The Bronx	149 St-Grand Concourse [2, 4, 5]	3112547	3454530	3660150	3979328	4169699	190371	0.048	110
5	The Bronx	161 St-Yankee Stadium [B, D, 4]	7836990	8576546	8410256	8434247	8605893	171646	0.02	36
6	The Bronx	167 St [4]	2715327	2876058	2892398	2931947	2978748	46801	0.016	160
7	The Bronx	167 St [B, D]	2834640	2920517	2874428	2907900	2952368	44468	0.015	166
8	The Bronx	170 St [4]	2523549	2735272	2831174	2950997	2954573	3576	0.001	164
9	The Bronx	170 St [B, D]	2019834	2092449	2014364	1940700	1988409	47709	0.025	232
10	The Bronx	174 St [2, 5]	2047981	2159617	2127174	2239254	2161875	-77379	-0.035	213
11	The Bronx	174-175 Sts [B, D]	1490168	1557954	1475159	1451367	1492092	40725	0.028	288
12	The Bronx	176 St [4]	1454231	1604117	1378480	1445219	1720629	275410	0.191	263
13	The Bronx	182-183 Sts [B, D]	1592710	1561529	1506398	1495831	1489026	-6805	-0.005	289
14	The Bronx	183 St [4]	1073820	1864350	1913656	1945021	1978737	33716	0.017	234
15	The Bronx	219 St [2, 5]	854553	926893	957529	1024424	1032343	7919	0.008	350
16	The Bronx	225 St [2, 5]	1095409	1173697	1191808	1283398	1336790	53392	0.042	306
17	The Bronx	231 St [1]	2531151	2752351	2669185	2660882	2731068	70186	0.026	175
18	The Bronx	233 St [2, 5]	1375404	1485276	1508683	1614436	1655372	40936	0.025	267
19	The Bronx	238 St [1]	1097387	1164038	1094292	1067352	902601	-164751	-0.154	362
20	The Bronx	3 Av-138 St [6]	2023274	2159647	2222695	2190627	2280268	89641	0.041	202

Task Bundle 2: Extreme Trends

- Identify the stations that has had, over the span of the data:
  - Decreasing ridership in each successive year
  - Increasing ridership in each successive year
    - Summarize in a condensed table, the percentage of stations in each borough that satisfied this quality
  - Largest percentage increase in ridership
  - Largest percentage decrease in ridership

Strategy

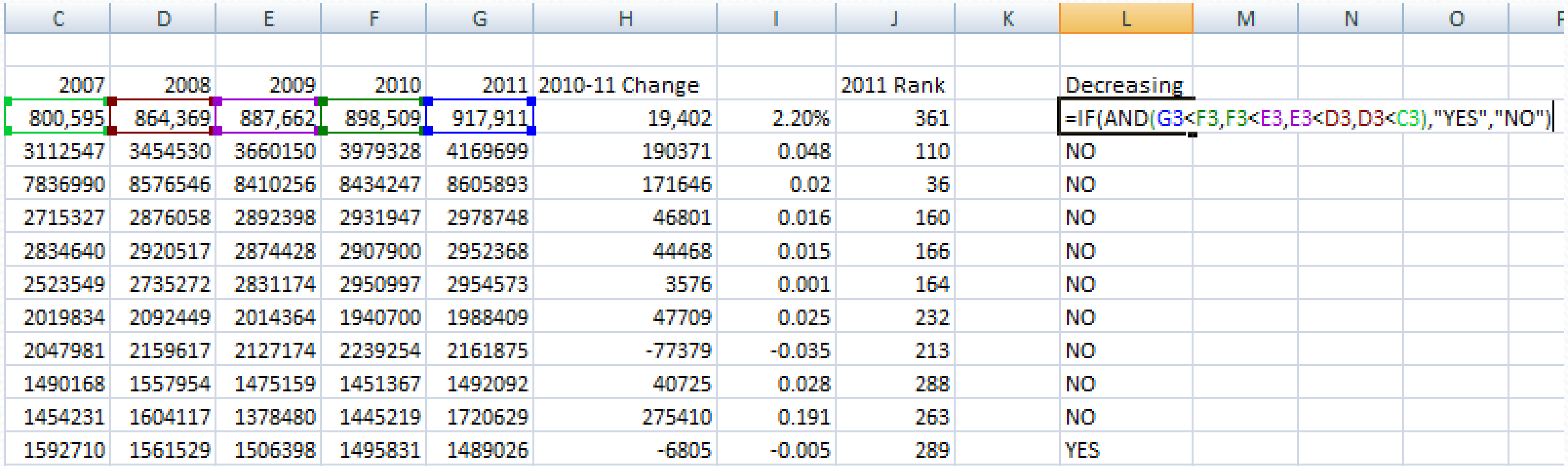
- For stations with increasing ridership in each successive year:
  - Value in column G > value in column F
  - AND column F > column E
  - AND column E > column D
  - AND column D > column C



	A	B	C	D	E	F	G	H	I	J
1		Station (alphabetical by borough)								
2			2007	2008	2009	2010	2011	2010-11 Change		2011 Rank
3	The Bronx	138 St-Grand Concourse [4, 5]	800,595	864,369	887,662	898,509	917,911	19,402	2.20%	361
4	The Bronx	149 St-Grand Concourse [2, 4, 5]	3112547	3454530	3660150	3979328	4169699	190371	0.048	110
5	The Bronx	161 St-Yankee Stadium [B, D, 4]	7836990	8576546	8410256	8434247	8605893	171646	0.02	36
6	The Bronx	167 St [4]	2715327	2876058	2892398	2931947	2978748	46801	0.016	160
7	The Bronx	167 St [B, D]	2834640	2920517	2874428	2907900	2952368	44468	0.015	166
8	The Bronx	170 St [4]	2523549	2735272	2831174	2950997	2954573	3576	0.001	164
9	The Bronx	170 St [B, D]	2019834	2092449	2014364	1940700	1988409	47709	0.025	232

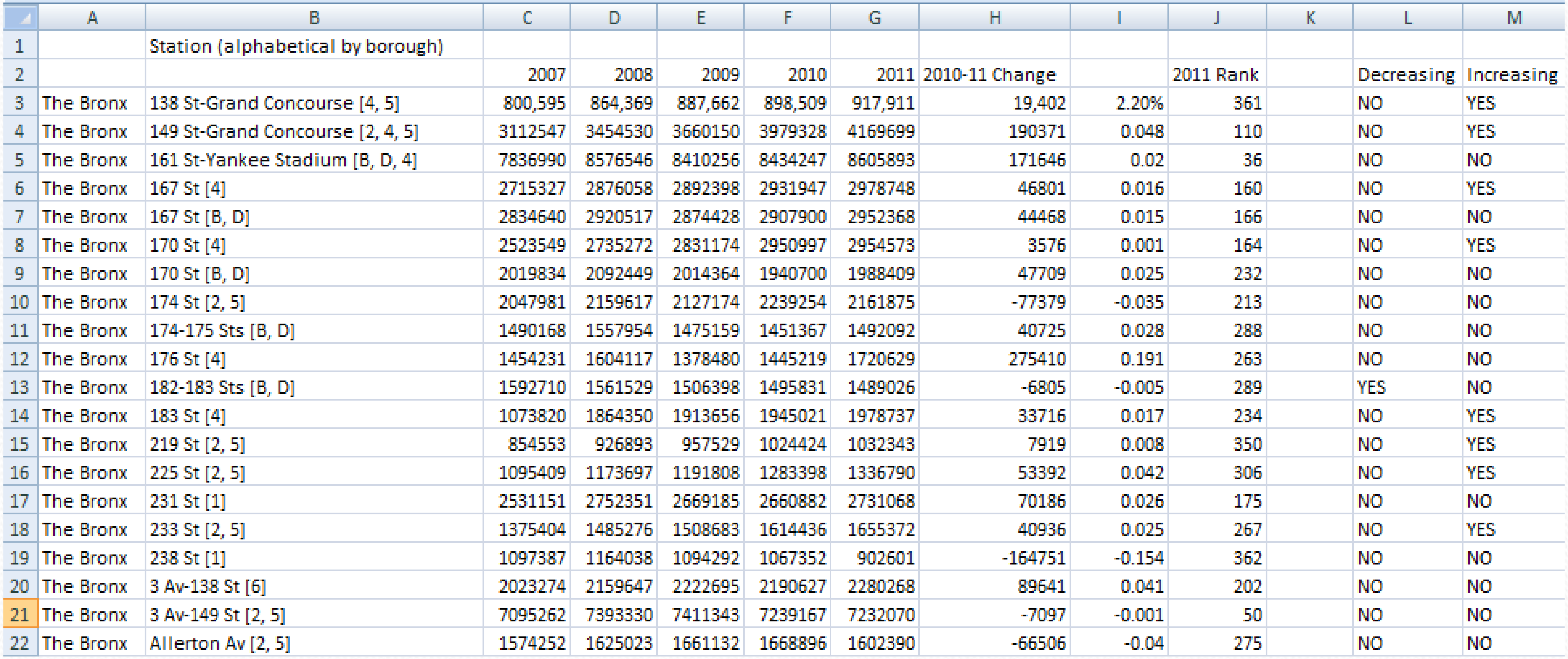
AND Operator

- In VBA programming:
  - *If [condition 1] AND [condition 2] Then …*
- But in Excel cells:
  - AND([condition 1], [condition 2] …)



C	D	E	F	G	H	I	J	K	L	M	N	O
2007	2008	2009	2010	2011	2010-11 Change		2011 Rank		Decreasing			
800,595	864,369	887,662	898,509	917,911	19,402	2.20%	361		=IF(AND(G3<F3,F3<E3,E3<D3,D3<C3),"YES","NO")			
3112547	3454530	3660150	3979328	4169699	190371	0.048	110		NO			
7836990	8576546	8410256	8434247	8605893	171646	0.02	36		NO			
2715327	2876058	2892398	2931947	2978748	46801	0.016	160		NO			
2834640	2920517	2874428	2907900	2952368	44468	0.015	166		NO			
2523549	2735272	2831174	2950997	2954573	3576	0.001	164		NO			
2019834	2092449	2014364	1940700	1988409	47709	0.025	232		NO			
2047981	2159617	2127174	2239254	2161875	-77379	-0.035	213		NO			
1490168	1557954	1475159	1451367	1492092	40725	0.028	288		NO			
1454231	1604117	1378480	1445219	1720629	275410	0.191	263		NO			
1592710	1561529	1506398	1495831	1489026	-6805	-0.005	289		YES			

Decreasing and Increasing



	A	B	C	D	E	F	G	H	I	J	K	L	M
1		Station (alphabetical by borough)											
2			2007	2008	2009	2010	2011	2010-11 Change		2011 Rank		Decreasing	Increasing
3	The Bronx	138 St-Grand Concourse [4, 5]	800,595	864,369	887,662	898,509	917,911	19,402	2.20%	361		NO	YES
4	The Bronx	149 St-Grand Concourse [2, 4, 5]	3112547	3454530	3660150	3979328	4169699	190371	0.048	110		NO	YES
5	The Bronx	161 St-Yankee Stadium [B, D, 4]	7836990	8576546	8410256	8434247	8605893	171646	0.02	36		NO	NO
6	The Bronx	167 St [4]	2715327	2876058	2892398	2931947	2978748	46801	0.016	160		NO	YES
7	The Bronx	167 St [B, D]	2834640	2920517	2874428	2907900	2952368	44468	0.015	166		NO	NO
8	The Bronx	170 St [4]	2523549	2735272	2831174	2950997	2954573	3576	0.001	164		NO	YES
9	The Bronx	170 St [B, D]	2019834	2092449	2014364	1940700	1988409	47709	0.025	232		NO	NO
10	The Bronx	174 St [2, 5]	2047981	2159617	2127174	2239254	2161875	-77379	-0.035	213		NO	NO
11	The Bronx	174-175 Sts [B, D]	1490168	1557954	1475159	1451367	1492092	40725	0.028	288		NO	NO
12	The Bronx	176 St [4]	1454231	1604117	1378480	1445219	1720629	275410	0.191	263		NO	NO
13	The Bronx	182-183 Sts [B, D]	1592710	1561529	1506398	1495831	1489026	-6805	-0.005	289		YES	NO
14	The Bronx	183 St [4]	1073820	1864350	1913656	1945021	1978737	33716	0.017	234		NO	YES
15	The Bronx	219 St [2, 5]	854553	926893	957529	1024424	1032343	7919	0.008	350		NO	YES
16	The Bronx	225 St [2, 5]	1095409	1173697	1191808	1283398	1336790	53392	0.042	306		NO	YES
17	The Bronx	231 St [1]	2531151	2752351	2669185	2660882	2731068	70186	0.026	175		NO	NO
18	The Bronx	233 St [2, 5]	1375404	1485276	1508683	1614436	1655372	40936	0.025	267		NO	YES
19	The Bronx	238 St [1]	1097387	1164038	1094292	1067352	902601	-164751	-0.154	362		NO	NO
20	The Bronx	3 Av-138 St [6]	2023274	2159647	2222695	2190627	2280268	89641	0.041	202		NO	NO
21	The Bronx	3 Av-149 St [2, 5]	7095262	7393330	7411343	7239167	7232070	-7097	-0.001	50		NO	NO
22	The Bronx	Allerton Av [2, 5]	1574252	1625023	1661132	1668896	1602390	-66506	-0.04	275		NO	NO

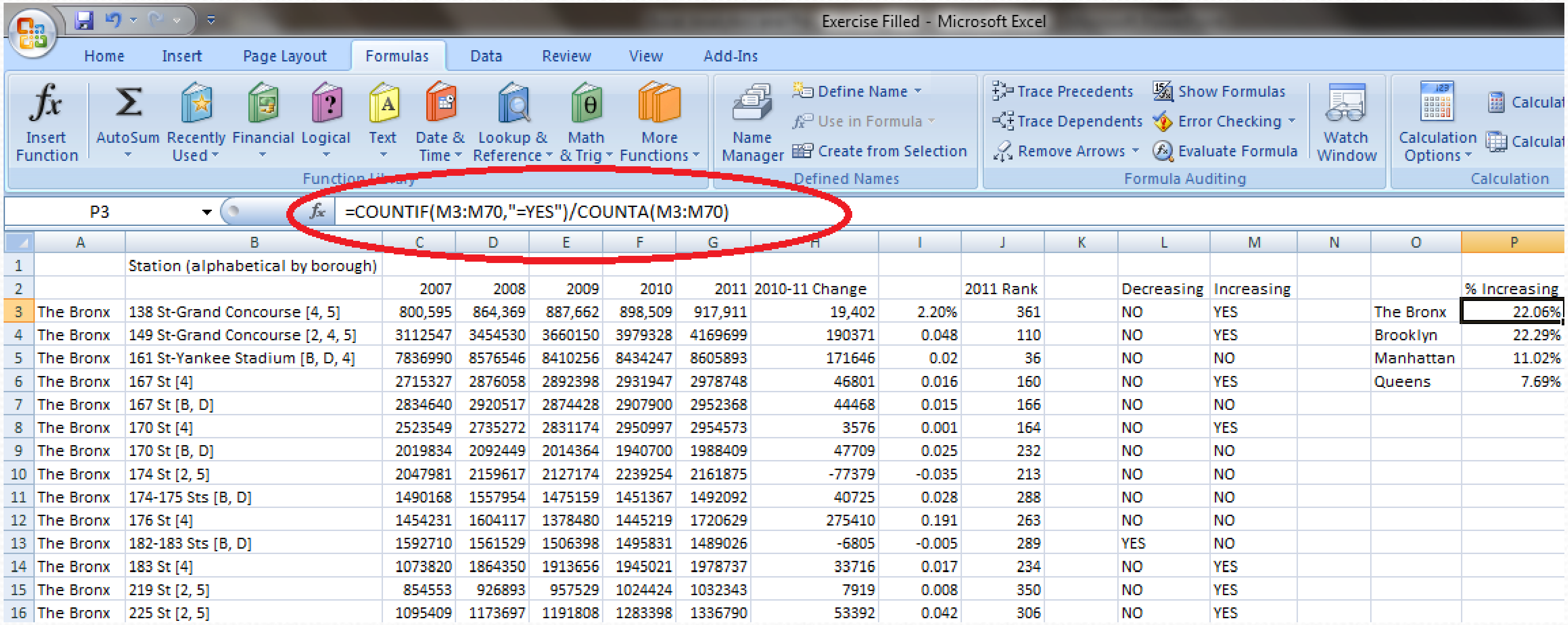
Borough Tally Chart

- Count the number of entries with “YES”
  - =COUNTIF([Range], “=YES”)
- Count the number of total entries
  - =COUNTA([Range])
  - Row ranges of the boroughs can be hard-coded, since there are only 4 of them, and that they are “one-time” items

Different COUNT Methods

- Empty vs. blank
  - A cell with “” is blank but NOT empty
- =COUNT([Range])
  - Cells that contain numbers
- =COUNTA([Range])
  - Cells that are not empty
- =COUNTBLANK([Range])
  - Cells that are blank
- =COUNTIF([Range], [Condition])
  - Cells that satisfy the condition

Results Table



Exercise Filled - Microsoft Excel

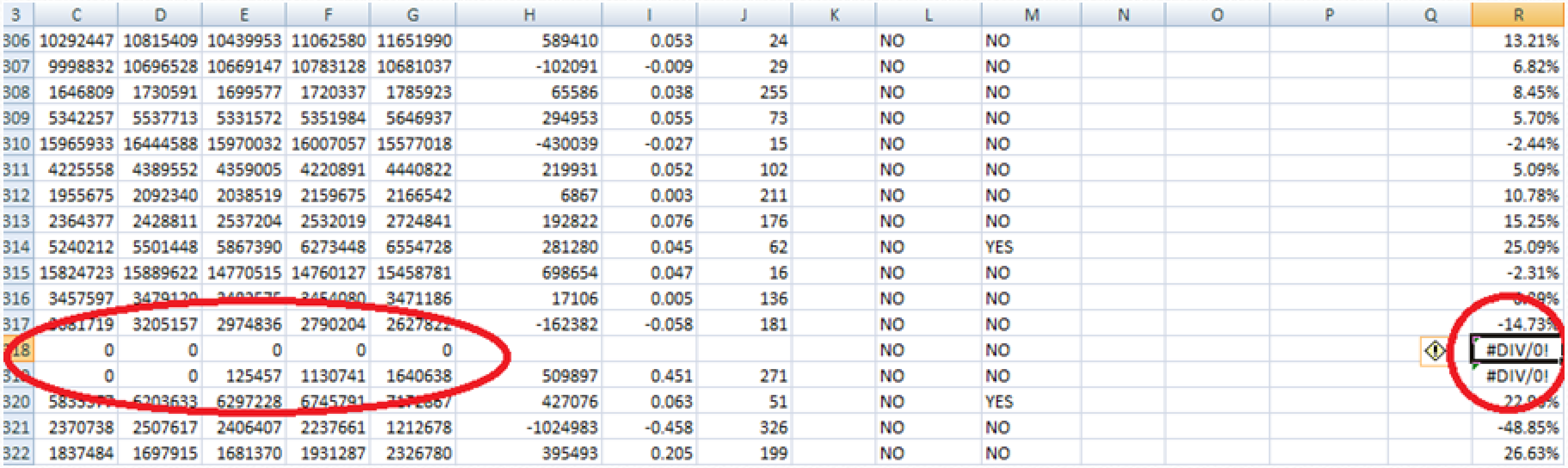
P3

=COUNTIF(M3:M70,"=YES")/COUNTA(M3:M70)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1		Station (alphabetical by borough)														
2			2007	2008	2009	2010	2011	2010-11 Change		2011 Rank		Decreasing	Increasing			% Increasing
3	The Bronx	138 St-Grand Concourse [4, 5]	800,595	864,369	887,662	898,509	917,911	19,402	2.20%	361		NO	YES		The Bronx	22.06%
4	The Bronx	149 St-Grand Concourse [2, 4, 5]	3112547	3454530	3660150	3979328	4169699	190371	0.048	110		NO	YES		Brooklyn	22.29%
5	The Bronx	161 St-Yankee Stadium [B, D, 4]	7836990	8576546	8410256	8434247	8605893	171646	0.02	36		NO	NO		Manhattan	11.02%
6	The Bronx	167 St [4]	2715327	2876058	2892398	2931947	2978748	46801	0.016	160		NO	YES		Queens	7.69%
7	The Bronx	167 St [B, D]	2834640	2920517	2874428	2907900	2952368	44468	0.015	166		NO	NO			
8	The Bronx	170 St [4]	2523549	2735272	2831174	2950997	2954573	3576	0.001	164		NO	YES			
9	The Bronx	170 St [B, D]	2019834	2092449	2014364	1940700	1988409	47709	0.025	232		NO	NO			
10	The Bronx	174 St [2, 5]	2047981	2159617	2127174	2239254	2161875	-77379	-0.035	213		NO	NO			
11	The Bronx	174-175 Sts [B, D]	1490168	1557954	1475159	1451367	1492092	40725	0.028	288		NO	NO			
12	The Bronx	176 St [4]	1454231	1604117	1378480	1445219	1720629	275410	0.191	263		NO	NO			
13	The Bronx	182-183 Sts [B, D]	1592710	1561529	1506398	1495831	1489026	-6805	-0.005	289		YES	NO			
14	The Bronx	183 St [4]	1073820	1864350	1913656	1945021	1978737	33716	0.017	234		NO	YES			
15	The Bronx	219 St [2, 5]	854553	926893	957529	1024424	1032343	7919	0.008	350		NO	YES			
16	The Bronx	225 St [2, 5]	1095409	1173697	1191808	1283398	1336790	53392	0.042	306		NO	YES			

Determining Max / Min Change

- Create a new column (column R) that calculates the % change in ridership from 2011 to 2007
- Determine the max and min from the column, and determine the station name for those rows

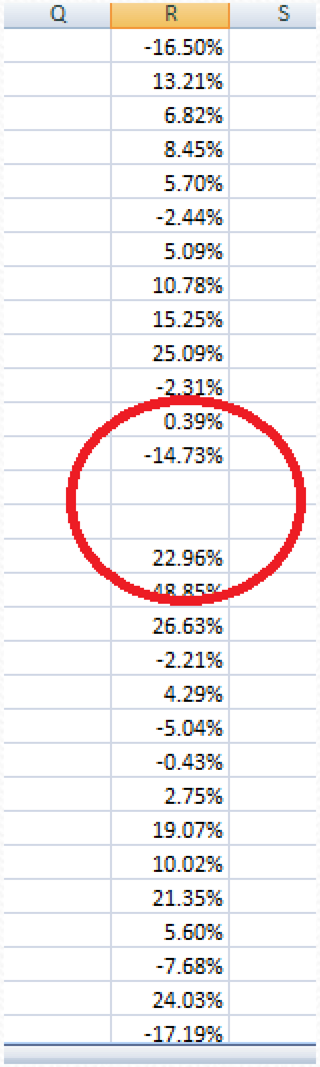


	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
306	10292447	10815409	10439953	11062580	11651990	589410	0.053	24		NO	NO					13.21%
307	9998832	10696528	10669147	10783128	10681037	-102091	-0.009	29		NO	NO					6.82%
308	1646809	1730591	1699577	1720337	1785923	65586	0.038	255		NO	NO					8.45%
309	5342257	5537713	5331572	5351984	5646937	294953	0.055	73		NO	NO					5.70%
310	15965933	16444588	15970032	16007057	15577018	-430039	-0.027	15		NO	NO					-2.44%
311	4225558	4389552	4359005	4220891	4440822	219931	0.052	102		NO	NO					5.09%
312	1955675	2092340	2038519	2159675	2166542	6867	0.003	211		NO	NO					10.78%
313	2364377	2428811	2537204	2532019	2724841	192822	0.076	176		NO	NO					15.25%
314	5240212	5501448	5867390	6273448	6554728	281280	0.045	62		NO	YES					25.09%
315	15824723	15889622	14770515	14760127	15458781	698654	0.047	16		NO	NO					-2.31%
316	3457597	3479120	[illegible]	3454080	3471186	17106	0.005	136		NO	NO					0.39%
317	3081719	3205157	2974836	2790204	2627822	-162382	-0.058	181		NO	NO					-14.73%
318	0	0	0	0	0					NO	NO				⟐	#DIV/0!
319	0	0	125457	1130741	1640638	509897	0.451	271		NO	NO					#DIV/0!
320	[illegible]	6203633	6297228	6745791	7172867	427076	0.063	51		NO	YES					[illegible]
321	2370738	2507617	2406407	2237661	1212678	-1024983	-0.458	326		NO	NO					-48.85%
322	1837484	1697915	1681370	1931287	2326780	395493	0.205	199		NO	NO					26.63%

Created by George Zhao

Error Trap

- Even with a calculation as simple as percentage change: (G3-C3)/C3
- Needs to take into consideration of "bad data"
- Having #DIV/0! Error will not allow for further calculations, such as the max and min
- =IFERROR((G3-C3)/C3,"")



Q	R	S
	-16.50%	
	13.21%	
	6.82%	
	8.45%	
	5.70%	
	-2.44%	
	5.09%	
	10.78%	
	15.25%	
	25.09%	
	-2.31%	
	0.39%	
	-14.73%	
	22.96%	
	48.85%	
	26.63%	
	-2.21%	
	4.29%	
	-5.04%	
	-0.43%	
	2.75%	
	19.07%	
	10.02%	
	21.35%	
	5.60%	
	-7.68%	
	24.03%	
	-17.19%	

Problem with =VLOOKUP()

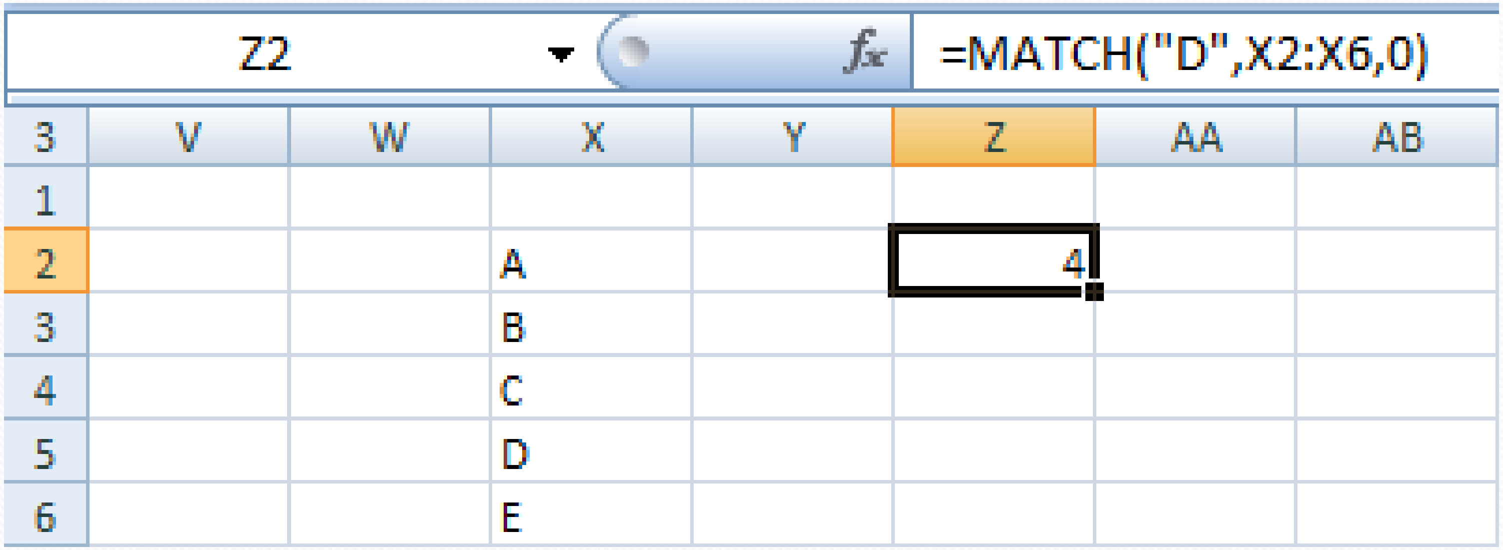
- Suppose we wanted to use this:
  - =VLOOKUP(MAX(R3:R423),A3:R423,2,FALSE)
- After all, the station name is in column B (second column)
- Problem is that the "lookup value" must be in the first column of the search table
- Alas, the "lookup value" (% change values) are on the very last column

Offset?

- Can we locate the row containing the max value, and then use offset to move over to the correct column?
- =OFFSET() needs a reference cell
- Unfortunately, =VLOOKUP() returns only the value, not the reference
- Need a function that returns a reference in an array, given a lookup value

=MATCH()

- =MATCH([lookup value], [lookup array], [match type])
- 0 as the match type for exact match

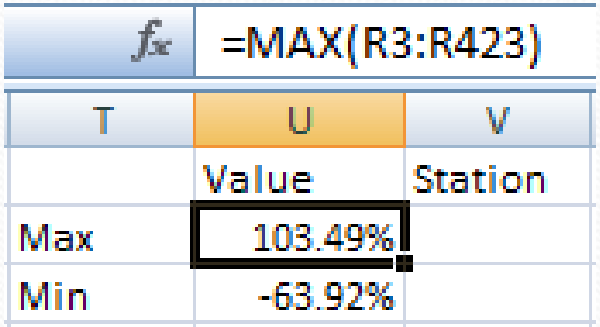


=MATCH("D",X2:X6,0)

3	V	W	X	Y	Z	AA	AB
1							
2			A		4		
3			B				
4			C				
5			D				
6			E				

Back to Determining Max / Min

- First create a column to house the numerical max and min values



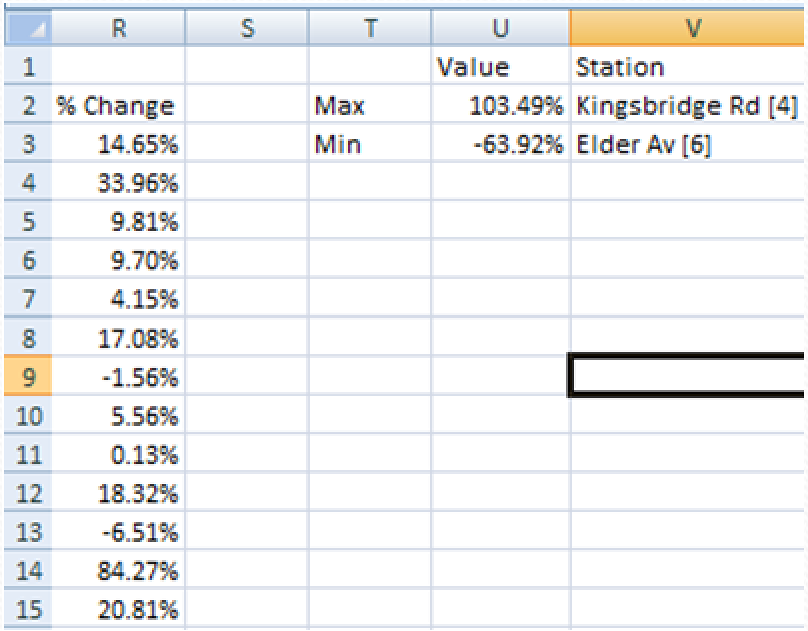
```
fx  =MAX(R3:R423)
```

	T	U	V
		Value	Station
	Max	103.49%	
	Min	-63.92%	

- Think about it:
  - =OFFSET() needs a baseline reference, and two directions to branch out to
  - =MATCH() needs a data set, and returns a subsequent reference
  - Any way to use both simultaneously?

Work from Inside Out

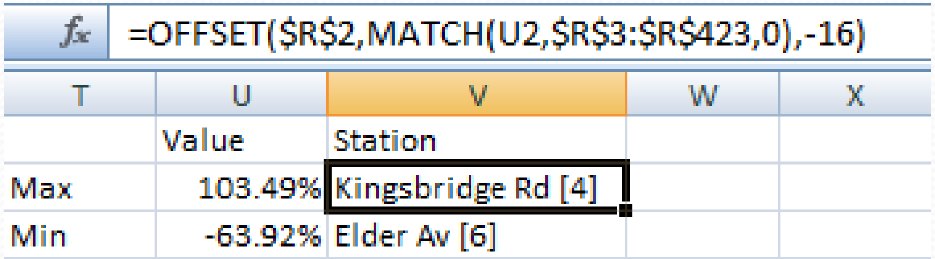
- MATCH(U2,$R$3:$R$423,0)
  - Returns n, where the max value is the n-th entry in the numerical data in column R
  - Or, think of it as the number of row to offset down from R2



	R	S	T	U	V
1				Value	Station
2	% Change		Max	103.49%	Kingsbridge Rd [4]
3	14.65%		Min	-63.92%	Elder Av [6]
4	33.96%				
5	9.81%				
6	9.70%				
7	4.15%				
8	17.08%				
9	-1.56%				
10	5.56%				
11	0.13%				
12	18.32%				
13	-6.51%				
14	84.27%				
15	20.81%				

What We Want

- We want the station name (column B)
- Recall from previous slide: MATCH(U2,$R$3:$R$423,0) is the number of row to offset down from R2
- What about the number of columns?
  - -16 (with reference at column R, column B is 16 columns to the left)
- So the overall formula we want is:
  - =OFFSET($R$2,MATCH(U2,$R$3:$R$423,0),-16)



```
fx  =OFFSET($R$2,MATCH(U2,$R$3:$R$423,0),-16)
```

T	U	V	W	X
	Value	Station		
Max	103.49%	Kingsbridge Rd [4]		
Min	-63.92%	Elder Av [6]		

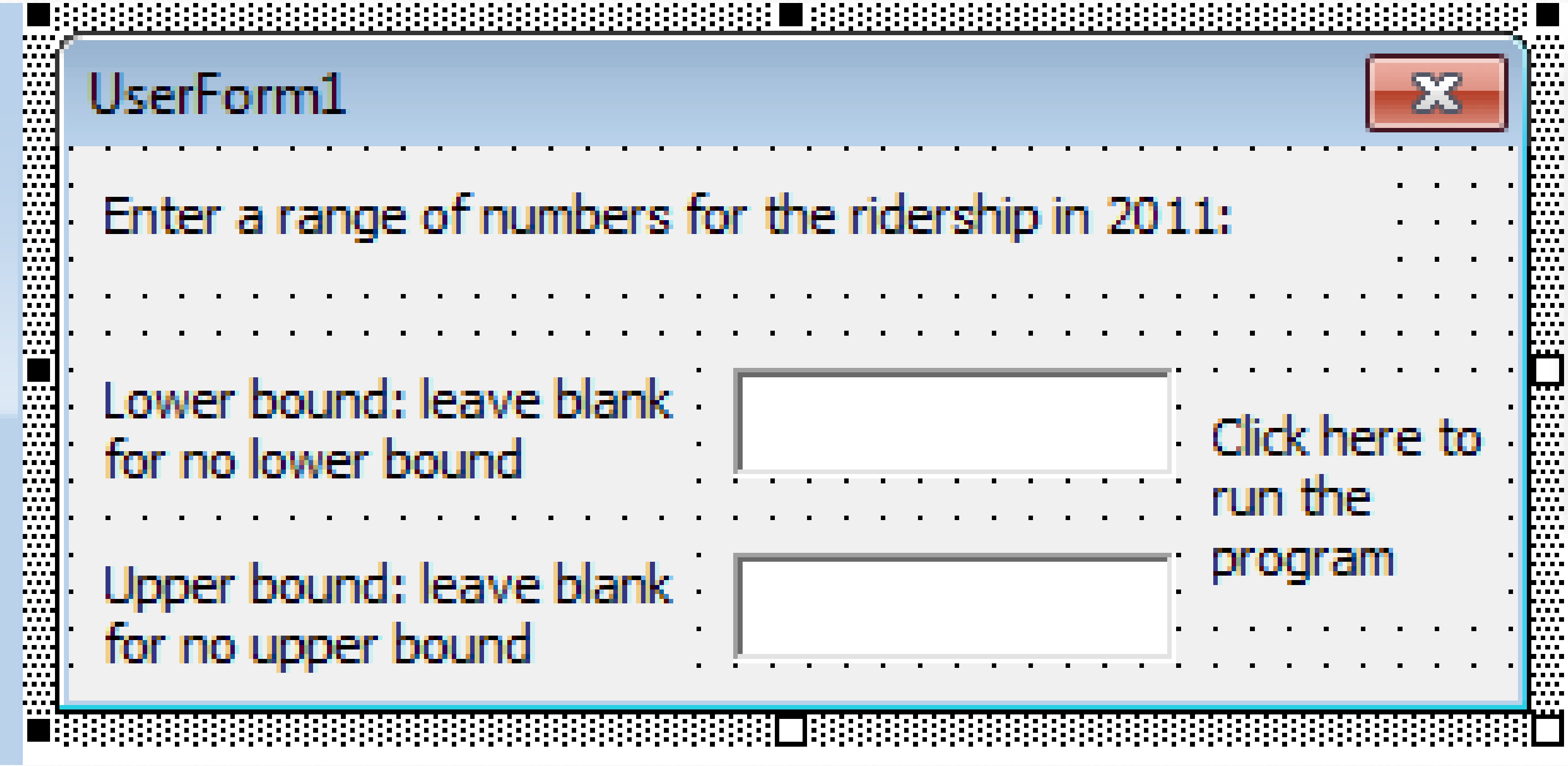
Task Bundle 3: Select Numbers

- Create an userform that allows the user to input two numbers to create a range
  - Allow for blank entries for unbounded
- Pull all of the name of stations whose 2011 ridership was between the user-chosen range
- Given the stations that fall into this criteria, create a chart sorting the stations by boroughs
  - Even though the second task did not specify that the borough be pulled also, it may seem helpful to do that as well, given this final task

Userform Design

- Which option in the Toolbox would be the most helpful?
- We want a form with the following features:
  - Text to display directions
    - Label merely with text
  - Two boxes for users to type numbers
    - TextBox
  - Button for user to click when done
    - Label with an option to run more programs when clicked

Userform Design



UserForm1

Enter a range of numbers for the ridership in 2011:

Lower bound: leave blank for no lower bound

Upper bound: leave blank for no upper bound

Programs for Clicking

```
Private Sub Label4_Click()
End Sub
```

- Need to write codes here:
  - Look through column G (2011 ridership stat) and find entries who value is between the range
  - Deal with unbounded values
  - Paste the station and borough name down column X and Y (arbitrarily chosen: merely our next empty columns)

Unbounded Values

- If TextBox1.Value is blank, we could set it to 0
- Similarly, if TextBox2.Value is blank, set to some enormous number like 999,999,999
- In general, this is not a good practice, especially when we don't know the extent of the data we're dealing with
- But the benefit is simplicity: otherwise we need more If-Else for when they are blank
- Since we roughly know the range of data here, acceptable to hardcode the values here

Unbounded Values

```
Private Sub Label4_Click()
If TextBox1.Value = "" Then
Min = 0
Else: Min = 1# * TextBox1.Value
End If

If TextBox2.Value = "" Then
Max = 999999999
Else: Max = 1# * TextBox2.Value
End If
```

Quick Review & Important Note

- In the previous slide, we would not have been able to use IsEmpty in replace of *TextBox1.Value = ""*
- By inputting nothing, we entered a blank entry, but not an empty entry
- Best way to capture blank entry is to test equality to ""
- In "*1# * TextBox1.Value*," it was necessary to include 1# * because otherwise, it would treat TextBox1.Value as a text, not number, and make inequity impossible to check

Traversing Data

- We could use ActiveCell and Offset to traverse down column G until ActiveCell is empty
  - “Select” this
- We then need another tracker for the data we’re pasting into column X and Y (24 and 25)
  - Do not “select” this
  - currentRow = currentRow + 1

```
Range("G3").Select
currentRow = 1
```

Traversing Data

```
Columns("X:Y").Clear
Do Until IsEmpty(ActiveCell)
If ActiveCell.Value >= Min And ActiveCell.Value <= Max Then
Cells(currentRow, 24) = ActiveCell.Offset(0, -6)
Cells(currentRow, 25) = ActiveCell.Offset(0, -5)
currentRow = currentRow + 1
End If
ActiveCell.Offset(1, 0).Select
Loop
```

Auto Fit Column & Hide Form

```
Columns(“X:Y").EntireColumn.AutoFit
UserForm1.Hide
End Sub
```

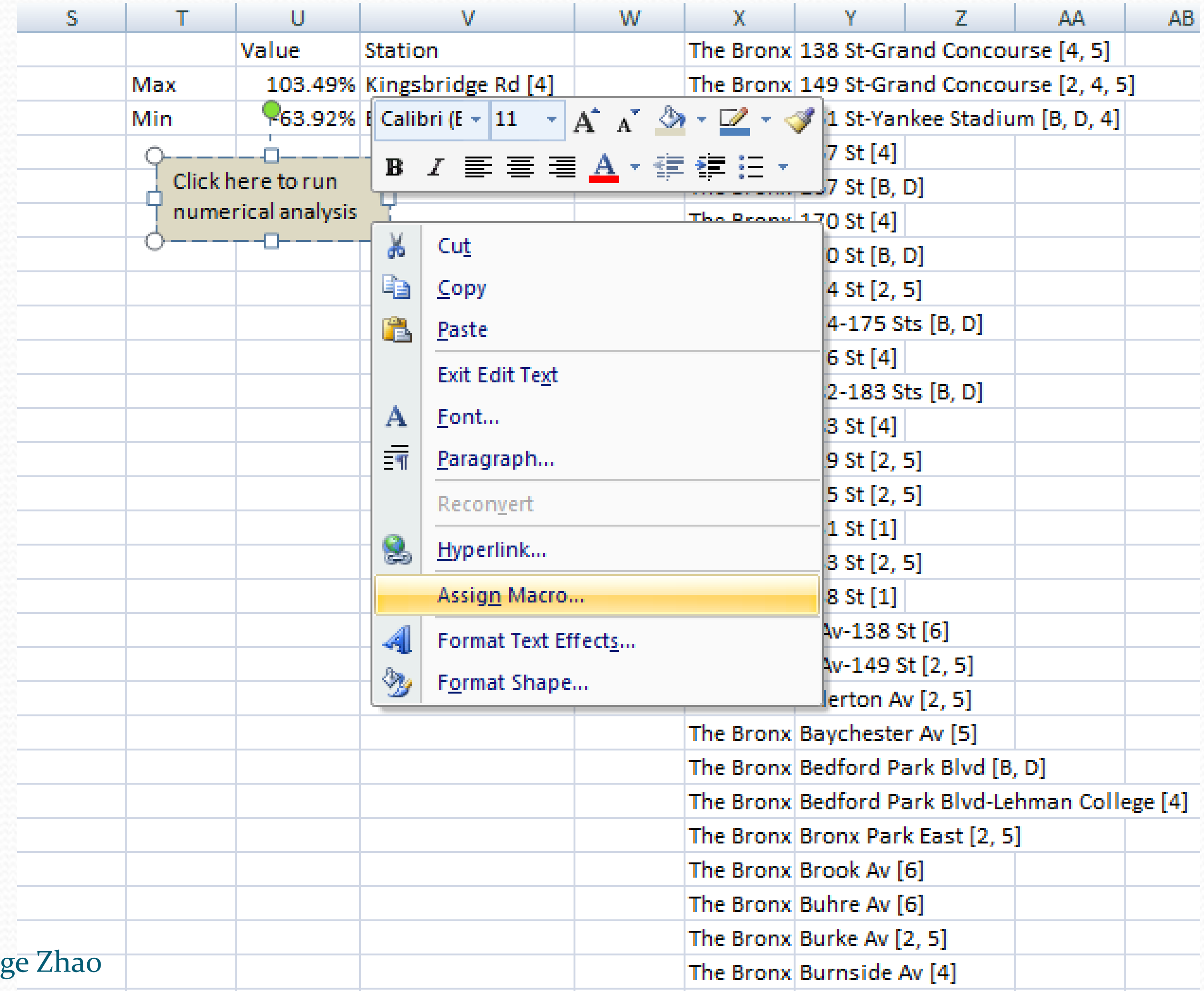
- Recording a simple macro can give the necessary syntax for the auto-fitting
- After recording the macro, recognize the need to change the parameters to columns X and Y
- Able to add other customization (font size, cell style, etc) this way (recall the tutorial on Recording Macros)

To Run This

- Recall these codes are within the userform
- We need the userform to be displayed
  - We can write a short code for that
  - Then simply assign an object (text box, etc) to run that short code upon being clicked

```
Sub Show()
UserForm1.Show
End Sub
```

Assign Macro to Objects



S	T	U	V	W	X	Y	Z	AA	AB
		Value	Station		The Bronx	138 St-Grand Concourse [4, 5]			
	Max	103.49%	Kingsbridge Rd [4]		The Bronx	149 St-Grand Concourse [2, 4, 5]			
	Min	63.92%	E			1 St-Yankee Stadium [B, D, 4]			
						7 St [4]			
						7 St [B, D]			
					The Bronx	170 St [4]			
						0 St [B, D]			
						4 St [2, 5]			
						4-175 Sts [B, D]			
						6 St [4]			
						2-183 Sts [B, D]			
						3 St [4]			
						9 St [2, 5]			
						5 St [2, 5]			
						1 St [1]			
						3 St [2, 5]			
						8 St [1]			
						Av-138 St [6]			
						Av-149 St [2, 5]			
						lerton Av [2, 5]			
					The Bronx	Baychester Av [5]			
					The Bronx	Bedford Park Blvd [B, D]			
					The Bronx	Bedford Park Blvd-Lehman College [4]			
					The Bronx	Bronx Park East [2, 5]			
					The Bronx	Brook Av [6]			
					The Bronx	Buhre Av [6]			
					The Bronx	Burke Av [2, 5]			
					The Bronx	Burnside Av [4]			

Click here to run numerical analysis

Final Task: Design the Chart

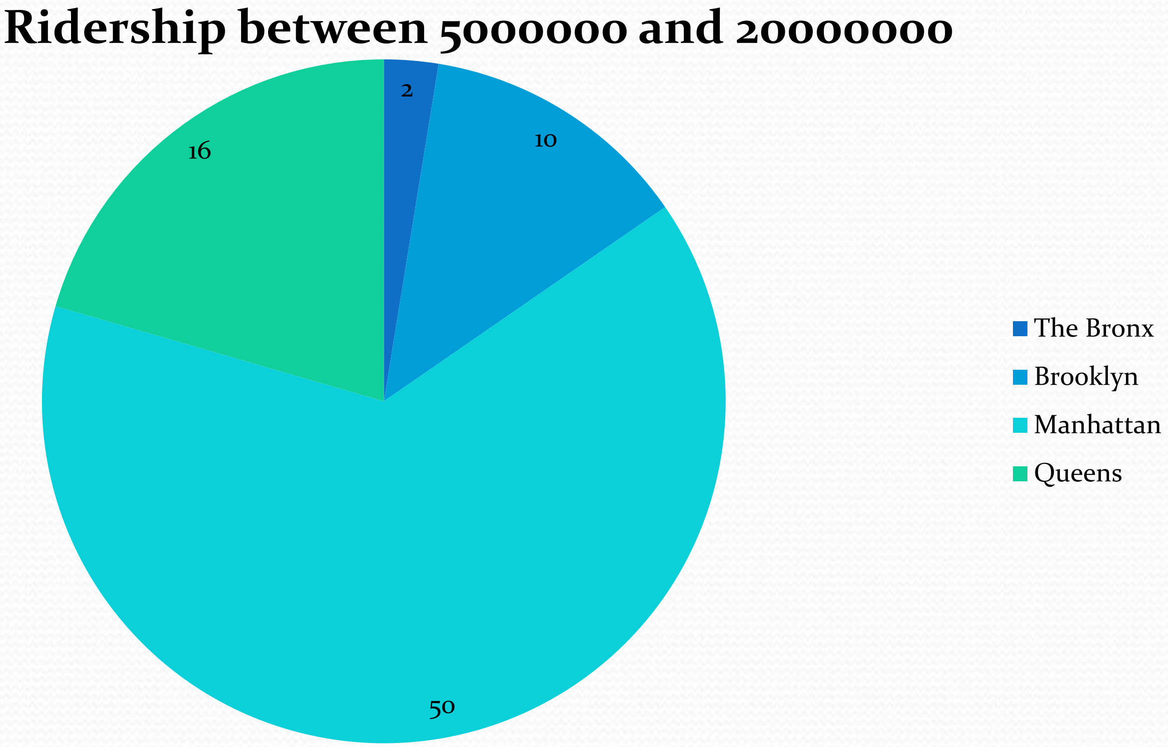
- Part one: pull data
  - Take the data from column X and Y (only really need X)
  - Count the frequency for each borough
  - Paste the frequency by borough in columns AA and AB
- Part two: create the chart
  - Create pie chart using the data in columns AA and AB
  - Record the macro to determine the formatting syntax
  - Data for the data will have fixed size – easier to automate

Pull Data

- Use =COUNTIF() method for each borough
  - Look up the borough names in column X
- Efficient coding:
  - We're repeating the same operation 4 times, with the only difference of the borough name
  - To avoid writing similar codes, use arrays to store the borough names

Create Chart

- Record a macro and manually create the desired chart



Recording Macro

```
Range("AA1:AB4").Select
ActiveSheet.Shapes.AddChart.Select
ActiveChart.SetSourceData Source:=Range("'Case 7'!$AA$1:$AB$4")
ActiveChart.ChartType = xlPie
ActiveChart.SeriesCollection(1).Select
ActiveChart.SeriesCollection(1).ApplyDataLabels
ActiveChart.ChartArea.Select
ActiveChart.SetElement (msoElementChartTitleCenteredOverlay)
ActiveChart.ChartTitle.Text = "Ridership between " & Min & " and " & Max
```

Other Materials

Assignments

- Short mini case studies, demonstrating the use of Excel and VBA to handle data in different ways
- Includes function syntaxes as references
  - More information includes than needed
  - Need to discern which functions will be useful in each case
- Snippets of each assignment shown in the following sides…

Assignment 1

- *At a fair, the participants were asked to input their name in the first column and their Columbia University UNI on the second column of an Excel document. Write the codes in VBA that can complete the task of automating the process by which the collected UNI's can be bundled together into one text, which can be directly pasted into an email to send out to all participants…*

Assignment 2

- *…the responses were typed up onto column A in an Excel worksheet. Participants may have used different cases of letters or added spaces between the words, but they refer to same entry. For example, the following entries should all be treated the same: “Coldplay”, “ColdPlay”, “Cold play”, “ c o Ld   p lAy   ”*
- *Write the codes in VBA that will create a sorted tally table that displays the distinct entries with their count in the dataset…*

Assignment 3

- *Suppose that at an institution, the unique ID code for each student is comprised of two letters (use lower-case in this exercise) of the alphabet, followed by one, two, or three digits of numbers. If two or more digits are present in the ID, the numerical portion of the ID will not begin with 0….*
- *Write a VBA that will generate all of the combinations of the ID's in column A, and then randomize the order…*

Final Marks

- This workshop was designed to give an overview of some of the noteworthy functions and the foundations of VBA programming.
- There are many more functionalities in Excel. One can always learn more by browsing through the functions list or searching for them online. For VBA, one can always record some macros.
- Having a solid foundation enables one to more easily learn new essentials.

About the Author

George Zhao graduated with Bachelor of Science in Operations Research from Columbia University School of Engineering and Applied Science in 2014.

http://www.columbia.edu/~gz2165