Spreadsheet Skills and Modelling

- A free, interactive(*) guide

This guide is available in two forms: As a spreadsheet and as a PDF document. Of the two versions the more useful and interesting one is the spreadsheet (XLS): The PDF version is a book but the spreadsheet version is an **interactive** book.

[Comments related to interactivity in this text relate to the spreadsheet version of this guide]*

These are the book's goals:

- To show the reader examples of the things that are possible with spreadsheets. Surprisingly, there are very few compilations like this of a set of wide-ranging and consistently presented spreadsheet examples.
- To allow the reader to master some of the most important spreadsheet functions. The guide explains how important functions work and gives examples of how they can be used.
- To let the reader test their mastery of the spreadsheet functions described in this book by carrying out a set of exercises.

The layout of this book is as follows.

Chapter 1
Chapter 1 presents a set of spreadsheet applications. The applications are broad in scope, some are fairly basic and others are more complex. Their purpose is to illustrate many of the capabilities - some of them not well known - of spreadsheets. Almost all of the examples are interactive.

Interactive sections are marked by colours. Cells with a blue background that look like this 6.30% the reader can change.

At the end of the chapter is a cross reference that lists the spreadsheet functions and features used in the earlier examples. The cross reference shows that even quite complex applications often use only a small number of spreadsheet functions (an average of eight - in our examples). Across all of the examples in this chapter seven out of nine spreadsheet functions are never used. Important insights flow from this: 1) Mastery of spreadsheets requires knowledge about a relatively small subset of spreadsheet functions, 2) Most other functions can be ignored.

Chapter 2
Most complex things are built from a set of simple and fundamental components. That's true also with spreadsheets. Each example in the preceding chapter was made by choosing about 8 functions from a set of 40.

In this chapter we look at the "building blocks" of spreadsheets - the individual functions that can be combined to make complex applications. We descibe what the functions do, how they are used and give examples of their applications.

Chapter 3
Individual spreadsheet functions - as described in the preceding chapter - serve specific and usually simple purposes. But what if we want to do something that an individual function can't? Then we need to combine two or more functions - in effect to design a "super-function". The more functions you need to combine - the harder this is to do. In this chapter we take first steps in this design process - by combining two or three functions to build "mini-applications".

Chapter 4
Having completed the earlier chapters you are now in a position to test your design skills in a more open-ended environment. You are given a set of problems to solve and you need to work out which functions to use and how to combine them to solve the problems.

Chapter 5
This chapter gives you links to our on-line resources whereby you can further advance your spreadsheet skills.

About this book
This book is aimed at those with intermediate levels of skills in spreadsheets - it is not a "beginner's" book. The concepts, examples and exercises range from intermediate to advanced levels. As with other free works on the internet some advertising is included. The advertising relates to workshops we run and services we offer.

Good luck! We hope you find this guide useful in increasing your expertise with spreadsheets and their applications. If, at some stage, you are interested in attending a spreadsheet or financial modelling workshop please consider the ones we offer. Also, please tell those you know who are interested in spreadsheets about this guide and our workshops.

Contents

Chapter 1 - Sample Spreadsheet Applications

Business and finance activities can often be described in terms of "processes". A process is a set of actions performed on some input to generate an output. Business and finance processes could involve aggregating, summarising, reporting, valuing, filtering, prioritising, ranking etc. In this section we look at examples of such business processes and show how they can be represented and implemented in spreadsheets.

The applications shown in this section have a finance emphasis but most can equally well be applied outside of the finance area. Each example lists the spreadsheet functions and features that were used to build the example. At the end of the chapter we review the spreadsheet functions and features used in the examples and draw some conclusions from those.

Aggregating

Overview

Aggregating involves grouping together or consolidating information. In the following example we aggregate transactions by grouping them into date intervals. We then show the number of transactions that occurred in each interval. The number of transactions is shown graphically - but in a somewhat unusual way - rather than using a chart we use conditional formatting.

Spreadsheet functions used in this example

Conditional format, IF, LEN, MAX, MIN, OFFSET, ROW, SUMPRODUCT

Transaction data

Date	Code	Amount
2-Sep	DII-34	36.74
7-Oct	HEJ-61	3.92
6-Nov	JFG-72	83.02
25-Oct	HGD-98	2.30
12-Nov	BIB-77	24.06
26-Nov	ADC-58	80.93
10-Oct	FED-17	69.58
14-Sep	FGD-28	28.15
16-Oct	EID-69	79.88

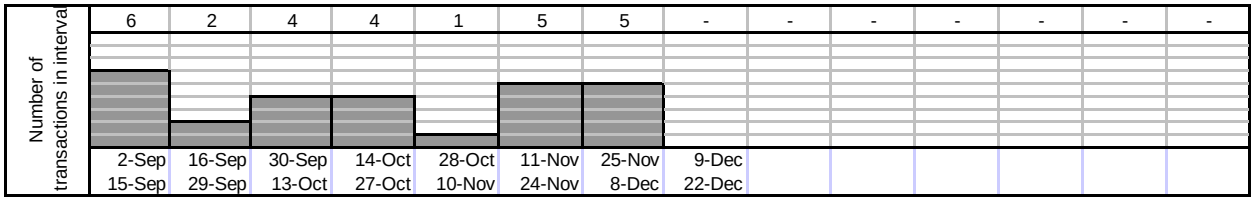
Date	Code	Amount
2-Sep	EAD-53	68.98
28-Nov	HFF-90	25.93
26-Nov	ICC-26	64.74
14-Sep	IHE-25	52.75
7-Sep	GIE-35	35.87
11-Oct	CAA-60	14.07
15-Nov	CJF-32	44.01
11-Nov	EGE-33	91.38
8-Oct	ACD-74	81.85

Date	Code	Amount
2-Sep	BIC-30	94.54
14-Oct	AHG-81	89.01
26-Nov	JHC-24	49.96
25-Nov	FFB-37	77.52
18-Oct	CGJ-33	45.08
25-Sep	FCI-84	65.10
15-Nov	HHE-97	58.66
11-Nov	ABD-80	87.32
22-Sep	HCG-33	88.07

Aggegation (grouping) interval

Grouping interval (days)	14

Aggegated results



Averaging

Averaging reduces the information content of a set of data and summarises that data in a concise but representative form. Averaging can be across various dimensions - including that of time. By averaging over time we reduce the "noiseiness" of data to discern trends and slower-moving features.

Our workshops review the finance principles and assumptions underlying the main spreadsheet financial functions.

In the following example we take a time-series and average it over 1 to 8 periods. Also, optionally, we highlight the times at which the averaged data "crosses" the source data.

Spreadsheet functions used in this example
AND, AVERAGE, Conditional format, IF, OFFSET, ROW

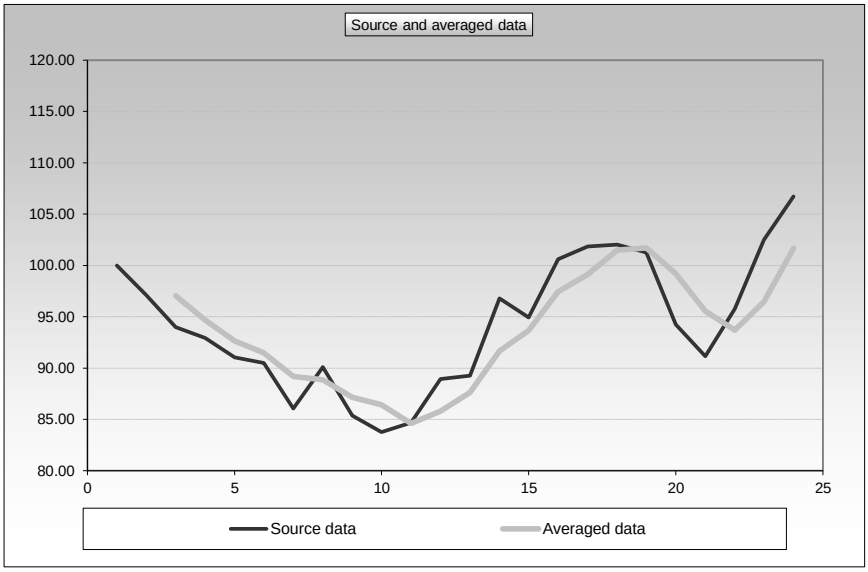
Average of a history of values.

Averaging period 3

☑ Highlight crossing points

Source data	Period	Value
	1	100.00
	2	97.13
	3	94.00
	4	92.92
	5	91.05
	6	90.51
	7	86.04
	8	90.08
	9	85.37
	10	83.76
	11	84.68
	12	88.94
	13	89.26
	14	96.79
	15	94.95
	16	100.59
	17	101.86
	18	102.01
	19	101.28
	20	94.24
	21	91.14
	22	95.76
	23	102.49
	24	106.72

Averaged data	Value
	0.00
	0.00
	97.04
	94.68
	92.66
	91.49
	89.20
	88.88
	87.17
	86.41
	84.60
	85.79
	87.62
	91.66
	93.67
	97.44
	99.14
	101.49
	101.72
	99.18
	95.55
	93.71
	96.46
	101.66



Back to contents

Back to top

Charting

Overivew

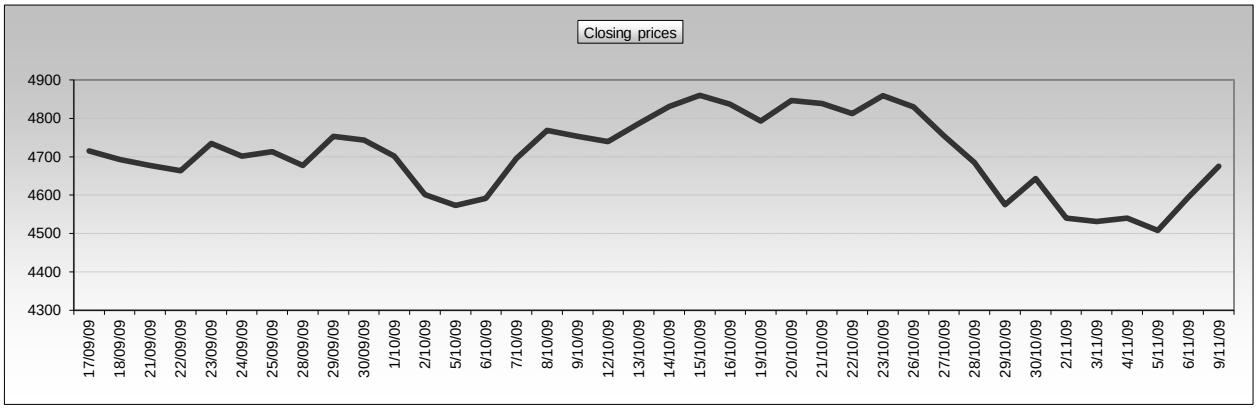
Charting makes it easier to see trends, patterns and anomalies in data than if the data were shown numerically. The following are simple examples of "dynamic" charts.

Spreadsheet functions used in these examples

Conditional format, MATCH, MAX, MIN, OFFSET, ROW, TEXT

Charting - values within a defined date range

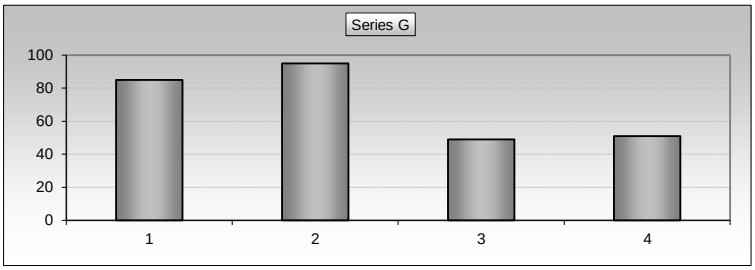
Start date	17/09/09
End date	9/11/09



Charting - one of a series

Series name	Data			
A	36	4	64	47
B	15	82	13	25
C	22	44	13	93
D	16	5	76	88
E	72	9	17	27
F	31	28	47	23
G	85	95	49	51
H	28	37	43	5
I	90	73	69	62
J	8	63	10	49
K	63	77	52	67
L	8	74	85	58

Series to chart	G



Back to contents

Back to top

Highlighting

Feedback from our Visual Basic course: "Good use of examples to solidify knowledge"

Overview

Spreadsheets often contain hundreds or thousands of numbers. Of those some may be more signficant than others. To emphasis those numbers they can be highlighted. Highlighting can be "static" (i.e. fixed) or it can be dynamic and can change automatically. The following examples all show dynamic formatting and highlighting.

Spreadsheet functions used in this example

AND, Conditional format, COUNTIF, LARGE, LEFT, MOD, OFFSET, ROW

Highlight top "N" amounts

Highlight the top "N" amounts: 0

Date	Transaction	Amount
1/12/09	DII-34	36.74
5/01/10	HEJ-61	3.92
4/02/10	JFG-72	83.02
23/01/10	HGD-98	2.30
10/02/10	BIB-77	94.06
24/02/10	ADC-58	80.93
8/01/10	FED-17	69.58
13/12/09	FGD-28	28.15
14/01/10	EID-69	79.88

Date	Transaction	Amount
1/12/09	EAD-53	68.98
26/02/10	HFF-90	25.93
24/02/10	ICC-26	64.74
13/12/09	IHE-25	52.75
6/12/09	GIE-35	35.87
9/01/10	CAA-60	95.67
13/02/10	CJF-32	44.01
9/02/10	EGE-33	91.38
6/01/10	ACD-74	81.85

Date	Transaction	Amount
1/12/09	BIC-30	14.54
12/01/10	AHG-81	89.01
24/02/10	JHC-24	49.96
23/02/10	FFB-37	77.52
16/01/10	CGJ-33	45.08
24/12/09	FCI-84	65.10
13/02/10	HHE-97	58.66
9/02/10	ABD-80	87.32
21/12/09	HCG-33	88.07

Highlight duplicate transactions

☐ Highlight duplicate transaction codes

Date	Transaction	Amount
1/12/09	DII-34	36.74
5/01/10	HEJ-61	3.92
4/02/10	JFG-72	83.02
23/01/10	HGD-98	2.3
10/02/10	BIB-77	94.06
24/02/10	ADC-58	80.93
8/01/10	FED-17	69.58
13/12/09	FFB-37	28.15
14/01/10	EID-69	79.88

Date	Transaction	Amount
1/12/09	EAD-53	68.98
26/02/10	HFF-90	25.93
24/02/10	ICC-26	64.74
13/12/09	DII-34	52.75
6/12/09	GIE-35	35.87
9/01/10	CAA-60	95.67
13/02/10	DII-34	44.01
9/02/10	EGE-33	91.38
6/01/10	FFB-38	81.85

Date	Transaction	Amount
1/12/09	BIC-30	14.54
12/01/10	AHG-81	89.01
24/02/10	JHC-24	49.96
23/02/10	FFB-38	77.52
16/01/10	CGJ-33	45.08
24/12/09	FCI-84	65.1
13/02/10	HHE-97	58.66
9/02/10	ABD-80	87.32
21/12/09	HCG-33	88.07

Highlight transactions that are similar

Highlight transactions beginning with <none>

Date	Transaction	Amount
1/12/09	DII-34	36.74
5/01/10	HEJ-61	3.92
4/02/10	JFG-72	83.02
23/01/10	HGD-98	2.30
10/02/10	BIB-77	94.06
24/02/10	ADC-58	80.93
8/01/10	FED-17	69.58
13/12/09	FGD-28	28.15
14/01/10	EID-69	79.88

Date	Transaction	Amount
1/12/09	EAD-53	68.98
26/02/10	HFF-90	25.93
24/02/10	ICC-26	64.74
13/12/09	IHE-25	52.75
6/12/09	GIE-35	35.87
9/01/10	CAA-60	95.67
13/02/10	CJF-32	44.01
9/02/10	EGE-33	91.38
6/01/10	ACD-74	81.85

Date	Transaction	Amount
1/12/09	BIC-30	14.54
12/01/10	AHG-81	89.01
24/02/10	JHC-24	49.96
23/02/10	FFB-37	77.52
16/01/10	CGJ-33	45.08
24/12/09	FCI-84	65.10
13/02/10	HHE-97	58.66
9/02/10	ABD-80	87.32
21/12/09	HCG-33	88.07

Highlight transactions on alternate lines

☐ Show highlights

Date	Transaction	Amount
1/12/09	DII-34	36.74
5/01/10	HEJ-61	3.92
4/02/10	JFG-72	83.02
23/01/10	HGD-98	2.30
10/02/10	BIB-77	94.06
24/02/10	ADC-58	80.93
8/01/10	FED-17	69.58
13/12/09	FGD-28	28.15
14/01/10	EID-69	79.88

Date	Transaction	Amount
1/12/09	EAD-53	68.98
26/02/10	HFF-90	25.93
24/02/10	ICC-26	64.74
13/12/09	IHE-25	52.75
6/12/09	GIE-35	35.87
9/01/10	CAA-60	95.67
13/02/10	CJF-32	44.01
9/02/10	EGE-33	91.38
6/01/10	ACD-74	81.85

Date	Transaction	Amount
1/12/09	BIC-30	14.54
12/01/10	AHG-81	89.01
24/02/10	JHC-24	49.96
23/02/10	FFB-37	77.52
16/01/10	CGJ-33	45.08
24/12/09	FCI-84	65.10
13/02/10	HHE-97	58.66
9/02/10	ABD-80	87.32
21/12/09	HCG-33	88.07

Back to contents

www.tykoh.com

Back to top

Filtering

Our modelling and spreadsheet skills courses can be done in one or two day modules.

Overview

On occasion we may want to focus on subsets of data rather than the whole data set. In other words - we may want to filter the data and see only the filtered result. The following is a simple example.

Spreadsheet functions used in this example

AND, IF, ISNA, MATCH, MAX, MIN, OFFSET, ROW, SUM

Source - unfiltered - data

Date	Transaction	Amount
1/12/09	DII-34	36.74
5/01/10	HEJ-61	3.92
4/02/10	JFG-72	83.02
23/01/10	HGD-98	2.30
10/02/10	BIB-77	24.06
24/02/10	ADC-58	80.93
8/01/10	FED-17	69.58
13/12/09	FGD-28	28.15
14/01/10	EID-69	79.88

Date	Transaction	Amount
1/12/09	EAD-53	68.98
26/02/10	HFF-90	25.93
24/02/10	ICC-26	64.74
13/12/09	IHE-25	52.75
6/12/09	GIE-35	35.87
9/01/10	CAA-60	14.07
13/02/10	CJF-32	44.01
9/02/10	EGE-33	91.38
6/01/10	ACD-74	81.85

Date	Transaction	Amount
1/12/09	BIC-30	94.54
12/01/10	AHG-81	89.01
24/02/10	JHC-24	49.96
23/02/10	FFB-37	77.52
16/01/10	CGJ-33	45.08
24/12/09	FCI-84	65.10
13/02/10	HHE-97	58.66
9/02/10	ABD-80	87.32
21/12/09	HCG-33	88.07

Filtering criteria

Show transaction that have values between	80	and	50

Filtered data

Date	Transaction	Value
8/01/10	FED-17	69.58
14/01/10	EID-69	79.88

Date	Transaction	Value
1/12/09	EAD-53	68.98
24/02/10	ICC-26	64.74
13/12/09	IHE-25	52.75

Date	Transaction	Value
23/02/10	FFB-37	77.52
24/12/09	FCI-84	65.10
13/02/10	HHE-97	58.66

Back to contents

www.tykoh.com

Back to top

Interest rates - Interpolating / valuing using / determining sensitivity to

Learn how to choose the best function to solve a problem when several functions are available by attending one of our workshops.

Overview

Interest (or discount) rates are fundamental in finance for they "tie together" the present and future values of money. Interest rates can be "flat" (e.g. interest rate on a 2 year loadn = interest rate on a 5 year loan) or they can be "curved" - have a term structure in other words.

The following example relates to interest rates that have a term structure. Three types of calculations are done:

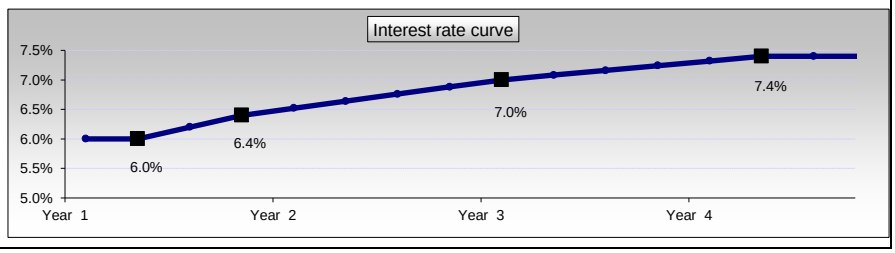
- An interest rate curve is constructed by interpolating from four given points.
- A series of cash flows are valued using the interpolated rate structure.
- One by one points on the rate curve are increased by 1%. The cash flows are revalued using the changed rate points. This gives the sensitivity of the present value of the cash flows to each point in the rate curve.

Spreadsheet functions used in this example

Arrays, Data Table, IF, MATCH, MAX, MOD, OFFSET, RIGHT, ROW, SUM

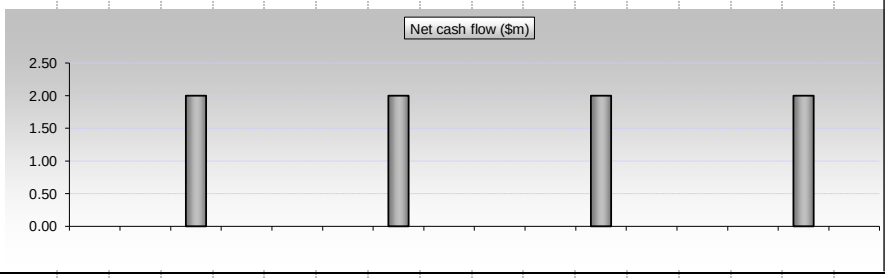
Interest rate curve - generating by interpolating using four points

	Year 1		Year 2	Year 3	Year 4
Interest rates (zero coupon):	6.0%	6.4%		7.0%	7.4%

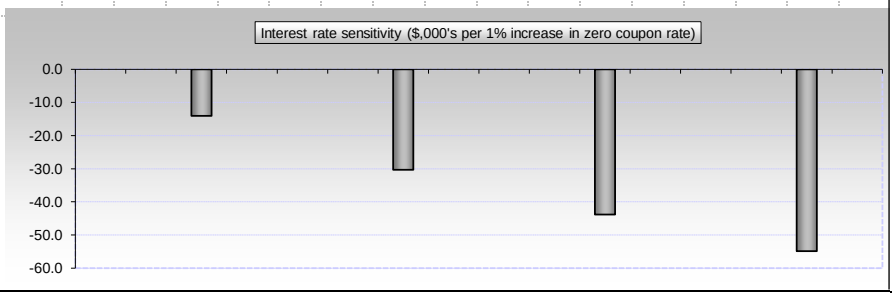


Payment schedule - sets timing and magnitude of cash flows

				Year 1				Year 2				Year 3				Year 4			
Amount	Number per year	First payment quarter	Term (years)	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
-	2	Q1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	1	Q4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	2	Q3	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	4	Q1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2	1	Q3	4	-	-	2	-	-	-	2	-	-	-	2	-	-	-	2	-
-	1	Q3	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-



Interest rate sensitivity - calculation of sensitivity of cash flow value to each point on the rate curve



Back to contents

Back to top

Interpolating

Feedback from our Visual Basic course: "The material covered will be very useful at work as the CD and workbook provided"

Overview

Interpolating involves "filling in the gaps" in numeric data. The examples below all use the same data set. Note that, even with the same data set, different interpolation methods can give significantly different answers. Which is the "correct" answer? The appropriate method to use depends on the context.

Spreadsheet functions used in these examples

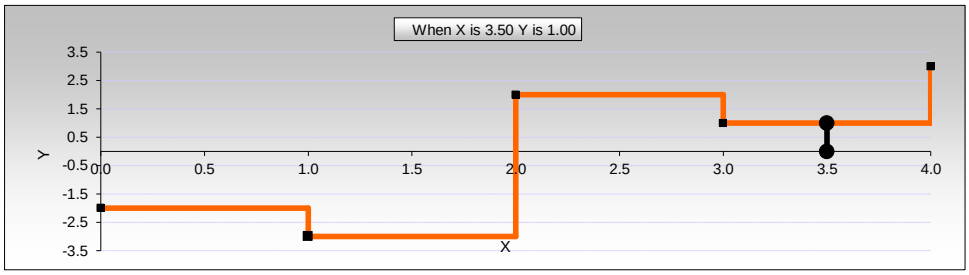
INT, LOOKUP, MATCH, MINVERSE, MMULT, OFFSET, ROW, TEXT, TREND

Stepped interpolation (as is done by LOOKUP, VLOOKUP and HLOOKUP)

Interpolation table	
x	*y*
0	-2
1	-3
2	2
3	1
4	3

X value to interpolate	3.5

When X is 3.50 Y is 1.00

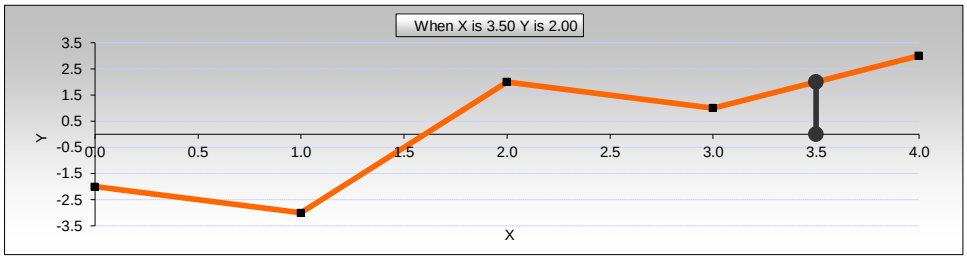


Piecewise linear interpolation. A combination of functions is needed to achieve this.

Interpolation table	
x	*y*
0	-2
1	-3
2	2
3	1
4	3

X value to interpolate	3.5

When X is 3.50 Y is 2.00

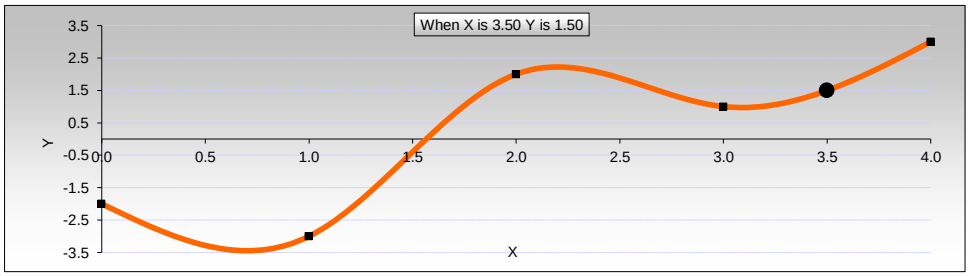


Cubic spline interpolation. A combination of functions is needed to achieve this.

Interpolation table	
x	*y*
0	-2
1	-3
2	2
3	1
4	3

X value to interpolate	3.5

When X is 3.50 Y is 1.50

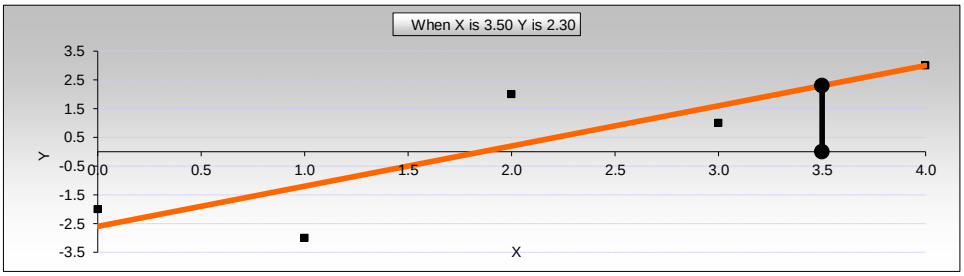


Statistical - line of best fit - interpolation

Data points	
x	*y*
0	-2
1	-3
2	2
3	1
4	3

X value to interpolate	3.5

When X is 3.50 Y is 2.30



Back to contents

Back to top

Looking up

Overview

The looking-up functions (VLOOKUP, HLOOKUP and LOOKUP) accept a single "key" and search for that key in a table. When the key is found a data item is retrieved from the table. If the key isn't found then either an error is returned or the nearest match is used (depending on the lookup). The following examples show applications of the VLOOKUP function. In the first example we find a single entry in a two-dimensional table. In the second example we find the sum of entries that meet a defined two-D criterion.

Spreadsheet functions used in this example

AND, Arrays, Conditional format, MATCH, MAX, MIN, OFFSET, ROW, SUMPRODUCT, VLOOKUP

2D lookup #1

In the area below choose a month and year. The data point for that month and year will be looked up.

Month to look up	Oct
Year to look up	2008
Answer	72

	2005	2006	2007	2008	2009
Jan	65	55	4	68	47
Feb	22	18	69	87	12
Mar	48	6	93	49	1
Apr	17	13	38	54	61
May	70	54	80	9	97
Jun	33	76	84	94	40
Jul	68	34	30	63	93
Aug	91	29	46	14	47
Sep	1	71	87	38	31
Oct	41	38	9	72	80
Nov	54	65	20	99	43
Dec	65	34	70	94	6

2D lookup #2

In the area below choose start and end months and years. The sum of the data points that lie between (and including) the start and end dates will be calculated.

Start month to look up	Oct
Start year to look up	2007
End month to look up	May
End year to look up	2009
Total of selected amounts:	1058

	2005	2006	2007	2008	2009
Jan	65	55	4	68	47
Feb	22	18	69	87	12
Mar	48	6	93	49	1
Apr	17	13	38	54	61
May	70	54	80	9	97
Jun	33	76	84	94	40
Jul	68	34	30	63	93
Aug	91	29	46	14	47
Sep	1	71	87	38	31
Oct	41	38	9	72	80
Nov	54	65	20	99	43
Dec	65	34	70	94	6

Prioritising

Overview

A dollar used for one purpose cannot be used for another. So a common finance task is allocating funds to competing purposes. A prioritising or ranking or allocating mechanism is needed to ensure each dollar is most appropriately applied. There are many possible allocation mechanisms. The examples below illustrate two. In each example there are five "claims" or "needs" (A through E).

Our courses are presented internationally.

In the first example claim A has the highest priority and claim B cannot be dealt with until a defined amount has been allocated to A. Similarly claims B through E can be serviced only when the prior and higher-ranking claims have been honored. In the second example available funds are "pro-rata'd" across all claims.

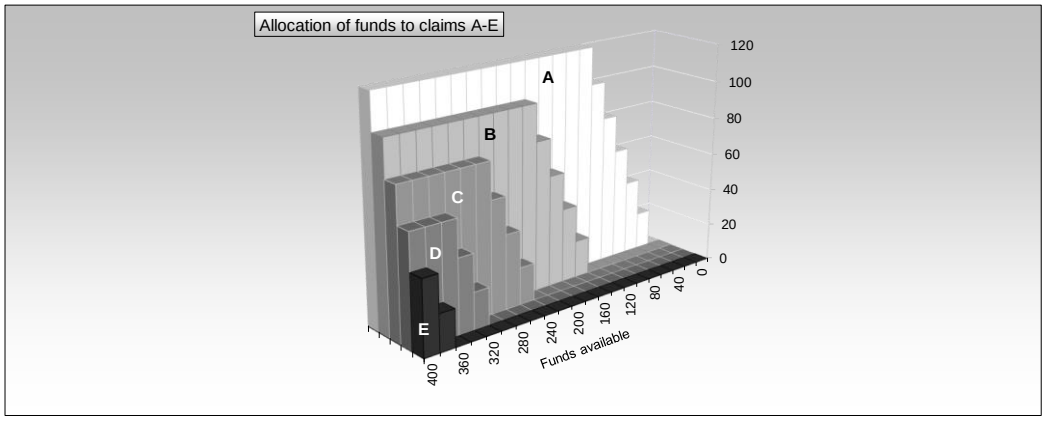
Spreadsheet functions used in these examples

IF, MIN, OFFSET, ROW, SUM

Claim A has highest priority, claim E has lowest.

	Funds available							
	400	320	160	240	320	400	140	260

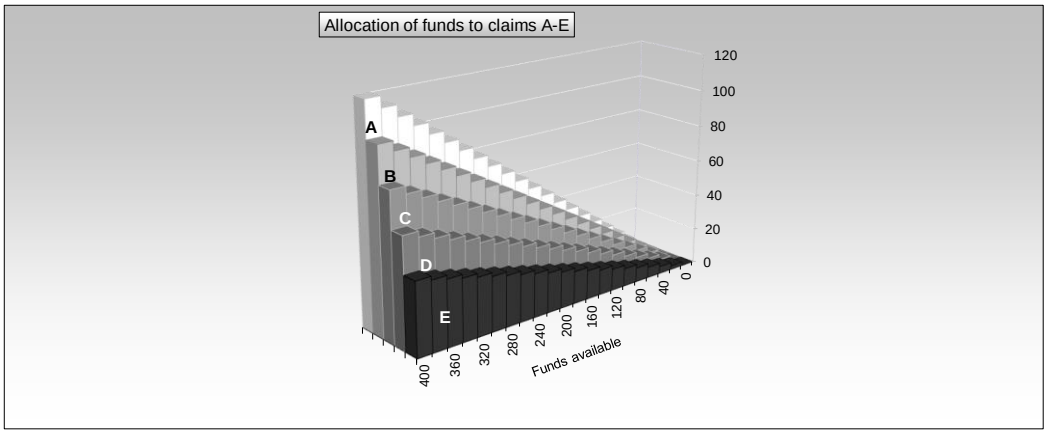
Claim	Requirement	Allocations of funds to claims A-E							
A	120	120	120	120	120	120	120	120	120
B	100	100	100	40	100	100	100	20	100
C	80	80	80	0	20	80	80	0	40
D	60	60	20	0	0	20	60	0	0
E	40	40	0	0	0	0	40	0	0



Claims A - E have equal priority and are serviced on a pro-rata'd basis

	Funds available							
	0	40	150	380	120	300	350	400

Claim	Requirement	Allocations of funds to claims A-E							
A	30%	0	12	45	114	36	90	105	120
B	25%	0	10	37.5	95	30	75	87.5	100
C	20%	0	8	30	76	24	60	70	80
D	15%	0	6	22.5	57	18	45	52.5	60
E	10%	0	4	15	38	12	30	35	40



Ranking

XNPV gives a different answer if cells are blank than when they contain zero - these and other "traps" are explained in our workshops

Overview

In ranking a set of items we need to define a "measure". The measure allows us to compare one item with another and to say which ranks higher. In the following example data is ranked according to three different criteria.

Spreadsheet functions used in this example

CHOOSE, CODE, IF, LARGE, MATCH, MID, OFFSET, RIGHT, ROW, TEXT

Source data

Date	Transaction	Amount
1/12/09	DII-34	36.74
5/01/10	HEJ-61	3.92
4/02/10	JFG-72	83.02
23/01/10	HGD-98	2.30
11/02/10	BIB-77	24.06
24/02/10	ADC-58	80.93
8/01/10	FED-17	69.58
13/12/09	FGD-28	28.15
14/01/10	EID-69	79.88

Date	Transaction	Amount
2/12/09	EAD-53	68.98
26/02/10	HFF-90	25.93
25/02/10	ICC-26	64.74
15/12/09	IHE-25	52.75
6/12/09	GIE-35	35.87
9/01/10	CAA-60	14.07
12/02/10	CJF-32	44.01
10/02/10	EGE-33	91.38
6/01/10	ACD-74	81.85

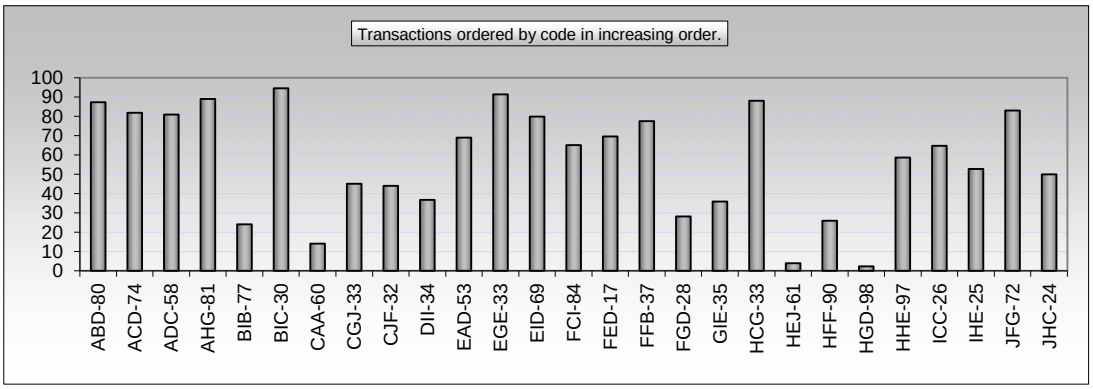
Date	Transaction	Amount
4/12/09	BIC-30	94.54
12/01/10	AHG-81	89.01
22/02/10	JHC-24	49.96
23/02/10	FFB-37	77.52
16/01/10	CGJ-33	45.08
22/12/09	FCI-84	65.10
13/02/10	HHE-97	58.66
9/02/10	ABD-80	87.32
21/12/09	HCG-33	88.07

Ranking criteria

Order by: Transaction code

- (•) In increasing order
- ( ) In decreasing order

Ranked data



Back to contents

Back to top

Rounding

Our Visual Basic course shows how to increase work efficiency by making shortcuts for common tasks

Overview
Spreadsheets (and computers generally) represent decimal numbers only to a finite accuracy. This can cause arithmetic and logical errors. We can deal with these errors by using rounding functions.

Spreadsheet functions used in this example
IF, OFFSET, ROUND, ROW

Example of a rounding issue
Consider the formulae A and B below. Even though they should give the same answers - they don't: .

	Formula	*Result*
A	=1/10 - 1/10 + 1/10000	0.000100000000000000
B	=1/10 + 1/10000 -1/10	0.000100000000000003

The results differ in the eighteenth decimal place. Such differences can occur because spreadsheets store decimal fractions only to a finite accuracy and not to an infinite accuracy.

Arithmetic "errors" like the above can accumulate in calculations. Often these effects are benign but on occasion they can cause LOOKUPs, IF statements and the like to "fail" (i.e. to give unexpected results).

For example, if you were to use an IF statement to test whether formulae A and B above were equal the IF statement would say A and B **are not equal** (as shown below).

Result	*Formula*
Different	=IF(1/10 + 1/10000 - 1/10 = 1/10 - 1/10 + 1/10000, "Same", "Different")

We can prevent these arithmetic effects from causing unexpected results by using rounding. This is illustrated by the following modification to the earlier example. This modified version uses the ROUND function.

Result	Formula
Same	=IF(ROUND(1/10 + 1/10000 - 1/10,6) = ROUND(1/10 - 1/10 + 1/10000,6), "Same" ,"Different")

Valuing / Scenarios / Attributing

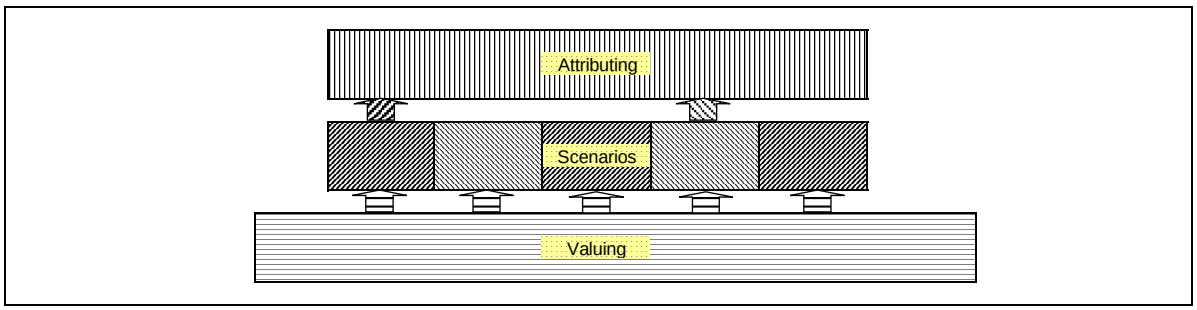
Our Visual Basic course shows how to extend or change Microsoft Office products' functionality

Overview

This example is built on three levels. The lowest level is concerned with **valuing** an enterprise based on a number of assumptions. The enterprise is valued on the basis of finding the present value of the free (i.e. available) cash flows it is expected to generate.

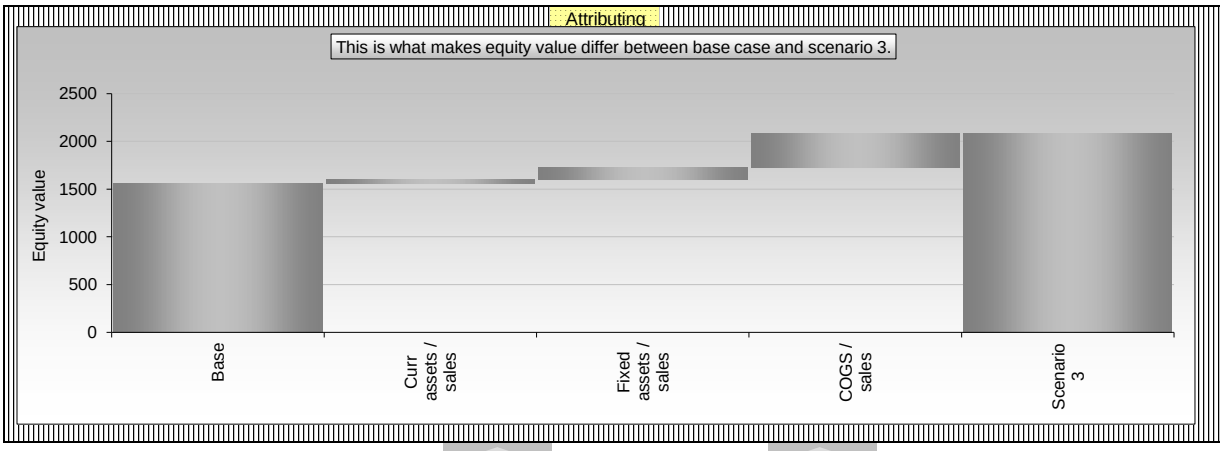
The middle level consists of a number of **scenarios**. Each scenario differs in its individual assumptions and each scenario generates an associated enterprise value.

The highest level is concerned with **attributing**. Here we look at how the enterprise value changes between two scenarios. We "explain" that change in terms of the assumptions that differ between the scenarios.



Spreadsheet functions used in this example

ABS, AND, Conditional format, COUNT, Data Table, EOMONTH, IF, MATCH, MAX, MIN, OFFSET, OR, ROW, SUM, XNPV



Please choose a scenario -> 3

Scenarios

Scenario	Chosen	Base	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
Annual sales growth	10%	10%	10%	8%	10%	18%	18%	18%
Current assets / sales	10%	12%	12%	12%	10%	12%	12%	12%
Current liabilities / cost of goods sold	6%	6%	6%	6%	6%	6%	6%	6%
Net fixed assets / sales	65%	70%	70%	70%	65%	70%	70%	70%
Cost of goods sold / sales	50%	55%	55%	55%	50%	55%	55%	55%
Depreciation rate	15%	15%	15%	19%	15%	20%	20%	20%
Interest rate on debt	8%	8%	8%	8%	8%	8%	8%	8%
Interest earned on cash balance	5%	5%	5%	5%	5%	5%	5%	5%
Tax rate	33%	33%	35%	30%	33%	22%	27%	26%
Dividend payout ratio	55%	55%	55%	55%	55%	55%	55%	55%
Discount rate	18%	18%	18%	18%	18%	18%	18%	18%
Terminal value multiple (TVM)	12	12	11	10	12	12	12	12
Output / Equity value		1,563	1,405	1,276	2,082	2,005	1,739	1,792

Valuing

Year	31/12/08	31/12/09	31/12/10	31/12/11	31/12/12	31/12/13	31/12/14	31/12/15
Income statement								
Sales	1,000	1,100	1,210	1,331	1,464	1,611	1,772	1,949
Cost of goods sold	(500)	(550)	(605)	(666)	(732)	(805)	(886)	(974)
Interest payments on debt	(26)	(26)	(26)	(26)	(26)	(26)	(26)	(26)
Interest earned on cash balance	6	2	13	16	20	24	29	34
Depreciation	(100)	(116)	(107)	(118)	(130)	(143)	(157)	(173)
Profit before tax	380	411	485	538	596	661	732	810
Taxes	(126)	(135)	(160)	(177)	(197)	(218)	(241)	(267)
Profit after tax	255	275	325	360	400	443	490	543
Dividends	(140)	(151)	(179)	(198)	(220)	(244)	(270)	(298)
Earnings retained	115	124	146	162	180	199	221	244
Balance sheet								
Cash	32	252	319	394	478	572	676	793
Non-cash current assets	150	110	121	133	146	161	177	195
Fixed assets								
at cost	1,070	1,131	1,309	1,506	1,722	1,960	2,222	2,510
accumulated depreciation	(300)	(416)	(523)	(641)	(770)	(913)	(1,070)	(1,243)
net fixed assets	770	715	787	865	952	1,047	1,152	1,267
Total assets	952	1,077	1,226	1,392	1,576	1,779	2,005	2,254
Current liabilities	32	33	36	40	44	48	53	58
Debt	320	320	320	320	320	320	320	320
Stock	450	450	450	450	450	450	450	450
Accumulated retained earnings	150	274	420	582	762	961	1,182	1,426
Total liabilities and equity	952	1,077	1,226	1,392	1,576	1,779	2,005	2,254
Cash flow statement								
Cash flow from operating activities								
Profit after tax		275	325	360	400	443	490	543
Plus depreciation		116	107	118	130	143	157	173
Changes in working capital								
less increase in current assets		40	(11)	(12)	(13)	(15)	(16)	(18)
plus increase in current liabilities		1	3	4	4	4	5	5
Net cash from operating activities		432	424	470	520	575	636	703
Cash flow from investing activities								
Capex		(61)	(179)	(197)	(216)	(238)	(262)	(288)
Net cash used in investing activities		(61)	(179)	(197)	(216)	(238)	(262)	(288)
Cash flow from financing activities								
Dividends paid		(151)	(179)	(198)	(220)	(244)	(270)	(298)
Net cash from financing		(151)	(179)	(198)	(220)	(244)	(270)	(298)
Net increase in cash		220	67	75	84	94	105	117
Free cash flow								
Year	**31/12/08**	**31/12/09**	**31/12/10**	**31/12/11**	**31/12/12**	**31/12/13**	**31/12/14**	**31/12/15**
Profit after tax		275	325	360	400	443	490	543
Plus depreciation		116	107	118	130	143	157	173
Less increase in current assets		40	(11)	(12)	(13)	(15)	(16)	(18)
Plus increase in current liabilities		1	3	4	4	4	5	5
Less increase in fixed assets at cost		(61)	(179)	(197)	(216)	(238)	(262)	(288)
Plus after tax interest on debt		17	17	17	17	17	17	17
Less after tax interest on cash		(1)	(8)	(11)	(13)	(16)	(19)	(23)
Free cash flow		387	254	280	308	338	372	410
Valuation								
Free cash flow (FCF)	-	387	254	280	308	338	372	410
Terminal value multiple								12
Terminal value [FCF * multiple]	-	-	-	-	-			4,914
Total	-	387	254	280	308	338	372	5,324
NPV	2,370							
Plus cash	32							
Enterprise value	2,402							
Less debt	(320)							
Equity value	2,082							

Back to contents

Back to top

Scheduling events / cash flows

Feedback from our Financial Modelling course: "Understanding the complex Excel tips was very beneficial"

Overview

Activities and events often proceed in a certain order. In spreadsheets the order of activities and events can be set in various ways: 1) By "hard-coding" them on a fixed timeline, 2) by using formulae to define prerequisites (e.g. event B in row 26 depends on event A in row 25), 3) in the way shown below. The example below implements a dynamic way of defining event dependencies and durations. In the example below a task can proceed only when its "prerequisite" has completed.

One challenge in managing a set of prerequisites is to deal with the possibility of "circularity". If you try to define a circular set of prerequisites below (e.g. A's prerequiste is B and B's prerequisite is A) you will see how this particular example deals with such a condition.

Spreadsheet functions used in this example

AND, IF, ISNA, LEFT, LEN, MATCH, MAX, MOD, OFFSET, OR, ROW, VLOOKUP

Messages
No messages

			Event Schedule																												
Task	*Length*	*Prerequisite*	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A	1	C																														
B	4	D																														
C	4	<none>																														
D	4	E																														
E	2	F																														
F	2	G																														
G	2	<none>																														

Back to contents

Back to top

Seasonalising

Learn how to choose the best function to solve a problem when several functions are available by attending one of our workshops.

Overview

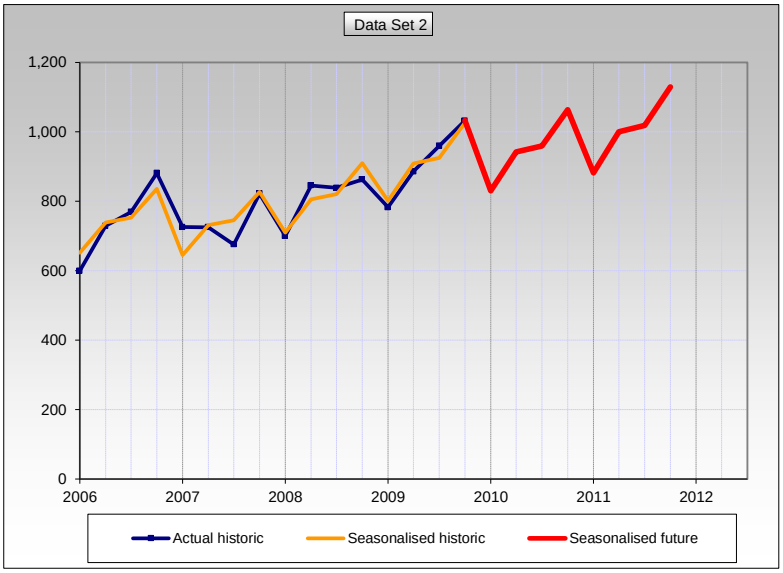
In seasonalising we determine patterns and trends in historic data. We can then "back-fill" the actual historic data with seasonalised historic data to see how well the seasonalised data fits the historic. We can also roll-forward the seasonalised data to predict future values. An example is given below.

Spreadsheet functions used in this example

AVERAGE, CHOOSE, Conditional format, IF, OFFSET, ROW, SUM, TEXT, TREND

Choose data set	2
Show seasonalised historic data	☑
Show seasonalised future data	☑

	Data sets		
	1	*2*	*3*
2006 Q1	7,932	600	200
Q2	6,537	729	270
Q3	6,353	770	226
Q4	7,782	881	194
2007 Q1	8,091	726	210
Q2	6,528	725	192
Q3	6,770	676	165
Q4	7,866	822	151
2008 Q1	8,375	700	146
Q2	6,914	846	164
Q3	6,938	838	143
Q4	8,393	863	130
2009 Q1	8,948	783	135
Q2	8,288	886	141
Q3	7,960	960	141
Q4	9,709	1,032	156



Back to contents

Back to top

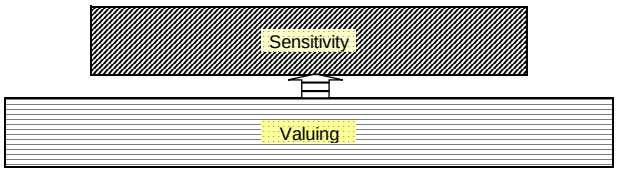
Valuing / Sensitising

Our Visual Basic course shows how to extend or change Microsoft Office products' functionality

Overview

This example is built on two levels. The lower level is concerned with valuing an enterprise based on a number of assumptions. The enterprise is valued on the basis of finding the present value of the free (i.e. available) cash flows it

The upper level determines the sensitivity of a nominated output (equity value) to the assumptions it depends on. Each assumption is changed by 1% and the resulting change in equity value is calculated. The resulting percentage change in value are then ranked in order of significance and shown in a chart. If a sensitivity is shown in pink on the right hand part of the chart that means it is a "positive" sensitivity. A positive sensitivity means that increasing the assumption increases the equity value. If a sensitivity is shown in blue on the right hand side it represents a "negative" sensitivity. A negative sensitivity means that increasing the assumption decreases the equity value.



Spreadsheet functions used in this example

ABS, AND, CHOOSE, Data Table, EOMONTH, IF, LARGE, MATCH, OFFSET, ROW, SUM, XNPV

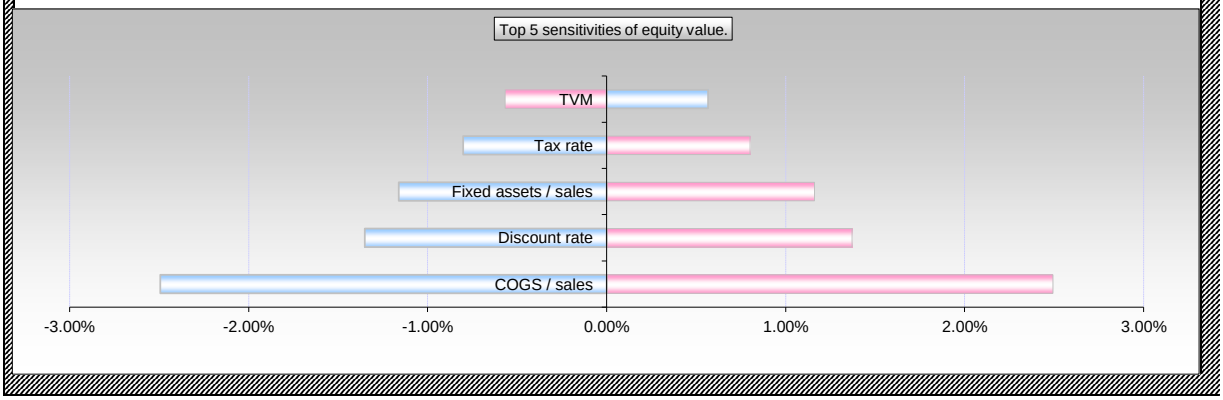
Assumptions

Annual sales growth	10%	Discount rate	18%	Interest rate on debt	8%
Current assets / yearly sales	12%	Depreciation rate	15%	Interest earned on cash	5%
Current liabilities / cost of goods sold	6%	Tax rate	33%	Cost of goods sold / sales	55%
Net fixed assets / yearly sales	70%	Dividend payout ratio	55%	Terminal value multiple (TVM)	12

Sensitivities

Sensitivity

Number of most significant sensitivities to show in chart: 5



Valuing

Income statement

Year	31/12/09	31/12/10	31/12/11	31/12/12	31/12/13	31/12/14	31/12/15	31/12/16	31/12/17	31/12/18
Sales	1,000	1,100	1,210	1,331	1,464	1,611	1,772	1,949	2,144	2,358
Cost of goods sold	(550)	(605)	(666)	(732)	(805)	(886)	(974)	(1,072)	(1,179)	(1,297)
Interest payments on debt	(26)	(26)	(26)	(26)	(26)	(26)	(26)	(26)	(26)	(26)
Interest earned on cash balance	6	2	8	10	12	14	17	20	23	27
Depreciation	(100)	(116)	(116)	(127)	(140)	(154)	(169)	(186)	(205)	(225)
Profit before tax	330	356	411	456	506	560	620	685	758	838
Taxes	(109)	(117)	(136)	(151)	(167)	(185)	(204)	(226)	(250)	(276)
Profit after tax	221	238	276	306	339	375	415	459	508	561
Dividends	(122)	(131)	(152)	(168)	(186)	(206)	(228)	(253)	(279)	(309)
Earnings retained	100	107	124	138	152	169	187	207	229	253

Balance sheet										
Cash	32	161	199	241	289	343	403	470	545	629
Current assets	150	132	145	160	176	193	213	234	257	283
Fixed assets										
at cost	1,070	1,186	1,378	1,590	1,823	2,079	2,361	2,671	3,012	3,387
accumulated depreciation	(300)	(416)	(531)	(658)	(798)	(952)	(1,121)	(1,307)	(1,511)	(1,736)
net fixed assets	770	770	847	932	1,025	1,127	1,240	1,364	1,501	1,651
Total assets	952	1,063	1,191	1,333	1,490	1,663	1,855	2,068	2,303	2,562
Current liabilities	32	36	40	44	48	53	58	64	71	78
Debt	320	320	320	320	320	320	320	320	320	320
Stock	450	450	450	450	450	450	450	450	450	450
Accumulated retained earnings	150	257	381	519	671	840	1,027	1,233	1,462	1,715
Total liabilities and equity	952	1,063	1,191	1,333	1,490	1,663	1,855	2,068	2,303	2,562

Cash flow statement										
Cash flow from operating activities										
Profit after tax		238	276	306	339	375	415	459	508	561
Plus depreciation		116	116	127	140	154	169	186	205	225
Changes in working capital										
less increase in current assets		18	(13)	(15)	(16)	(18)	(19)	(21)	(23)	(26)
plus increase in current liabilities		4	4	4	4	5	5	6	6	7
Net cash from operating activities		376	382	422	467	516	570	630	695	768
Cash flow from investing activities										
Capex		(116)	(193)	(212)	(233)	(256)	(282)	(310)	(341)	(375)
Net cash used in investing activities		(116)	(193)	(212)	(233)	(256)	(282)	(310)	(341)	(375)
Cash flow from financing activities										
Dividends paid		(131)	(152)	(168)	(186)	(206)	(228)	(253)	(279)	(309)
Net cash from financing		(131)	(152)	(168)	(186)	(206)	(228)	(253)	(279)	(309)
Net increase in cash		129	37	42	48	54	60	67	75	84

Free cash flow										
Year	**31/12/09**	**31/12/10**	**31/12/11**	**31/12/12**	**31/12/13**	**31/12/14**	**31/12/15**	**31/12/16**	**31/12/17**	**31/12/18**
Profit after tax		238	276	306	339	375	415	459	508	561
Plus depreciation		116	116	127	140	154	169	186	205	225
Less increase in current assets		18	(13)	(15)	(16)	(18)	(19)	(21)	(23)	(26)
Plus increase in current liabilities		4	4	4	4	5	5	6	6	7
Less increase in fixed assets at cost		(116)	(193)	(212)	(233)	(256)	(282)	(310)	(341)	(375)
Plus after tax interest on debt		17	17	17	17	17	17	17	17	17
Less after tax interest on cash		(1)	(5)	(7)	(8)	(10)	(11)	(13)	(16)	(18)
Free cash flow		277	201	221	243	267	294	323	356	391

Valuation										
Free cash flow (FCF)		277	201	221	243	267	294	323	356	391
Terminal value multiple										12
Terminal value [FCF * multiple]	-	-	-	-	-	-	-	-	-	4,697
Total		277	201	221	243	267	294	323	356	5,088
NPV	1,871									
Plus cash	32									
Enterprise value	1,903									
Less debt	(320)									
Equity value	1,583									

Back to contents

Back to top

Valuing contingencies

Our Visual Basic course shows how to automate routine work and save the time and trouble of doing the same thing over and over

Overview

To put a value on contingencies we need to take probabilities into account (e.g. how likely is the contingency to occur). An example of a contingency is an option position. Whether the option pays off is contingent on the performance of an underlying asset. Fortunately Black & Scholes developed a formula for pricing some options that saves us many difficult probability calculations. The Black / Scholes formula is used in the example below.

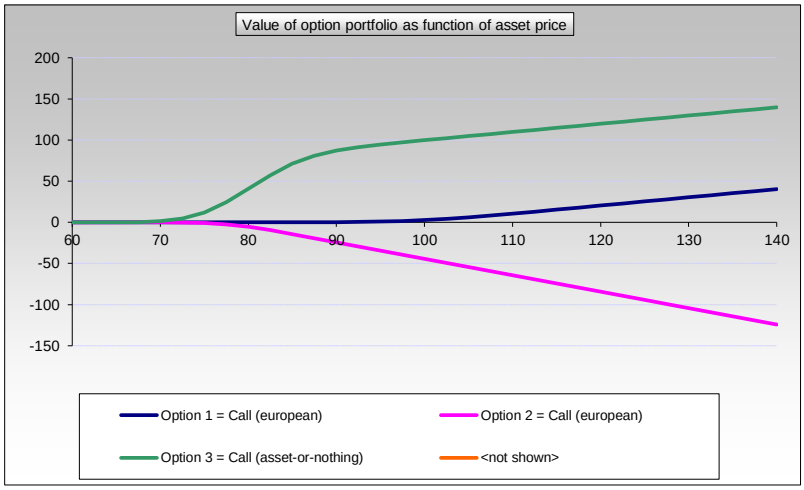
A call option gives you the right - but not the obligation - to buy something at a future date at a price that is set today. The price at which the purchase will be done (if it is done) is called the exercise price. A put option gives you the right to sell something at a future date at a price that is set today. Again, the price at which the sale will be done (if it is done) is the exercise price.

Spreadsheet functions used in this example

CHOOSE, Data Table, EXP, IF, LN, MATCH, MAX, NORMSDIST, OFFSET, ROW, SQRT, SUM

Option type	*Exercise price*	*Maturity (in years)*	*Position [number held / (sold)]*	*Show on chart*
Call (european)	100	0.1	1	☑
Call (european)	78	0.05	-2	☑
Call (asset-or-nothing)	80	0.1	1	☑
Total				☐

Volatility	20%
Yield	1.5%
Risk free rate	5.5%



Back to contents

Back to top

Visual Basic

Our financial modelling course covers the finance and accounting concepts that underlay financial modelling

Visual Basic is part of most Microsoft Office products and it gives you the power to:

- Automate routine work and save the time and trouble of doing the same thing over and over
- Increase work efficiency by making shortcuts for common tasks
- Customize Microsoft Office products to your needs by extending or changing those products' functionality
- Integrate workflows across Microsoft Office applications
- Increase the "wow-factor" of your developed applications

Visual Basic is not used in any of the examples in this spreadsheet. However, if you are interested in some examples of Visual Basic's application or wish to enrol on a Visual Basic workshop please visit the following web site:

Visual Basic Workshop

Cross reference of functions used in the examples / Summary

Our Visual Basic course shows how to increase the "wow-factor" of your developed applications

Overview

Below is a cross reference showing the spreadsheet functions and features used in each of the examples in this chapter
Several interesting observations can be made from the information in the cross reference:

- The examples use only a small number of functions. The average number of functions used in each example is approximately 9.

- The total number of functions used in all the examples is 40. That is about 12% of the functions available. In other words seven of every eight functions available were not used in any of the examples.

Cross reference

		Functions and features used *(Functions in blue are described in Chapter 3)*																																						
Example	num functions used	ABS	AND	Arrays	AVERAGE	CHOOSE	CODE	Conditional format	COUNT	COUNTIF	Data Table	EOMONTH	EXP	IF	INT	ISNA	LARGE	LEFT	LEN	LN	LOOKUP	MATCH	MAX	MID	MIN	MINVERSE	MMULT	MOD	NORMSDIST	OFFSET	OR	RIGHT	ROUND	ROW	SQRT	SUM	SUMPRODUCT	TEXT	TREND	VLOOKUP	XNPV
Aggregating	8							o						o					o				o		o					o				o			o				
Averaging	6		o		o			o						o																o				o							
Charting	7							o														o	o		o					o				o				o			
Highlighting	8		o					o		o							o	o										o		o				o							
Filtering	9		o											o		o						o	o		o					o				o		o					
Interpolating	9														o						o	o				o	o			o				o				o	o		
Looking up	10		o	o				o														o	o		o					o				o			o			o	
Prioritising	5													o											o					o				o		o					
Ranking	10					o	o							o			o					o		o						o		o		o				o			
Rounding	4													o																o			o	o							
Scenarios	15	o	o					o	o		o	o		o								o	o		o					o	o			o		o					o
Scheduling	12		o											o		o		o	o			o	o					o		o	o			o						o	
Seasonalising	9				o	o		o						o																o				o		o		o	o		
Sensitivities	12	o	o			o					o	o		o			o					o								o				o		o					o
Interest rates	10			o							o			o								o	o					o		o		o		o		o					
Valuing contingency	12					o					o		o	o						o		o	o						o	o				o	o	o					

Summary

Even though some of the examples in this spreadsheet are complex we can see that they are built from a small number of functions. So where does the complexity and power and usefulness of the examples come from? It comes from the **arrangement** of individually simple functions into a complex whole. This points to the importance and benefits of **design** - being able to build something complex from things that are simple.

Another lesson from the examples is the benefit of **focusing** ones' learning on a subset of spreadsheet functions: In these examples approximately 40 functions out of 320 are particularly useful to know about. The other 280 - odd might be applicable only infrequently.

Chapter 2 - Functions

This chapter contains a description of some spreadsheet functions. As with the rest of this book - some sections are interactive. Most if not all of the functions described in this chapter are used in the application examples in Chapter 2.

Functions

What do we mean by the word function? A function is part (or all) of a spreadsheet formula. Consider the formula below.

=D2 + SUM(A1:B2) * C2

In the formula above **SUM** is a function. You can tell it's a function because it consists of letters of the alphabet immediately followed by an open bracket. Anytime you see letters of the alphabet followed by a open bracket - you know you're looking at a function.

What does a function do? That depends on the function. Different functions do different things. The SUM function adds numbers, the AVERAGE function averages numbers, the CHOOSE function lets you choose between numbers, and so on.

If you're using the SUM function to sum numbers how does the function "know" which numbers you want to add? You tell it by giving it the numbers directly [as in =SUM(1, 2, 56)] or by giving it cell references that contain the numbers [as in =SUM(A1, A3, B6:G7)].

So to use a function you almost always pass it information for it to work with. That information is contained in a list of **parameters** or **arguments.**

In the SUM example above **A1:B2** describes the numbers we want to sum. So **A1:B2** is the argument of the SUM function. In this example the SUM function has a single argument. If a function has more than one argument then they are separated by commas. Consider the following example.

=SUM(1, A1:F4, 2 * 3, B2)

This function has four arguments. The first is 1, the second is A1:F4, the third is 2 *3 and the fourth is B2.

Now that we've introduced the concept of functions and arguments we'll describe some of the most useful spreadsheet functions. In this chapter the functions are ordered alphabetically.

ABS function

The ABS function returns the absolute value of its argument. The absolute value of a number is obtained by removing any minus sign it might have. The argument to the ABS must be a single number or cell reference. If the argument is positive then the number is returned unchanged. If the number is negative then the positive version of that number is returned (i.e. the sign is removed).

Where would you use an ABS function? We'll look at an example to do with calculating buying and selling commissions. In the illustration below we will calculate a brokerage fee to pay on a sale of $300,000. We will use a convention whereby cash coming in is a positive number and cash going out is negative. Because we are selling something that will be cash coming in and so we'll represent that as a positive number.

The brokerage calculation is in cell D6. The brokerage is calculated by multiplying the brokerage percentage fee by the amount of the sale or purchase. Both the percentage fee and the sale amount are positive numbers. But since the brokerage fee is a cash outgoing we need to make that negative. So there is a negative sign at the front of the formula in D6.

	A	B	C	D	E	F	G	H
1								
2	Brokerage fee		[%]	2%				
3								
4	Sale / (purchase)		[$,000's]	300				
5								
6	Brokerage		[$,000's]	-6	<- =-D2*D4			
7								

*[Cell E6 is a comment. It shows the formula in the cell to its left. So by looking at cell E6 we can tell that the formula in D6 is - D2 * D4 . You'll notice the D2 and D4 cells in the E6 formula are colour coded. The cells E6 refers to are also colour coded with coloured borders: Cell D2 has a red border and D4 has a blue border. The colour coding makes it easy to see which cells in the spreadsheet a formula refers to.]*

Continuing with our example - We find the brokerage fee to be $6,000 and it is a negative number.

Suppose we now use our spreadsheet to calculate the brokerage on a **purchase** of $400,000. The number in D4 is now negative since a purchase is a cash outlay.

	A	B	C	D	E	F	G	H
1								
2	Brokerage fee		[%]	2%				
3								
4	Sale / (purchase)		[$,000's]	-400				
5								
6	Brokerage		[$,000's]	8	<- =-D2*D4			
7								

The brokerage is now positive. According to the spreadsheet our broker will pay us to do the transaction. This is obviously wrong. The reason it's wrong is that the brokerage is always a cost irrespective of whether we're buying or selling. In other words, the broker is concerned only with the **absolute value** of our transaction and isn't concerned with whether it's a cash inflow or outflow for us.

So this is an opportunity to use the ABS function. The spreadsheet below has been corrected and redesigned to use the ABS function. Look at the formula in D6. We pass the transaction amount (in D4) to the ABS function. So irrespective of whether we're buying or selling a positive number is always returned.. The result of the ABS function is multiplied by the brokerage percentage. A minus sign at the front of the formula then converts the result to a cost.

	A	B	C	D	E	F	G	H	I
1									
2	Brokerage fee		[%]	2%			2%		
3									
4	Sale / (purchase)		[$,000's]	300			-400		
5									
6	Brokerage		[$,000's]	-6	<- =-D2*ABS(D4)		-8	<- =-G2*ABS(G4)	
7									

Notice that the brokerage fee calculation is now correct (i.e. is a cost - a cash outlay) for both sales (in column D) and purchases (in column G).

Things you can't do with an ABS function

The example below shows some things you can't do with an ABS function.

The formula in E2 attempts to find the ABS of a range of cells A2:C2. But that's not possible to do and so an error is returned.

The formula in B4 attempts to find the ABS of a non-numeric item - and again - an error is returned.

	A	B	C	D	E	F	G	H
1								
2	3	4	-5		#VALUE!	<- =ABS(A2:C2)		
3								
4	Limit	#VALUE!	<- =ABS(A4)					
5								
6								

Participants on our courses have the opportunity to do on-line pre and post-course self-evaluations.

Back to contents

Back to top

AND function

The AND function accepts a number of TRUE and FALSE parameters and returns TRUE if all of the parameters are TRUE. It returns FALSE otherwise.

	A	B	C	D	E	F	G	H
1								
2	TRUE	<- =AND(TRUE,TRUE)						
3								
4	FALSE	<- =AND(TRUE,FALSE)						
5								
6	FALSE	<- =AND(FALSE,FALSE)						
7								

Following is an interactive example. You can set A1, C1 and E1 to TRUE or FALSE. The AND function in A4 will return TRUE only when A1, C1 and E1 are all TRUE.

	A	B	C	D	E	F	G	H
1	TRUE		TRUE		FALSE			
2								
3								
4	FALSE	<- =AND(A1,C1,E1)						
5								

Logical tests and conditions

In the example above the arguments to the AND function are explicitly TRUE or FALSE. In practice the TRUE and FALSE arguments will be the result of logical tests or conditions - as illustrated in the example below.

	A	B	C	D	E	F	G	H
1	TRUE	<- =E1<G1			5		7	
2								
3	FALSE	<- =E3>G3			5		7	
4								

Cell A1 contains the logical test E1<G1. Logical tests involve comparing two quantities and determining whether they are equal to each other, or one is less than the other, one is greater than or equal to the other, and so on. The result of the logical test is TRUE or FALSE. The test in A1 is TRUE because E1 (5) is less than G1 (7). The test in A3 is FALSE because E3 is not greater than G3.

The items being compared in logical tests can be text ("strings"). This is illustrated in the following example. Strings are compared on the basis of their alphabetical order. So, for example, "London" is less than "Singapore".

	A	B	C	D	E	F	G	H
1	TRUE	<- =E1<G1			London		Singapore	
2								
3	TRUE	<- =E3<G3			Dept-H		Dept-Q	
4								
5	FALSE	<- =E5<G5			New York		45	
7								

It doesn't really make sense to compare numbers with strings and it's difficult to imagine why anyone would want to do this - but it can be done - as illustrated in the formula in A5 above.

In the illustration below we use an AND function to determine whether each number in the G column is "between" the numbers in the E and I columns. Only on row 2 are both tests in the AND satisfied and so only on row 2 is G between E and I.

	A	B	C	D	E	F	G	H	I
1					Num 1		Num 2		Num 3
2	TRUE	<- =AND(E2<G2,G2<I2)			3		7		9
3									
4	FALSE	<- =AND(E4<G4,G4<I4)			3		9		7
5									
6	FALSE	<- =AND(E6<G6,G6<I6)			9		3		7
7									

The following interactive example relates to budgeting. We want to know whether the actual figures in both 2008 and 2009 were within budget. You can change the actual figures and observe whether the "Within budget" result is as you'd expect.

	A	B	C	D	E	F	G	H
1		2008	2009					
2	Budget	15	18					
3								
4	Actual	14	17					
5								
6	Within budget:			TRUE	<- =AND(B4<=B2,C4<=C2)			
7								

The following example builds from the previous one. The result of the AND function is TRUE or FALSE. We pass the TRUE or FALSE as the first argument into an IF function. The IF function returns "within" or "outside" depending on whether the AND returns TRUE or FALSE.

	A	B	C	D	E	F	G	H
1		2008	2009					
2	Budget	15	18					
3								
4	Actual	14	19					
5								
6	Within budget:		outside	<- =IF(AND(B4<=B2,C4<=C2),"within","outside")				
7								

OR function

The OR function accepts a number of TRUE and FALSE parameters and returns TRUE if any of the parameters are TRUE. It returns FALSE otherwise.

A simple example below shows an application of the OR function. We want to "flag" in column E when one (or both) of two things happened in a year: 1) Growth was negative, 2) the result wasn't within budget.

	A	B	C	D	E	F	G	H	I
1	Year	Growth	Within budget		Flag				
2	1997	5%	yes						
3	1998	3%	yes						
4	1999	10%	yes						
5	2000	-12%	yes		x	<- =IF(OR(B5<0,C5="no"),"x","")			
6	2001	5%	yes						
7	2002	3%	yes						
8	2003	5%	no		x	<- =IF(OR(B8<0,C8="no"),"x","")			
1	2004	7%	yes						
2	2005	-3%	no		x				
3	2006	2%	yes						
4	2007	6%	yes						
5	2008	1%	yes						
6	2009	-2%	yes		x				
7									

We use an OR function to test whether growth (in column B) was less than zero or whether the "within budget" figure (in column C) was "no". If either or both of those are true then the OR function will return TRUE. And in that case the IF function will set a "x" marker in the flag column (column E).

Back to contents

Back to top

AVERAGE function

The AVERAGE function returns the average of one or more arguments. The arguments can contain individual numbers or cells or groups of cells. Some usages of the average function are shown below.

In cell D1 we use the AVERAGE function to find the average of two individual cells: A1 contains 1 and C1 contains 3 and so the average is 2.

In cell D3 we use AVERAGE to find the average of a block of cells: A3:C3. Note that cell B3 is empty and that AVERAGE is ignoring that cell. So the calcuation it is making is this: =(1 + 3)/2 = 2. If AVERAGE had included B3 in its calculation it would have arrived at a different result: =(1 + 0 + 3)/3 = 1.333.

In cell D5 AVERAGE is working with a similar set of cells as it did on row 3. However, in this example, the "middle" cell has 0 explicitly in it instead of being empty. We can see that this changes the result of the average from 2 to 1.3333

	A	B	C	D	E	F	G	H
1	1		3	2	<- =AVERAGE(A1,C1)			
2								
3	1		3	2	<- =AVERAGE(A3:C3)			
4								
5	1	0	3	1.333333	<- =AVERAGE(A5:C5)			
6								
7				#DIV/0!	<- =AVERAGE(A7:C7)			

In cell D7 AVERAGE's argument is a range that contains no numbers at all. In this case AVERAGE's calculation is this: =0/0. You cannot divide by zero and so the error occurs.

[For an example of the AVERAGE function used with the OFFSET function click here]
[For an example of an application that uses AVERAGE click here]

Back to contents

Back to top

CHOOSE function

We specialise in presenting one and two day finance / technical workshops

Overview

The CHOOSE is used to select one of several options. The CHOOSE function's first argument determines which of the following arguments are returned. If the first argument is 1 then the first following argument (i.e. the second) is returned. If the first argument is 2 then the second following argument (i.e. the third) is returned.

A CHOOSE function can be used in scenario analysis. For example, the user could make the CHOOSE function return an optimistic, neutral or pessimistic value for a growth rate.

In the following example the CHOOSE function's first argument is 1. So the first following argument (i.e. the second - A1) is returned.

	A	B	C	D	E	F	G	H
1	45		82		17			
2								
3	45	<- =CHOOSE(1,A1,C1,E1)						
4								

In the following example the CHOOSE function's first argument is 2. So the second following argument (i.e. the third - C1) is returned.

	A	B	C	D	E	F	G	H
1	45		82		17			
2								
3	82	<- =CHOOSE(2,A1,C1,E1)						
4								

Choosing a scenario

The following example shows a CHOOSE function being used to choose one of three scenarios. This example is interactive. The cell with a blue background can be changed. The "output" is the cell with the black background.

	A	B	C	D	E	F	G	H
1	Scenario:	1 - Neutral		2 - Optimistic		3 - Pessimistic		
2		45		82		17		
3								
4	Choose a	1		Result:	45	<- =CHOOSE(B4,B2,D2,F2)		
5	scenario:							

Our Visual Basic course shows how to increase work efficiency by making shortcuts for common tasks

If the first argument is greater than the number of remaining arguments then an error is returned. In this case the CHOOSE runs off the end of the list. That is illustrated next.

	A	B	C	D	E	F	G	H
1	45		82		17			
2								
3	#VALUE!	<- =CHOOSE(4,A1,C1,E1)						
4								

The most arguments a CHOOSE (or indeed any spreadsheet function) can have is 30. So if you want to choose between 30 or more values you will not be able to use the CHOOSE function. In that case you could probably use the OFFSET function.

You can't use the first argument as "index" into a block of cells and attempt to retrieve the indexed cell. If you try to do that an error will be returned. This is illustrated in the following example.

	A	B	C	D	E	F	G	H
1	45	82	17					
2								
3	#VALUE!	<- =CHOOSE(2,A1:C1)						
4								

Our Visual Basic course shows how to extend or change Microsoft Office products' functionality

One might expect the CHOOSE to return 82 in this case (i.e. the second number in the A1:C1 range). But CHOOSE works a different way. The way it interprets "block" arguments (i.e. arguments that refer to a block of cells rather than just a single cell) is illustrated next.

In the following example the CHOOSE function has four arguments. The first argument is 2. Since the first argument is 2 the CHOOSE function returns the second following argument. The second following argument is A3:C3. So the CHOOSE function returns A3:C3. A3:C3, in turn, is passed to the SUM function and the SUM function returns the sum of the values in that range (4 + 5 + 6 = 15).

	A	B	C	D	E	F	G	H
1	1	2	3					
2								
3	4	5	6					
4								
5	7	8	9					
6								
7	15	<- =SUM(CHOOSE(2,A1:C1,A3:C3,A5:C5))						
8								

So now we can see the reason for the earlier #VALUE! error. The error arose when the range returned by the CHOOSE function (A1:C1) didn't "fit" into the single cell we tried to put it in.

You can experiment with the CHOOSE function in the following example.

	A	B	C	D	E	F	G	H
1	Choose:	1		1	2			
2								
3		5	6		-4			
4								
5	7	8	9		-8			
6								
7	Result:	3	<- =SUM(CHOOSE(B1,D1:E1,B3:C3,A5:C5,E3,E5))					
8								
9								

We can tailor courses to specific audiences - e.g. audiences with a derivatives focus.

COUNT function

Learn how to choose the best function to solve a problem when several functions are available by attending one of our workshops.

The COUNT function counts the number of numbers in a range. Consider the following example. The COUNT function in B5 is counting the number of numbers in the range B2:D3. B2 and C3 are numbers. C2 is an error and B3 is text. D2 and D3 are empty. So only two of the six cells contain numbers - and that is what COUNT returns.

	A	B	C	D	E	F	G	H
1								
2		3	#DIV/0!					
3		above	4					
4								
5		2	<- =COUNT(B2:D3)					
6								

The following example illustrates how the COUNT function can be used. In the example we're checking that all of the items in a range are actually numbers.

Feedback from our Visual Basic course: "The material covered will be very useful at work as the CD and workbook provided"

	A	B	C	D	E	F	G	H
1								
2	3	4	5					
3	6	7	8					
4	9	10	11					
5	12	13	14					
6								
7	ok	<- =IF(ROWS(A2:C5)*COLUMNS(A2:C5)<>COUNT(A2:C5),"Error","ok")						
8								

This is how the checking formula in B7 works. We begin by finding the number of rows in the range by using the ROWS function. We multiply the number of rows by the number of columns to find the number of cells in our range. Then we compare that number with that returned by the COUNT function. If the two numbers are different there has been an error and the IF function returns "Error". Otherwise it returns "Ok".

You might say that this test is redundant since it is obvious whether the cells are all numbers or not. But that's not true. Compare the example above with the one below. They appear the same but the second is in error. Why is that? Try to observe the cause. After that, if you click on the "Show explanation" checkbox below an explanation will be shown.

Review how financial statements can be modelled in spreadsheets - attend one of our workshops

	A	B	C	D	E	F	G	H
1								
2	3	4	5					
3	6	7	8					
4	9	10	11					
5	12	13	14					
6								
7	Error	<- =IF(ROWS(A2:C5)*COLUMNS(A2:C5)<>COUNT(A2:C5),"Error","ok")						
8								

☐ Show explanation

COUNTIF function

The COUNTIF function is used to count conditionally - to count only items that meet a defined criterion. The function takes two arguments. The first argument is reference to a range of cells. These are the cells to be counted conditionally. The second argument is a test or condition that needs to be met for cells to be counted.

The example below shows how the COUNTIF function can be used: We have a list of departmental "performance levels". Performance levels are A,B,C or D. We want to count the number of times an "A" performance has occurred.

Consider the COUNTIF function in cell E4. Its first argument - C2:H2 - describes the range of cells that will be counted. These cells contain the list of performance levels. The second argument describes the counting criterion: We want to count "A" performances. There are two ways of specifying the criterion. The first way is to have the criterion as "A" (as used in E4) or as "=A" as shown in E6.

	A	B	C	D	E	F	G	H
1			Oct-09	Nov-09	Dec-09	Jan-10	Feb-10	Mar-10
2	Dept. Performance:		A	B	C	B	D	C
3								
4	Number of times had "A" performance:				1	<- =COUNTIF(C2:H2,"A")		
5								
6	Number of times had "A" performance:				1	<- =COUNTIF(C2:H2,"=A")		
7								

The COUNTIF function returns the number of "A" performances (1).

What if we wanted to count the number of "B" and lower performances? The following example shows how to do that. The second argument of the COUNTIF function is ">=B". This will make the function count all the performances that are B and also all the performances that are alphabetically greater than B (i.e. C, D, etc).

	A	B	C	D	E	F	G	H
1			Oct-09	Nov-09	Dec-09	Jan-10	Feb-10	Mar-10
2	Dept. Performance:		A	B	C	B	D	C
3								
4	Number of times "B" or worse performance:				5	<- =COUNTIF(C2:H2,">=B")		
5								

Rather than "hard-coding" the second argument (i.e. explicitly giving the performance level(s) we wish to count) we can have a cell reference as the second argument. As in the following example. Here the second argument is the cell reference D4. In this case the COUNTIF function will count all the performance levels that equal that shown in cell D4.

	A	B	C	D	E	F	G	H
1			Oct-09	Nov-09	Dec-09	Jan-10	Feb-10	Mar-10
2	Dept. Performance:		A	B	C	B	D	C
3								
4	Choose performance level:			A				
5								
6	Number of times had chosen performance:				1	<- =COUNTIF(C2:H2,D4)		
7								

If we want to count performances greater than or equal to that shown in D4 then we do that the following way. We "hard-code" the ">=" and follow that with the "&" symbol. The "&" is a "joining" or "concatenation" operator. And the last part of the second argument is the cell reference D4.

	A	B	C	D	E	F	G	H
1			Oct-09	Nov-09	Dec-09	Jan-10	Feb-10	Mar-10
2	Dept. Performance:		A	B	C	B	D	C
3								
4	Choose performance level:			C				
5								
6	Number of times had chosen			3	<- =COUNTIF(C2:H2,">="&D4)			
7	performance or worse:							

The following example is interactive. You can choose a performance level and the example will report: 1) The number of times that performance has occurred, 2) the number of times that performance or worse has occurred, and 3) the number of times that performance or better has occurred.

	A	B	C	D	E	F	G	H
1			Oct-09	Nov-09	Dec-09	Jan-10	Feb-10	Mar-10
2	Dept. Performance:		A	B	C	B	D	C
3								
4	Choose performance level:			B				
5								
6	Number of times chosen			2	<- =COUNTIF(C2:H2,D4)			
7	performance has occurred:							
8								
9	Number of times chosen			5	<- =COUNTIF(C2:H2,">="&D4)			
10	performance or worse occurred:							
11								
12	Number of times chosen			3	<- =COUNTIF(C2:H2,"<="&D4)			
13	performance or better occurred:							

Exercise

To apply the COUNTIF function to solving a problem click here.

Back to contents

Back to top

IF function

The IF function takes three arguments. The first describes a test or condition (which should return TRUE or FALSE). If the first argument is true the IF statement returns its second argument and otherwise it returns the third argument.

[For more information about logical tests and conditions click here.]

In the example below the IF statement's first argument is A1 > C1. A1 (5) is not greater than C1 (6) and so the test returns FALSE.

	A	B	C	D	E	F	G	H
1	5		6					
2								
3	less or equal		<- =IF(A1>C1,"greater","less or equal")					
4								

The following example is an interactive version of the one above.

	A	B	C	D	E	F	G	H
1	7		7					
2								
3								
4								
5	less or equal		<- =IF(A1>C1,"greater","less or equal")					
6								

Nested or compound IF functions

As we have seen the IF function returns one of two results (either the second or third argument). What if we want to return one of **three** results? One way of achieving this is to "nest" IF functions - i.e. have one "inside" the other.

We'll look at an example in which one of three results ('greater", "less", and "equal") is to be returned.

We will compare two values (Value 1 and Value 2) and determine whether Value 1 is greater than, equal to or less than Value 2.

Look at the formula in C3 below. The IF function in that formula makes the test A3>B3. A3 is 7 and B3 is 6. So the test is TRUE. Since the test is true the second argument of the IF function ("greater") is returned.

	A	B	C	D	E	F	G	H
1								
2	Value 1	Value 2	Test					
3	7	6	greater	<- =IF(A3>B3,"greater",IF(A3=B3,"equal","less"))				
4								
5	6	6	equal	<- =IF(A5>B5,"greater",IF(A5=B5,"equal","less"))				
6								
7	5	6	less	<- =IF(A7>B7,"greater",IF(A7=B7,"equal","less"))				

Now look at the formula in C5 above. This time the IF function's test isn't satisfied since A5 isn't greater than B5. So the third argument of the IF function is looked at. The third argument is a whole other IF function =IF(A5=B5,"equal","less"). This function's test is satisfied (A5 does equal B5) and so it returns its second argument ("equal").

In the formula in C7 both of the IF functions tests return FALSE and so the third argument of the last IF function is returned.

The example below is an interactive version of the one above.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Value 1	Value 2		Test								
2	7	6		greater	<- =IF(A2>B2,"greater",IF(A2=B2,"equal","less"))							
3												
4												
5												
6	Value 1 is greater than Value 2			<- ="Value 1 is " & IF(A2>B2,"greater than",IF(A2=B2,"equal to","less than")) & " Value 2"								

[To carry out an exercise on compound IFs click here.]

ISNA function

The ISNA function is often used to detect whether lookups have "failed". Failed lookups are those in which the item being looked for isn't found. We'll illustrate how ISNA works by considering an example in which we perform a lookup on a table containing departmental information.

The departmental table is defined in cells J2:K6 below. The first column of the table is a list of department codes and the second column gives the number of staff for each department. We want to do lookups of that table to find the number of staff for a given department.

In cell B2 we do a lookup of the table to find the number of staff in Department A. The VLOOKUP function in B2 works this way:

The first argument (A2) specifies what we're looking for. In this case it's "A". The second argument specifies where we're looking for "A". In this case it's in the departmental table at J2:K6. The third argument specifies which column of the table we want to retrieve information from. We want the "staff" column. Column numbering starts at the leftmost column of the table being searched and so the staff column is column number 2. The last argument specifies what to do if the item being looked for isn't in the table. If the fourth parameter is FALSE thenan error (#N/A) is returned if the item isn't found. If the fourth parameter is TRUE or is absent then the "nearest" item will be found.

[For more information on the VLOOKUP function click here]

	A	B	C	D	E	F	G	H	I	J	K
1	Dept	Staff					Lookup status			Dept	Staff
2	A	350	<- =VLOOKUP(A2,J2:K6,2,FALSE)				FALSE	<- =ISNA(B2)		A	350
3										D	120
4	C	#N/A	<- =VLOOKUP(A4,J2:K6,2,FALSE)				TRUE	<- =ISNA(B4)		F	200
5										G	12
6	Lookup failed C doesn't exist		<- =IF(ISNA(B4),"Lookup failed " &A4&" doesn't exist",B4)							H	6
7											

The lookup in B2 succeeds and returns the number of staff for Department A. Now consider the lookup in cell B4. That lookup fails because C - the department code being looked for - isn't in the department table.

Let's look now at how ISNA can detect the status of our lookups. ISNA takes a single argument. ISNA returns TRUE if its argument is #NA. ISNA returns FALSE if its argument isn't #NA.

The ISNA function in G2 detects the status of the VLOOKUP in B2. That lookup succeeds, the value it generates isn't #NA, and so ISNA returns FALSE. The ISNA function in G4 detects the status of the VLOOKUP in B4. That lookup fails and generates the result #NA. In this case ISNA returns TRUE.

An example of how the ISNA function can be used is shown in cell A6 above: IF and ISNA are used together to inform the user if the lookup fails. Note that the error message we generate is much more user friendly and informative than is #NA.

Interactive example

The example below is interactive. You can select a department. Depending on whether or not the selected department is found and appropriate result is returned.

	A	B	C	D	E	F	G	H	I	J	K
1							Dept	Staff	Location		
2							A	350	Singapore		
3							D	120	Auckland		
4							F	200	Sydney		
5	Dept:	A					G	12	New York		
6							H	6	London		
7											
8	Staff:	350	<- =IF(ISNA(VLOOKUP(B5,G2:I6,2,FALSE)),"no match", VLOOKUP(B5,G2:I6,2,FALSE))								

VLOOKUP is an "expensive" function in terms of the load it imposes on computers so it's best to minimise their use. The example above uses VLOOKUP twice in the formula in B8. If many such calculations are to be done its probably better to split the formula into two seperate cells and to use VLOOKUP only once.

LARGE function

The LARGE function returns the largest number from a group. More generally, the function returns the "Nth" largest number. The function takes two parameters. The first parameter is a reference to a cell or group of cells. The second parameter is a number that indicates "how large". If the second parameter is 1 then the largest number in the range is returned, if 2 then the second largest is returned, and so on.

	A	B	C	D	E	F	G	H
1								
2	17	3	27					
3	-34	19	4					
4								
5	27	<- =LARGE(A2:C3,1)						
6								
7	17	<- =LARGE(A2:C3,3)						

The following shows interactively how LARGE works. In cell C1 you choose "how large" (e.g. 1 = largest, 2 = second largest, etc). The answer is shown in cell A7. In A7 note how the IF and CHOOSE functions are used to add the "st" in "1st", the "nd in 2nd", the "rd" in 3rd and the "th" thereafter.

	A	B	C	D	E	F	G	H	I	J	K
1	"Nth" largest to show:		2		17	3	27				
2					-34	19	4				
3											
4											
5											
6											
7	The 2nd largest is 19	<- ="The " &C1&IF(C1>3,"th",CHOOSE(C1,"st","nd","rd"))&" largest is " & LARGE(E1:G2,C1)									
8											

Sorting

The LARGE function can be used to dynamically sort data. The example below gives a simple illustration.

Column A contains unsorted data. Column B contains the integers starting at 1 and increasing by 1 on each successive row.

	A	B	C	D	E	F	G	H
1	Unsorted	Rank	Sorted					
2	23	1	92	<- =LARGE($A$2:$A$7,B2)				
3	17	2	78	<- =LARGE($A$2:$A$7,B3)				
4	78	3	61	<- =LARGE($A$2:$A$7,B4)				
5	12	4	23	<- =LARGE($A$2:$A$7,B5)				
6	92	5	17	<- =LARGE($A$2:$A$7,B6)				
7	61	6	12	<- =LARGE($A$2:$A$7,B7)				

Column C contains sorted data and each cell's value is set by a LARGE function. The LARGE function on the top row has a second argument of 1. So C1 is the largest of the numbers in the A column. The LARGE function on the second row has a second argument of 2. So C2 is the second largest of the A numbers. And so on. We can see that the LARGE functions generate a sorted set of data in column C.

Paired sorting

Following is a more complex sorting example. Column A is unsorted. Column B is 'paired' with column A. So the 12 in cell B2 is paired with the 13 in cell A2. In the sorted columns D & E the 12 should still be paired with the 13.

An additional complication is that numbers in the unsorted column can be duplicated. The duplicates may have different "pairs".

	A	B	C	D	E	F	G	H
1	Unsorted	Paired		Sorted	Paired			
2	13	12		67	56			
3	25	17		45	19			
4	36	32		36	32			
5	45	19		25	17			
6	1	10		22	16			
7	67	56		13	12			
8	13	5		13	5			
9	22	16		1	10			

LEFT, MID, RIGHT and LEN functions

The LEFT function

The LEFT function works with "strings" - text in other words. The LEFT function can take one or two arguments. The first argument is a string. If there isn't a second argument then the LEFT function returns the leftmost character in its first argument. This is

Look at the formula in cell B1. The LEFT function's single argument is A1, A1 contains "FGH234". Because the LEFT function isn't provided with a second argument it returns

In the formula in cell B3 the LEFT function is provided with a second argument - 2. So the left 2 characters of FGH234 are returned.

If the second argument is larger than the number of characters in the first argument then the LEFT function returns all of the first argument - as shown by the formula in cell B5.

	A	B	C	D	E	F	G
1	FGH234	F	<- =LEFT(A1)				
2							
3		FG	<- =LEFT(A1,2)				
4							
5		FGH234	<- =LEFT(A1,20)				
6							
7							

Below is a simpled example showing the left function being used to abbreviate the names of the months to a standard three character length.

	A	B	C	D	E
1	January		Jan	<- =LEFT(A1,3)	
2	February		Feb	<- =LEFT(A2,3)	
3	March		Mar	<- =LEFT(A3,3)	
4	April		Apr	<- =LEFT(A4,3)	
5	May		May	<- =LEFT(A5,3)	
6	June		Jun	<- =LEFT(A6,3)	
7	July		Jul	<- =LEFT(A7,3)	
8	August		Aug	<- =LEFT(A8,3)	
9	September		Sep	<- =LEFT(A9,3)	
10	October		Oct	<- =LEFT(A10,3)	
11	November		Nov	<- =LEFT(A11,3)	
12	December		Dec	<- =LEFT(A12,3)	

[For a more complex example using the LEFT function click here.]

The RIGHT function

The RIGHT function works much the same as the LEFT function except that it takes the rightmost characters of a string. The following interactive example lets you choose the number of characters returned by the RIGHT function.

	A	B	C	D	E	F
1	Number characters returned from the string:				2	
2						
3	Text:	This is a long piece of text				
4						
5	Result:	xt			<- =RIGHT(B3,E1)	
6						

The MID function

The MID function returns the middle characters in a string. The function takes three arguments. The first argument is the string whose middle is being found. The second argument specifies the position of the first character that will be returned (character numbering starts at 1). The third parameter is the length of the returned string.

	A	B	C	D	E	F	G	H
1	Start position:		2					
2								
3	Number characters:		3					
4								
5								
6	String:	ABCDEF						
7								
8	Result:	BCD	<- =MID(B6,C1,C3)					
9								

The LEN function

The LEN function returns the number of characters in a string. The illustration below gives examples of the results returned by the LEN function.

	A	B	C	D	E	F	G	H
1	Z	1	<- =LEN(A1)					
2								
3	Zt	2	<- =LEN(A3)					
4								
5	2AC	3	<- =LEN(A5)					
6								
7	S D	3	<- =LEN(A7)		[Counts the middle space]			
8								
9	T	5	<- =LEN(A9)		[Counts the leading spaces]			
10								

LOOKUP function

Overview

The LOOKUP function performs a lookup on a table that has either a vertical organisation (i.e. in columns) or a horizontal organisation (i.e. in rows). Another function in the looking-up family - VLOOKUP - works only with tables in a vertical organisation. And HLOOKUP works only with tables that have a horizontal organisation.

There are two forms of the LOOKUP function - a Vector form - and an array form. In this section we'll look only at the Vector form.

Below is an example showing the vector form of the LOOKUP function. This form takes three arguments.

- The first argument specifies what is being looked for. In this case we're looking for a tax code of B (specified in cell A3).
- The second argument specifies which column (or row) is being searched. In this case we're searching column G.
- The third argument is the result column (or row) and contains the values the lookup will return.

If the lookup finds what it's looking for in the second position in the search column then it will return the second entry from the result column.

	A	B	C	D	E	F	G	H
1								
2	Tax code	Rate				Code		Rate
3	B	15%	<- =LOOKUP(A3,F3:F7,H3:H7)			A		5%
4						B		15%
5						C		35%
6						D		42%
7						E		47%
8								

Both the search column and the result column should be only one column wide. There is no requirement - unlike with VLOOKUP - to have the result column in any particular position relative to the search column. Consider the following example where they're "back to front" and offset.

	A	B	C	D	E	F	G	H
1						Rate		
2	Tax code	Rate				5%		
3	B	15%	<- =LOOKUP(A3,H4:H8,F2:F6)			15%		Code
4						35%		A
5						42%		B
6						47%		C
7								D
8								E

And in the following example the search and result both have a horizontal orientation.

	A	B	C	D	E	F	G	H
1	Rate	5%	15%	35%	42%	47%		
2								
3	Code	A	B	C	D	E		
4								
5	Tax code	Rate						
6	B	15%	<- =LOOKUP(A6,B3:F3,B1:F1)					
7								
8								

And in this example the search is horizontal and the result is vertical.

	A	B	C	D	E	F	G	H
1	Code	A	B	C	D	E		Rate
2								5%
3								15%
4								35%
5	Tax code	Rate						42%
6	B	15%	<- =LOOKUP(A6,B1:F1,H2:H6)					47%
7								
8								

The LOOKUP function isn't as flexible as the VLOOKUP function in terms of allowing you to specify whether or not you want an exact match. LOOKUP always returns a nearest match. As in the following example.

	A	B	C	D	E	F	G	H
1								
2	Tax code	Rate				Code		Rate
3	B	5%	<- =LOOKUP(A3,F3:F7,H3:H7)			A		5%
4						C		15%

5						D		35%
6						E		42%
7						F		47%
8								

B - the item being searched for - isn't in the search column. So the "nearest" result (the one for A) is returned.

For this to work reliably the search column must be in sorted ascending order.

See the description of the VLOOKUP function for an outline of how LOOKUP, VLOOKUP and HLOOKUP searching works to find the "nearest" result.

Note that LOOKUP will return an error if the first entry in the search column is larger than the item being looked up. As in the following example.

	A	B	C	D	E	F	G	H
1								
2	Tax code	Rate				Code		Rate
3	A	#N/A	<- =LOOKUP(A3,F3:F7,H3:H7)			B		5%
4						C		15%
5						D		35%
6						E		42%
7						F		47%
8								

The item being looked for is A. The first item in search column G is B. And the LOOKUP fails.

The following example lets you experiment with the LOOKUP function. The cell with a blue background can be changed. The "output" is the cell with the black background.

	A	B	C	D	E	F	G	H
1								
2	Tax code	Rate				Code		Rate
3	H	47%	<- =LOOKUP(A3,F3:F7,H3:H7)			B		5%
4						C		15%
5						E		35%
6						F		42%
7						G		47%
8								

Back to contents

Back to top

MATCH function

The MATCH function searches for a specified item in a list and returns the position of the item. For example, if the item is found in the first position of the list, the MATCH function will return 1.

The MATCH function takes three arguments. The first argument specifies what is being searched for, the second item specifies the row or column that is searched, and the last argument determines the kind of search done.

Consider the example shown below. Look at the MATCH function in cell A4. The first argument is "A". So the MATCH function will search for A. The second argument is $C$2:$G$2. This specifies the cells the MATCH function will search. So the function will search for "A" in the cells $C$2:$G$2. The third argument is 0. The third argument specifies whether MATCH returns an error if it doesn't find what it's looking for, or whether it returns the nearest it can find. A third argument of 0 - as in this case - requires the function to return an exact match or an error if it cannot find an exact match.

Third argument of 0 - exact match required

In the example below "A" is found in the first position of the search list so the MATCH function returns 1 - the position of the "A" in the list.

The MATCH function in cell A5 searchs for "B". "B" is fifth in the search list and so the function returns 5.

In cell A6 the MATCH function searches for "D". However, "D" isn't in the search list and, because the MATCH is required to find an exact match and one isn't found, an error is returned.

	A	B	C	D	E	F	G	H
1								
2			A	C	F	G	B	
3								
4	1	<- =MATCH("A",$C$2:$G$2,0)						
5	5	<- =MATCH("B",$C$2:$G$2,0)						
6	#N/A	<- =MATCH("D",$C$2:$G$2,0)						
7								

Third argument of 1 - nearest match in ascending list

If the third argument of the MATCH is a 1 then the function will return the nearest match it can find. For the function to work reliably in this case the search list must be sorted in **ascending** order. The example below shows how the MATCH function works in this case.

If there is an exact match the function will return the position of the MATCH. This case is shown in cell A4 below. The MATCH is searching for C and finds it in the second position.

The formula in cell A5 shows what happens if the item being searched for isn't found. "F" is being searched for but isn't in the list. The MATCH returns 3 which is the position of "E" in the list. So the MATCH function (when the third argument is 1) returns the position of the largest item that is less than or equal to what is being searched for.

The formula in cell A6 shows what happens when the item being searched for is less than - i.e. before - the first item in the search list: An error is returned.

	A	B	C	D	E	F	G	H
1								
2			B	C	E	G	I	
3								
4	2	<- =MATCH("C",$C$2:$G$2,1)						
5	3	<- =MATCH("F",$C$2:$G$2,1)						
6	#N/A	<- =MATCH("A",$C$2:$G$2,1)						
7								

Third argument of -1 - nearest match in descending list

If the third argument is -1 then the MATCH function will return the position of the nearest match. For this to work reliably the search list must be sorted in **descending** order.

The example below illustrates how the MATCH function works in this case. The function is searching for "D" but "D" isn't in the search list. The MATCH returns 2 - which is the position of "F". In this case the MATCH function returns the position of the smallest item that is greater than or equal to the

	A	B	C	D	E	F	G	H
1								
2			H	F	C	B	A	
3								
4	2	<- =MATCH("D",$C$2:$G$2,-1)						
5								

Sample application of the MATCH function

The following illustration shows an application of the MATCH function. This is the background to the application. We have a set of growth figures. In some years growth has been negative (as marked by -1's in column B below) and in other years it was positive (as marked by 1's). We want to know the first year in which growth had been positive "N" years in a row. "N" is set by the user (i.e. you) in cell E16.

	A	B	C	D	E	F
1	**---- Growth data ----**					
2	Year	Growth	# years			
3	1998	1	1.000	<- =IF(B3>0,SUM(C2,1),0)		
4	1999	-1	0.000			
5	2000	-1	0.000			
6	2001	1	1.000			
7	2002	1	2.000			
8	2003	-1	0.000			
9	2004	-1	0.000			
10	2005	1	1.000			
11	2006	1	2.000			
12	2007	1	3.000			
13	2008	-1	0.000			
14	2009	1	1.000			
15						
16	**Number of years of consecutive growth**				3	
17						
18	**First year in which growth was positive for 3 years**					
19	2007	<- =A3-1 +MATCH(E16,C3:C14,0)				
20						

We begin by working out a way to determine when there have been "N" consecutive years of positive growth. A good way to do that is as shown by the formula in cell C3.

The number in column C increases by 1 when growth has been positive and resets to zero when growth is negative. When the number in the C column reaches "N" - that's when there have been "N" consecutive years of positive growth.

We can then use the MATCH function to find the first position in column C of "N". That is done in cell A19 in this part of the formula: **MATCH(E16,C3:C14,0)**

We need to add the position returned by the MATCH to the first year (less 1) to obtain the actual year in which the growth was positive "N" times.

Back to contents

Back to top

MAX & MIN functions

The MAX function returns the maximum of its argument(s). The arguments can be references to single cells, groups of cells or can be numbers or formulae. The MAX function ignores non-numeric arguments.

Feedback from our Financial Modelling course: "The DVD will be very helpful!"

Look at the examples in the spreadsheet below.

Consider the MAX function in cell E2. Its first argument is the range A2:B2 and its second argument is A4:C4. The function returns the largest of the numbers in A2:B2 and A4:C4. In this case the largest number is 17.

	A	B	C	D	E	F	G	H
1								
2	limit	-2			17	<- =MAX(A2:B2,A4:C4)		
3								
4	5	17	12		-2	<- =MIN(A2:B2,A4:C4)		
5								
6								

The MIN function is just like the MAX except that it returns the minimum rather than the maximum. An example is shown in cell E4 in the illustration above. The MIN function in that case finds the smallest (i.e. least positive or most negative) of the numbers in A2:B2 and A4:C4. That number is -2.

Where might we use the MAX or MIN functions? We'll look at a simple example relating to budgeting. In the example below cells C2:F2 contain actual expense figures for four departments. The corresponding budgeted figures are in cells C4:F4.

On row 6 we work out the amount by which each department is over budget. The calculation is simply the actual expense less the budgeted figure. Some of the resulting figures are positive (indicating the actual expense was above the budgeted expense) and some are negative (indicating the actual expense was below the budgeted expense).

	A	B	C	D	E	F	G	H
1	Deptartment		North	East	South	West		
2	Actual expenses		36	37	43	18		
3								
4	Budgeted expenses		35	33	46	19		
5								
6	Amount over budget		1	4	-3	-1	<- =F2-F4	
7								
8	Maximum amount over budget			4	<- =MAX(C6:F6)			

Feedback from our Financial Modelling course: "Lots of practical examples used in the workshop"

In cell D8 we report on the maximum amount by which a department was over budget. In this case the answer is 4. Whilst that figure is correct and resonable, for other sets of departmental figures, the figure may not be appropriate. Consider the example following. It is the same as the one above except that the actual expenses differ.

In the example below the actual expenses are all below budget so all of the "amount over budget" figures are negative. This causes our "Maximum amount over budget" figure in D8 to become negative. In this case in D8 we'd probably prefer to see the figure 0 rather than -1.

	A	B	C	D	E	F	G	H
1	Deptartment		North	East	South	West		
2	Actual expenses		34	32	43	18		
3								
4	Budgeted expenses		35	33	46	19		
5								
6	Amount over budget		-1	-1	-3	-1	<- =F2-F4	
7								
8	Maximum amount over budget			-1	<- =MAX(C6:F6)			

We can modify the formula in D8 so that it shows 0 if all departments are within budget. The illustration below shows how the formula can be modified.

	A	B	C	D	E	F	G	H
1	Deptartment		North	East	South	West		
2	Actual expenses		34	32	43	18		
3								
4	Budgeted expenses		35	33	46	19		
5								
6	Amount over budget		-1	-1	-3	-1	<- =F2-F4	
7								
8	Maximum amount over budget			0	<- =MAX(MAX(C6:F6),0)			

Our Visual Basic course shows how to increase the "wow-factor" of your developed applications

The "raw" over-budget figure is calculated by an "inner" MAX function - MAX(C6:F6). We pass that result into an "outer" MAX function whose second argument is 0. The effect of that MAX function is to return 0 if the first MAX function returns a negative number.

[For examples of the MAX and MIN functions working with other functions click here]

Back to contents

Back to top

OFFSET function

Feedback from our Financial Modelling course: "Understanding the complex Excel tips was very beneficial"

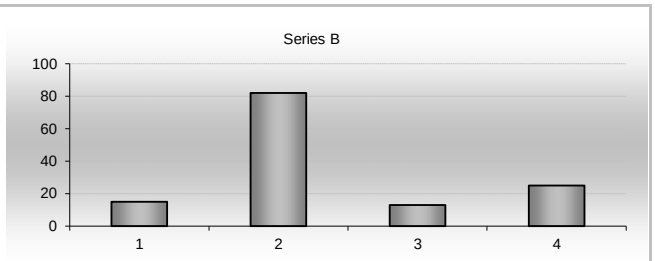
Overview

Usually when you write a formula you "hard-code" which cells the formula will reference. A formula might be =SUM($G5:H7). The $G5:H7 is a "hard-coded" cell reference. By "hard-coded" we mean that the cell reference is set explicitly at the time the formula is written.

But sometimes you can't hard-code which cells to reference. Consider the chart shown below for example. The chart shows one of six data series (A through F). Which of the six series is charted is determined by the user (i.e. you) making a selection in the blue area below.

Series to chart: B

Series name	Data			
A	36	4	64	47
B	15	82	13	25
C	22	44	13	93
D	16	5	76	88
E	72	9	17	27
F	31	28	47	23



In the chart we obviously can't "hard-code" a reference to a particular data series. That is because the location of the relevant series will change depending on the user's selection. Situations like this are where the OFFSET function is useful: The OFFSET function allows us to construct dynamic cell references based on changing criteria.

OFFSET fundamentals - OFFSET with three arguments

In its simplest version the OFFSET function takes three arguments. We'll illustrate by using the following example in which a user needs to be able to select one of four scenarios. The scenarios are defined in the cells shown with a light yellow background. Now look at the OFFSET formula in cell C6 below.

The OFFSET function's first argument is a "base" or "starting" position - A2 in this case.

The second argument specifies a row offset - it defines how many rows down to move from the base position. In this case the row offset is zero so we don't move any rows away from the base (i.e. we stay on row 2).

The third argument specifies a column offset - it defines how many columns to the right we move from the base position. In this case the column offset is determined by the number in C5. The number in C5 is 3. So we move three columns to the right from the base position.

	A	B	C	D	E	F	G	H
1	*Scenario*	*1*	*2*	*3*	*4*			
2	Growth	12%	14%	10%	11%			
3								
4	Chosen scenario							
5	Scenario number:		3					
6	Growth:		10%	<- =OFFSET(A2,0,C5)				
7								

We can customise our courses for particular audiences.

Having started at A2, moved down zero rows and moved across C5 columns we arrive at cell D2. The result of all this is that scenario 3's growth rate (in D2) is shown in the chosen scenario growth rate cell (C6). If the chosen scenario had been 4 then it would have been scenario 4's growth rate shown in C6. We can see that the OFFSET function lets the user select a scenario growth rate by typing the scenario's number into C5.

Positive and negative offsets

The second and third arguments of the OFFSET function can be negative or positive. A positive row offset defines a movement down the page and a negative offset is up the page. A positive column offset is a movement to the right and a negative offset is a movement to the left.

	A	B	C	D	E	F	G	H
1	3	4	5	12	19	28		
2	6	7	8	13	20	29		
3	9	10	11	14	21	30		
4	15	16	17	18	22	31		
5	23	24	25	26	27	32		
6								
7	20	<- =OFFSET(B4,-2,3)						
8								

In the example above we start at B4 and move two rows up and three columns to the right. The OFFSET function in this case generates the same as would the cell reference =E2.

OFFSET sample application - Scenario selection

The following example is a "live" version of the scenario example discussed earlier. You can select a scenario number (in the cell with the blue background). The selected scenario's parameters will then appear in the cells with the black background.

	A	B	C	D	E	F	G	H
1	Choose a scenario ->		2					
2								
3								
4	Scenario number		1	2	3			
5	Growth		12%	13%	14%			
6	Cost of capital		6%	7%	8%			
7								
8	Chosen scenario							
9	*Growth*		13%	<- =OFFSET($B5,0,$C$1)				
10	*Cost of capital*		7%	<- =OFFSET($B6,0,$C$1)				

OFFSET sample application - Setting an averaging period

In the following example we want to find the averages of a history of data points. The oldest data point is in cell B4 (its value is 100). The most recent data point is in cell B9 (its value is 80). We want to generate a series of averaged values in column D. We want the user to be able to select a two, three or four day averaging period by making a selection in cell C1.

	A	B	C	D	E	F	G	H	I
1	Averaging days		4						
2									
3	Day	Data		Averaged					
4	01/01/10	100							
5	02/01/10	10							
6	03/01/10	20							
7	04/01/10	40		42.5	<- =AVERAGE(B7:OFFSET(B7,-($C$1-1),0))				
8	05/01/10	60		32.5	<- =AVERAGE(B8:OFFSET(B8,-($C$1-1),0))				
9	06/01/10	80		50.0	<- =AVERAGE(B9:OFFSET(B9,-($C$1-1),0))				

How does the preceding example work? Suppose we're averaging over two days and are calculating the average value for row 7. To do that we need to find the average of the data points on rows 6 and 7. If we wanted averaging over three days we'd need to average the data on rows 5, 6 and 7. In other words, to find the average "today" over N days we need to average from "today" and back N-1 days.

So on row 7 our averaging formula begins like this

=AVERAGE(B7:

The average for today uses todays data point. The second part of the argument (i.e. the part after the ":") needs to be dynamically calculated by using the OFFSET function.

The second part is the following

OFFSET(B7,-($C$1-1,0))

We go back from today (i.e. B7) a certain number of rows. The number of rows we go back is the user's selection in $C$1 less 1.

Offset with five arguments

We have seen above that OFFSET can be used with three arguments. It can also be used with five (and four) arguments. In the case of five arguments the extra two arguments specify the size (in rows and columns) of the range returned. For example if the fourth and fifth arguments are 7 and 8 then the

range returned with have seven rows and eight columns.

In the example below we can see that OFFSET(A3,1,2,3,4) refers to the range C4:F6. How do we arrive at C4:F6? We start at A3 and go down 1 row (the second argument) and then move across 2 columns (the third argument). We're now at C4. We then return a range starting at C4 that is 3 rows down and 4 columns across: C4:F6.

	A	B	C	D	E	F	G	H
1	=OFFSET(A3,1,2,3,4)							
2								
3			---------- 4 columns ----------					
4								
5							3 rows	
6								
7								

Another illlustration of the OFFSET with five arguments is below. The formula in A5 returns a range one row by one columns starting at C2. So it returns the cell in C2 and passes that reference to the SUM function. So the SUM function returns the value of C2 (1).

The formula in A6 passes a one row, two column reference to the SUM function. That reference is equivalent to C2:D2. C2 contains 1 and D2 contains 2. So the SUM function returns 1 + 2 = 3.

The formula in A7 returns a reference equivalent to C2:C3. C2 contains 1 and C3 contains 8. So the SUM function returns 9.

The formula in A8 returns a reference equivalent to C2:D3. The SUM function returns the sum of the numbers in C2:D3.

	A	B	C	D	E	F	G	H
1								
2			1	2				
3			8	16				
4								
5	1	<- =SUM(OFFSET($C$2,0,0,1,1))						
6	3	<- =SUM(OFFSET($C$2,0,0,1,2))						
7	9	<- =SUM(OFFSET($C$2,0,0,2,1))						
8	27	<- =SUM(OFFSET($C$2,0,0,2,2))						
9								

The example below is a practical application of the five-argument version of the OFFSET function. It solves a problem discussed earlier on this page - but in a different way.

We want to find the average of "N" numbers. "N" is chosen by the user (i.e. you) and can be 2, 3 or 4. Look at the formula in F7 below. We can see that the fourth argument of the OFFSET function is linked to cell C1 - the user's choice of "N". So the OFFSET function passes a different "sized" column of cells to the AVERAGE depending on the user's choice.

	A	B	C	D	E	F	G	H	I
1	Averaging days		2						
2									
3	Day	Data		Averaged					
4	01/01/10	100							
5	02/01/10	10							
6	03/01/10	20							
7	04/01/10	40		40.0	<- =AVERAGE(OFFSET(B7,0,0,-($C$1-1),1))				
8	05/01/10	60		60.0	<- =AVERAGE(OFFSET(B8,0,0,-($C$1-1),1))				
9	06/01/10	80		80.0	<- =AVERAGE(OFFSET(B9,0,0,-($C$1-1),1))				

Back to contents

Back to top

SUM function

The SUM function takes up to thirty arguments. Each argument describes a number, cell or block of cells to add up. In the following example the SUM function adds the numbers in two cells (B2 and B4 and the number 4).

	A	B	C	D	E	F	G
1							
2	Num 1	5					
3							
4	Num 2	3					
5							
6	Sum	12	<- =SUM(B2,B4, 4)				
7							

In the example below the SUM function adds the numbers in two blocks of cells (B2:D2 and B4:C4) together with the number in the single cell (B6).

	A	B	C	D	E	F	G
1							
2	Group 1	1	2	4			
3							
4	Group 2	8	16				
5							
6	Num 3	32					
7							
8	Sum	63	<- =SUM(B2:D2,B4:C4,B6)				

Differences between SUM and explicit addition

SUM interprets cells that contain text as being zero. So, for example, SUM(6, "column title") would be 6. An example of this behaviour is shown below.

In column B we are accumulating a running total. In cell B3 we add the current value (in cell A3) to the previous running total in the row above. When we do this in the first row we pick up the column title rather than a real running total. However, since SUM interprets text as being zero, this works as it should.

Note that an alternative method of accumulating the running total (by explicit addition - as shown in column F) generates errors when text is added to numbers.

	A	B	C	D	E	F	G	H
1								
2	Value	Running total			Value	Running total		
3	1	1	<- =SUM(A3,B2)		1	#VALUE!	<- =E3+F2	
4	2	3			2	#VALUE!		
5	4	7			4	#VALUE!		
6								

Another difference between SUM and explicit addition is shown below. Both the SUM formula (in C6) and the explicit addition formula (in C8) are adding cells B2 and B4. But the two answers are different: SUM returns 3 and explicit addition returns 8. The reason is that SUM interprets cell B2 as zero.

The reason it does that is that B2 actually contains text rather than a number: A leading apostrophe (') marks the cell as text. And, as we have seen earlier, SUM interprets text as zero.

	A	B	C	D	E	F	G	H
1								
2	Num 1	5	<- 5					
3								
4	Num 2	3						
5								
6	SUM:		3	<- SUM(B2,B4)				
7								
8	Explicit addition:		8	<- =B2+B4				

The reason explicit addition returns the 8 in C8 is that explicit addition "promotes" text to numbers if the text is interpretable as being a number.

SUMPRODUCT function

The SUMPRODUCT function is one of the most powerful yet underutilised and underappreciated of the spreadsheet functions. In this section we'll look at a range of applications of this function - ranging from quite simple to the more complex.

The most basic use of the SUMPRODUCT function is to multiply two rows (or columns) of numbers together and to add the resulting subtotals to give a grand total.

In the example below we hold a "portfolio" of three types of units. Each unit has a quantity and unit price. The quantities are in column B and the prices are in column C. The SUMPRODUCT function in cell E1 calculates the value of our portfolio.

In the example below the SUMPRODUCT function multiplies each of the quantities in cells B2:B4 by the corresponding unit prices in cells C2:C4 and then adds the individual results to form a "grand total".

	A	B	C	D	E	F	G	H	I
1	Unit code	Quantity	Unit price		Portfolio value				
2	BHP	12	20		450	<- =SUMPRODUCT(B2:B4,C2:C4)			
3	RIO	15	10						
4	ANZ	30	2						

The example above shows the most common usage of the SUMPRODUCT function. But there is another usage that allows us to perform complex logical and arithmetic calculations. We'll introduce this alternative usage in a phased way. We'll look first at how TRUE and FALSE are represented in spreadsheets, then we'll look at how SUMPRODUCT can process arrays of TRUEs and FALSEs, and last we'll look at doing complex conditional calculations with SUMPRODUCT.

SUMPRODUCT working with 0s and 1s

We've seen above how the SUMPRODUCT function works with two columns of numbers. In the example below we are doing the same as above. The only difference is that all of the numbers are 0s or 1s.

	A	B	C	D	E	F	G	H
1								
2	1	0		1	<- =SUMPRODUCT(A2:A5,B2:B5)			
3	0	1						
4	1	1						
5	0	0						
6								

The SUMPRODUCT function above multiplies the 0's and 1's in one column by the 0's and 1's in the other column and adds the results. The number returned by the calculation is the number of times there are 1's in the same row. Now why is this behaviour important or useful? The usefulness relates to the fact that spreadsheets interpret TRUE and FALSE as being the same as 1 and 0.

TRUE=1 and FALSE=0

We can show that spreadsheets see TRUE as being the same as 1 and FALSE the same as 0 by looking at the following example.

	A	B	C	D	E	F	G	H
1								
2	TRUE	2	<- =A2+1					
3								
4	FALSE	1	<- =A4+1					
5								

Cell A2 contains the value TRUE. Cell B2 contains the formula =A2+1. Since cell B2 returns 2 we can conclude that the spreadsheet interprets TRUE as being the same as 1.

Cell A4 contains the value FALSE. Cell B4 contains the formula =A4+1. Since cell B4 returns 1 we can conclude that the spreadsheet interprets FALSE as being gthe same as 0.

SUMPRODUCT working with TRUEs and FALSEs

The example below is very similar to the example shown in the earlier section (SUMPRODUCT working with 0s and 1s). There are two small differences:

The first difference is that 1s have been replaced by TRUEs and 0s by FALSEs. The second difference is that the comma between the two SUMPRODUCT arguments has been replaced by a multiplication sign (*).

	A	B	C	D	E	F	G	H
1								
2	TRUE	FALSE		1	<- =SUMPRODUCT(A2:A5*B2:B5)			
3	FALSE	TRUE						
4	TRUE	TRUE						
5	FALSE	FALSE						
6								

We can see that the SUMPRODUCT calculates the number of rows on which both columns are TRUE.

Budgeting

We'll now apply SUMPRODUCT to a budgeting exercise. We'll first solve the problem using a "long-way" and then a "short-way". First, the long way:

The problem is to determine the number of departments that were within budget in both 2008 and 2009. The approach we'll use below is to generate two columns of TRUEs and FALSEs corresponding to whether the department was within budget in 2008 and 2009.

	A	B	C	D	E	F	G	H
1		Budget			Actual			
2	Dept	2008	2009		2008	2009		
3	A	34	40		33	41		
4	B	23	28		25	27		
5	C	19	22		16	20		
6	D	13	17		14	19		
7								
8	Within budget	2008	2009					
9	A	TRUE	FALSE	<- =C3>F3				
10	B	FALSE	TRUE	<- =C4>F4				
11	C	TRUE	TRUE	<- =C5>F5				
12	D	FALSE	FALSE	<- =C6>F6				
13								
14	**Number departments within budget in both years**							
15	1	<- =SUMPRODUCT(B9:B12*C9:C12)						
16								

Look at the calculation in cell C9. In that cell we are calculating whether Dept A's budget in 2009 exceeded the actual figure in 2009. The result of that particular calculation is FALSE. There are similar calculations in the other cells in B9:C12.

Having generated two columns of TRUEs and FALSEs we can use SUMPRODUCT to determine how many rows contain two TRUEs (i.e. the number of departments that were within budget in both years). That calculation is in cell A15.

That completes our "long-way" of calculating the answer. Now we'll do the "short-way". The short way involves calculating the TRUEs and FALSEs directly in the SUMPRODUCT function rather than in separate cells in the spreadsheet. The illustration below shows how this is done.

	A	B	C	D	E	F	G	H
1		Budget			Actual			
2	Dept	2008	2009		2008	2009		
3	A	34	40		33	41		
4	B	23	28		25	27		
5	C	19	22		16	20		
6	D	13	17		14	19		
7								
8	**Number departments within budget in both years**							
9	1	<- =SUMPRODUCT((B3:B6>E3:E6)*(C3:C6>F3:F6))						
10								

Look at the SUMPRODUCT formula in cell A1. This part of the formula generates an "internal" array of four TRUEs and FALSEs: **(B3:B6>E3:E6)**

This part of the formula generates another array of TRUEs and FALSEs: **(C3:C6>F3:F6)**

So the SUMPRODUCT, in effect, is doing this calculation: **=SUMPRODUCT(({TRUE;FALSE;TRUE;FALSE})*({FALSE;TRUE;TRUE;FALSE}))**

Grouping by date

Below is an example showing how SUMPRODUCT can be used to group transactions by date. In columns A and B (in cells A7:B16) is a list of date intervals. We want to know the value of transactions that occurred in each of the date intervals. The raw transactions are listed in rows 2 and 3.

Look at the SUMPRODUCT formula in cell C7. There are three components to that formula.

The first component is **($B$2:$L$2>=A7)**. This generates an array of TRUEs and FALSEs (in effect 1s and 0s). This component tests whether transaction dates in B2:L2 occurred after the "From" date in A7.

The second component is **($B$2:$L$2<=B7)**. This generates another array of 1's and 0's. This component tests whether transactions dates occurred before the "to" date in cell B7.

The third component is **($B$3:$L$3)**. This is a list of transaction amounts.

	A	B	C	D	E	F	G	H	I	J	K	L
1	**Raw transactions**											
2	Date	5-Jun-09	7-Jun-09	4-Jun-09	7-Jul-09	9-Sep-09	7-Oct-09	7-Oct-09	5-Jan-10	4-Feb-10	5-Feb-10	5-Mar-10
3	Amount	34	15	16	71	23	42	82	76	99	12	3
4												
5	**Grouped transactions**											
6	From	to	Value									
7	1-Jun-09	30-Jun-09	65	<- =SUMPRODUCT(($B$2:$L$2>=A7)*($B$2:$L$2<=B7)*($B$3:$L$3))								
8	1-Jul-09	31-Jul-09	71									
9	1-Aug-09	31-Aug-09	0									
10	1-Sep-09	30-Sep-09	23									
11	1-Oct-09	31-Oct-09	124									
12	1-Nov-09	30-Nov-09	0									
13	1-Dec-09	31-Dec-09	0									
14	1-Jan-10	31-Jan-10	76									
15	1-Feb-10	28-Feb-10	111									
16	1-Mar-10	31-Mar-10	3									
17												

We can see that the SUMPRODUCT function is multiplying the transaction amounts by a series of 1's and 0's and adding the subtotals. Only those transactions are included in the total that have transaction dates after the from date and before the to date.

[For an example showing the SUMPRODUCT function used with another function in a more complex application click here]

Back to contents

- Page 36 -

Back to top

VLOOKUP function

Overview

The VLOOKUP function performs a lookup on a table that has a vertical orientation (i.e. one that is laid out in columns). [Another function in the lookup family - LOOKUP - works with tables that have vertical AND horizontal

In the following example the table being looked up is shown as the coloured area in columns G and H. Column G contains a list of tax codes and column H has a list of associated tax rates.

	A	B	C	D	E	F	G	H
1							Tax Table	
2	Tax code	Rate					Code	Rate
3	B	15%	<- =VLOOKUP(A3,G3:H7,2,FALSE)				A	5%
4							B	15%
5							C	35%
6							D	42%
7							E	47%
8								

VLOOKUP takes three or four arguments. The first argument specifies what is being looked for (B in this example - in cell A3).

The second argument describes the location of the table being searched (G3:H7 in this case).

The third argument controls which column of the lookup table the answer is obtained from. In this example the third argument is 2. So the answer is retrieved from the second column. Column numbering is relative to the seach column (G in this case). The search column is treated as being column number 1. So in this example column 2 means the H column. The tax rate for tax code B is retrieved and found to be 15%.

The fourth argument of the VLOOKUP is optional. In our example the argument's value is FALSE. FALSE tells VLOOKUP to find an exact match for the item being looked for. If the item being looked for isn't found VLOOKUP will return an error. An example of this is shown next.

	A	B	C	D	E	F	G	H
1							Tax Table	
2	Tax code	Rate					Code	Rate
3	B	#N/A	<- =VLOOKUP(A3,G3:H7,2,FALSE)				A	5%
4							C	15%
5							D	35%
6							E	42%
7							F	47%
8								

In the example above "B" isn't in the search column of the lookup table (i.e it's not in column G) and a #N/A error is returned.

We can specify that, if the lookup doesn't find exactly what we specified, it returns the "nearest" it can find rather than an error. To do that we set the fourth argument to TRUE (or leave it out altogether).

	A	B	C	D	E	F	G	H
1							Tax Table	
2	Tax code	Rate					Code	Rate
3	C	5%	<- =VLOOKUP(A3,G3:H7,2,TRUE)				A	5%
4							D	15%
5							E	35%
6							F	42%
7							G	47%
8								

In the example above the item being looked for - C - doesn't exist in the lookup table. VLOOKUP returns a rate of 5% - which is A's tax rate. A is the "nearest" the lookup finds to C. Why does the VLOOKUP choose A's tax rate rather than D's? After all "D" is closer to "C" than is "A". This is how VLOOKUP's searching works:

VLOOKUP assumes the search column is sorted in increasing order. VLOOKUP starts at the top of the search column and scans down through that column until it finds the first item that is greater than (or equal to) the item being searched for. In this example VLOOKUP will step down the search column till it hits "D". It then realises it has "gone too far" and "backs up" one row. It arrives on the row with A. And A's tax rate is what VLOOKUP returns.

	A	B	C	D	E	F	G	H
1							Tax Table	
2	Tax code	Rate					Code	Rate
3	C	5%	<- =VLOOKUP(A3,G3:H7,2,TRUE)				A	5%
4			<- 4				D	15%
5							E	35%
6							F	42%
7							G	47%
8								

A similar but more extreme example is the following: We want a nearest match for "V". Even though V is alphabetically closer to W than it is to C - it is C's tax rate that is returned. As in the preceding example VLOOKUP "steps" through the search column till it realises it has gone too far (at W) and backs up by one row and returns C's tax rate.

	A	B	C	D	E	F	G	H
1							Tax Table	
2	Tax code	Rate					Code	Rate
3	V	15%	<- =VLOOKUP(A3,G3:H7,2,TRUE)				A	5%
4							C	15%
5							W	35%
6							X	42%
7							Y	47%
8								

This searching behaviour also happens if the search column contains numbers. In the following example the tax rate table contains numbers (income levels) in the first, search, column.

	A	B	C	D	E	F	G	H
1							Tax Table	
2	Income	Rate					Income	Rate
3	54,000	15%	<- =VLOOKUP(A3,G3:H7,2,TRUE)				10,000	0%
4							30,000	15%
5							55,000	35%
6							95,000	42%
7							150,000	47%
8								

In the example above we're looking for the tax rate for an income level of $54,000. Note that the rate returned is 15%. This rate will be returned for all income levels between $30,000 and up to (but not including) $55,000.

If a "nearest match" lookup is being done and the item being searched for is larger than the last item in the search column then the last entry is returned. As in the following example.

	A	B	C	D	E	F	G	H
1							Tax Table	
2	Income	Rate					Income	Rate
3	154,000	47%	<- =VLOOKUP(A3,G3:H7,2,TRUE)				10,000	0%
4							30,000	15%
5							55,000	35%
6							95,000	42%
7							150,000	47%
8								

Here the income - $154,000 - is greater than the last income in the search column - $150,000. So the tax rate for $150,000 is returned.

In a VLOOKUP if we specify a "nearest" lookup (by omitting the fourth argument or having it TRUE) and an exact match is found, then the exact item is returned. As in the following example: An exact match for "V" is found.

	A	B	C	D	E	F	G	H
1							Tax Table	
2	Tax code	Rate					Code	Rate
3	V	35%	<- =VLOOKUP(A3,G3:H7,2,TRUE)				A	5%
4							C	15%
5							V	35%
6							X	42%
7							Y	47%
8								

For a "nearest" lookup to work the search column has to be in sorted ascending order. If it is not then the results will be unpredictable or wrong. Consider the following example. The search column isn't in sorted order ("C" appears before "A"). The lookup fails. The reason it fails is that the lookup expects the search column to be sorted in ascending order. The first item it sees is "C". It's looking for B and so "gives up" straight away and

returns an error.

	A	B	C	D	E	F	G	H
1							Tax Table	
2	Tax code	Rate					Code	Rate
3	B	#N/A	<- =VLOOKUP(A3,G3:H7,2,TRUE)				C	5%
4							A	15%
5							D	35%
6							E	42%
7							F	47%
8								

If the searching column is not in sorted order and we are searching for a nearest match - even if an exact match exists - then the lookup may still fail. That is illustrated in the following scenario.

	A	B	C	D	E	F	G	H
1							Tax Table	
2	Tax code	Rate					Code	Rate
3	B	#N/A	<- =VLOOKUP(A3,G3:H7,2,TRUE)				C	5%
4							B	15%
5							D	35%
6							E	42%
7							F	47%
8								

The item being search for - B - is in the table but isn't found because the search column isn't sorted in ascending order.

VLOOKUP assumes the search column is the leftmost column in its second argument. If you try to have the search column elsewhere the lookup will fail. As in the following example. In this example the search column is the rightmost one. The lookup has no way of knowing that you intend the right column to be the search one so it uses the leftmost one. And not surprisingly - it fails.

	A	B	C	D	E	F	G	H
1							Tax Table	
2	Tax code	Rate					Rate	Code
3	B	#N/A	<- =VLOOKUP(A3,G3:H7,2,TRUE)				5%	A
4							15%	B
5							35%	C
6							42%	D
7							47%	E
8								

Another way that a VLOOKUP can fail is if the reference to the table being looked up doesn't contain enough columns. Note that in the following example the lookup table is only a single column (G). We want to retrieve data from the adjacent (H - Rate) column. But since the second argument to the lookup doesn't contain that column - the VLOOKUP fails.

	A	B	C	D	E	F	G	H
1							Tax Table	
2	Tax code	Rate					Code	Rate
3	V	#REF!	<- =VLOOKUP(A3,G3:G7,2,TRUE)				A	5%
4							C	15%
5							V	35%
6							X	42%
7							Y	47%
8								

The following example lets you experiment with the VLOOKUP. Cells with a blue background can be changed. The "output" is the cell with the black backgound.

	A	B	C	D	E	F	G	H
1							Tax Table	
2	Tax code		F				Rate	Code
3							B	5%
4	Search type		FALSE				D	15%
5							E	35%
6							F	42%
7							G	47%
8	Tax rate		42%	<- =VLOOKUP(C2,G3:H7,2,C4)				

Back to contents

Back to top

Chapter 3 - Combining Functions

In Chapter 2 we saw some examples of completed spreadsheet applications. And at the end of Chapter 2 is a table that lists all of the spreadsheet functions used to develop the sample applications.

In Chapter 3 we described most of the functions that were used in Chapter 2. We listed the functions one by one and explained how they worked and gave illustrations of how they behaved.

How can we use the functions described in Chapter 3 to build applications like those in Chapter 2? Obviously - by combining the functions appropriately. In this chapter we will start combining functions to build "mini" applications.

Quartiles

Suppose you have a set of sales figures as shown below. We want to know the 1st and 3rd quartile sales figures. The 1st quartile sales figure is the one that is at the lowest end of the top 25% sales. The 3rd quartile sales figure is the highest one in the bottom 25% sales.

	A	B	C	D	E	F	G	H
1	**Sales figures**							
2	67	2	11	53	47	3	51	44
3	26	82	68	90	12	74	34	19
4	16	56	56	21	7	41	39	77
5	70	45	24	12	76	64	28	38
6	69	10	94	60	2	93	13	36
7	19	69	16	79	65	94	6	94
8								

To fnd the quartiles we can use the LARGE and COUNT functions as shown below.

	A	B	C	D	E	F	G	H
1	**Sales figures**							
2	67	2	11	53	47	3	51	44
3	26	82	68	90	12	74	34	19
4	16	56	56	21	7	41	39	77
5	70	45	24	12	76	64	28	38
6	69	10	94	60	2	93	13	36
7	19	69	16	79	65	94	6	94
8								
9	1st quartile sales:		69	<- =LARGE(A2:H7,COUNT(A2:H7)*0.25)				
10	3rd quartile sales:		19	<- =LARGE(A2:H7,COUNT(A2:H7)*0.75)				

The LARGE function tells us which is the "Nth" largest in a range. The LARGE function's first argument is the range being counted. The second argument is N. So if, for example, we want to find the 25th biggest number in the range A2:H7 we could find that number by using LARGE(A2:H7, 25). If there were 100 numbers in the range A2:H7 then the number we would then obtain would be the 1st quartile sales figure. But if there are only 25 numbers in the range then we'd need to scale down the number in the second argument to 12.

In the example above we calculate the second argument of the LARGE function by using the COUNT function. The COUNT function tells how many numbers there are in the given range. We then multiply the number returned by COUNT by 0.25 to find the 1st quartile sales. We multiply by 0.75 to find the 3rd quarter sales.

The next example in interactive. You can choose which quartile figures are to be calculated. We have used conditional formatting to highlight the figures in the chosen quartile.

	A	B	C	D	E	F	G	H	I
1	**Sales figures**								
2	67	2	11	53	47	3	51	44	
3	26	82	68	90	12	74	34	19	
4	16	56	56	21	7	41	39	77	
5	70	45	24	12	76	64	28	38	
6	69	10	94	60	2	93	13	36	
7	19	69	16	79	65	94	6	94	
8									
9	Quartile:	1		Sales:	69	<- =LARGE(A2:H7,COUNT(A2:H7)*B9/4)			
10									

Back to contents

Back to top

Complex counting

Here we look at an illustration of how SUMPRODUCT can be used in complex conditional counting. By conditional counting we mean counting only when a certain criterion is satisfied.

The example below contains a list of product codes in column A in cells A6:A13. We want to find the number of product codes starting with a particular letter. That letter is specified by the user (i.e. you) in cell E1.

Our workshops cover techniques to perform complex calculations in a single cell that would otherwise take many cells to do.

	A	B	C	D	E	F	G	H	I
1	Count number of codes starting with:				F				
2									
3	Number codes starting with F:			2	<- =SUMPRODUCT((LEFT(A6:A13)=$E$1)*1)				
4									
5	**Codes**								
6	F234								
7	G123								
8	E435								
9	A011								
10	F120								
11	G230								
12	C111								
13	B012								

In this example we use the LEFT function to get the leftmost character of each product code in A6:A13. We then use the SUMPRODUCT function to find the number of times the left character equals the character we want to count. We need to use a "trick" with SUMPRODUCT. We multiply by 1 because SUMPRODUCT needs to have at least two things to multiply together.

[For more information about the SUMPRODUCT function click here]

If you find the example above a bit difficult to follow - and it is fairly complex - then look at the example below. It solves the same problem in much the same way as above but uses extra spreadsheet cells for "intermediate" calculations. Note the "dummy" column of TRUEs in column G. These are needed because the SUMPRODUCT needs at least two arguments to multiply together.

Our financial modelling course covers the finance and accounting concepts that underlay financial modelling

	A	B	C	D	E	F	G	H
1	Count number of codes starting with				F			
2								
3	Number codes starting with F:			2	<- =SUMPRODUCT(B6:B13*F6:F13)			
4								
5	**Codes**							
6	F234	TRUE	<- =LEFT(A6,1)=$E$1			TRUE	<- TRUE	
7	G123	FALSE	<- =LEFT(A7,1)=$E$1			TRUE	<- TRUE	
8	E435	FALSE	<- =LEFT(A8,1)=$E$1			TRUE	<- TRUE	
9	A011	FALSE	<- =LEFT(A9,1)=$E$1			TRUE	<- TRUE	
10	F120	TRUE	<- =LEFT(A10,1)=$E$1			TRUE	<- TRUE	
11	G230	FALSE	<- =LEFT(A11,1)=$E$1			TRUE	<- TRUE	
12	C111	FALSE	<- =LEFT(A12,1)=$E$1			TRUE	<- TRUE	
13	B012	FALSE	<- =LEFT(A13,1)=$E$1			TRUE	<- TRUE	

2D lookups

Elsewhere in this section we describe how to use the VLOOKUP and LOOKUP functions to read information from a table: To do that we specify a "key" - an item to search for - and the lookup returns a corresponding value from the table. This use of lookups is "one-dimensional" in that we specify a single key.

But what if we have two keys? (e.g. a month and a division code, say) Then we'd need to perform a two-dimensional lookup. The following examples show how to do this.

In the illustration below we have a two dimensional table showing data by month (in rows) and years (in columns). We want to retrieve data from the table by month and year. We might, for example, want the data for July, 2008.

Method 1

In the example below we use VLOOKUP and MATCH functions to find the data. The first argument of the VLOOKUP function specifies what we're looking for. In this case the first argument is B1 which contains the month we're looking for.

The second argument of the VLOOKUP specifies where we're looking for the data. The second argument is I3:L14. The leftmost column of this range (i.e. the I column) will be searched by the VLOOKUP to find the month specified in its first argument (i.e. the contents of cell B1).

The third argument of the VLOOKUP specifies which column of I3:L14 to retrieve the data from. We can't "hard-code" this number because the column will depend on which year the user has selected in cell B3. So we "dynamically" calculate the column by using a MATCH function. Look at the MATCH function in cell B12. That function calculates which column we need to retrieve our data from.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Month	Sep								2007	2008	2009
2												
3	Year	2008							Jan	65	55	4
4									Feb	22	18	69
5									Mar	48	6	93
6									Apr	17	13	38
7									May	70	54	80
8									Jun	33	76	84
9	Sales	71	<- =VLOOKUP(B1,I3:L14,MATCH(B3,I1:L1,0),FALSE)						Jul	68	34	30
10									Aug	91	29	46
11									Sep	1	71	87
12	*What the MATCH function does:*								Oct	41	38	9
13									Nov	54	65	20
14		3	<- *=MATCH(B3,I1:L1,0)*						Dec	65	34	70

The MATCH function's first argument specifies what it is looking for. In this case it's B3 which contains the year the user selects. The second argument specifies which range the MATCH will search. In this case it's I1:L1. The last argument is 0 and that makes the MATCH function look for an exact match for the item being looked for. The MATCH function will return the **position** (not value) of the item being looked for.

Now back to the VLOOKUP function in B5. We've seen how the MATCH function calculates which column of I3:L14 to retrieve data from. Column numbering starts at 1 in the leftmost column of the VLOOKUPs second argument (i.e. Column 1 = Column I, Column 2 = Column J, Column 3 = Column K and Column 4 = Column L).

The last argument of the VLOOKUP is FALSE. That makes the VLOOKUP require an exact match for the item being looked for.

[For more information on the VLOOKUP function click here]
[For more information on the MATCH function click here]

Method 2

We can solve the lookup problem another way: We can use MATCH and OFFSET rather than MATCH and VLOOKUP. The following example shows how to do this.

Look at the formula in cell B5: We use MATCH functions to find the row and column to retrieve data from. Once we know the row and column offsets those numbers are passed to the OFFSET function to retrieve the appropriate data item.

The MATCH functions in cells C11 and C12 show how the row and column offsets are calculated and have been provided only for explanation purposes - Cells C11 and C12 aren't needed by the OFFSET function in B5 because that formula uses MATCH itself to find the offsets.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Month	Aug								2007	2008	2009
2												
3	Year	2008							Jan	65	55	4
4									Feb	22	18	69
5									Mar	48	6	93
6									Apr	17	13	38
7									May	70	54	80
8									Jun	33	76	84
9	Sales	29	<- =OFFSET(I2,MATCH(B1,I3:I14,0),MATCH(B3,J1:L1,0))						Jul	68	34	30
10									Aug	91	29	46
11	*What the MATCH functions do:*								Sep	1	71	87
12									Oct	41	38	9
13	*"Row" MATCH:*	8	<- *=MATCH(B1,I3:I14,0)*						Nov	54	65	20
14	*"Column" MATCH:*	2	<- *=MATCH(B3,J1:L1,0)*						Dec	65	34	70

[For more information on the OFFSET function click here]

Applications of the MAX and MIN functions

In this section we look at applications of the MAX and MIN functions to problems that require other functions also to be used.

[To revise basics about the MAX and MIN functions click here.]

Detecting duplicates

In the following example we want to test whether transaction codes in a list of transaction codes are duplicated. The transaction codes are in cells A3:A7 below.

COUNTIF functions in cells D3:D7 count the number of times each transaction code occurs. If a transaction code occurs more than once that means it has been duplicated.

The MAX function in cell C24 finds the maximum of the number of occurrences of each transaction code. If the maximum is more than one then that means one or more transaction codes are duplicated. The IF function in cell C11 reports the result of the duplication test. That cell shows "Yes" or "No" depending on whether duplication has been found or not.

	A	B	C	D	E	F	G
1	--- Transactions ---			Frequency of			
2	*Code*	*Amount*		transactions			
3	BTR12	27		1	<- =COUNTIF($A$3:$A$7,A3)		
4	BTR13	32		2	<- =COUNTIF($A$3:$A$7,A4)		
5	CAF09	9		1	<- =COUNTIF($A$3:$A$7,A5)		
6	BTR13	8		2	<- =COUNTIF($A$3:$A$7,A6)		
7	ABD16	4		1	<- =COUNTIF($A$3:$A$7,A7)		
8							
9	Max frequency		2	<- =MAX(D3:D7)			
10							
11	Duplicates:		Yes	<- =IF(C9>1,"Yes","No")			
12							

The example above is interactive. You can set the transaction code in cell A6 to be either a duplicate of another or not to duplicate another.

Payback period

In the example below we determine the payback period on a project. We track the project cash flows. The cash flows are initially negative and then become positive. We find the date at which the first cumulative cash flow becomes positive.

Row 4 below contains the project cash flows. They are initially negative (i.e. cash outgoings - investments) and later become positive (i.e. cash incomings - return on investment).

On row 7 we calculate the cumulative cash flows.

	A	B	C	D	E	F	G	H	I	J
1										
2	Year:		2005	2006	2007	2008	2009	2010		
3										
4	Cash flow:		-32	-6	15	20	23	45		
5										
6										
7	Cumulative cash flow:		-25	-31	-16	4	27	72	<- =G7+H4	
8										
9	Cumulative cash flow		NA	NA	NA	2008	2009	2010	<- =IF(H7>0,H2,"NA")	
10	is +ve:									
11										
12	Payback year:		2008	<- =MIN(C9:H9)						
13										

On row 9 we check whether the cumulative cash flow is greater than zero. If it is then we "remember" the year in which that occurred. If the cash flow is not greater than zero we indicate that with an "NA".

Last, we use a MIN function in cell C12 to find the smallest (i.e. first) year in row 9. Note that MIN "ignores" the NAs rather than interpreting them as zero - for if MIN had treated NA as zero the payback year would have been calculated as zero.

The example above is interactive. You can change the cash flow in cell D4 to alter the payback year.

Back to contents

Back to top

Chapter 4 - Exercises

This chapter contains a number of exercises. To carry out the exercises you will need to apply your design skills. The following are important skills to successfully build spreadsheet solutions:

- Knowledge of the features and functions available in spreadsheets
- An ability to think logically and to be able to break larger problems down into smaller ones
- An ability to understand and interpret written material
- Mathematical ability

You will be able to exercise and develop your skills in all of the above areas if you carry out the tasks given in this chapter.

All of the exercises have a common format: A problem is described, a small spreadsheet is given and you need to complete the spreadsheet to solve the problem. Your formulae should be put into the cell(s) which have a bright yellow backgound. Some of the exercise spreadsheets give you an additional "workspace". At the bottom of each spreadsheet is a section titled "Status of your answer". That section will inform you of your progress as you work you way through each exercise.

The earlier exercises are the easier ones to do. The later ones are more difficult. The approximate "degree of difficulty" of each exercise is listed with the exercise.

Compound IF functions

Overiew
The IF function is one of the most important in spreadsheets as it allows you to customise calculations according to various criteria or contexts. One application of the IF function is to determine whether limits / thresholds or barriers are crossed. And that is the purpose of the following exercise.

Degree of difficulty (1=easy, 10=extremely difficult)
3

The exercise
You need to design a "compound" or "nested" IF statement to test whether an amount is within two thresholds. There are two thresholds: An upper threshold and a lower one.

Our Visual Basic course shows how to extend or change Microsoft Office products' functionality

- If the amount is above the upper threshold then your formula should show **above**
- If the amount is below the lower theshold your formula should show **below**
- If the amount is between (but not including) the thresholds your formula should show **between**
- If the amount is equal to the lower threshold the formula should show **at lower**
- And if the amount is equal to the upper threshold the formula should show **at upper**

Put a formula into cell F52 below to complete the exercise.

	D	E	F	G	H	I	J	K
46								
47	Upper threshold		37					
48	Lower threshold		35					
49	Amount		38	<<- You will need to use different values for Amount				
50				to test if your formula (below) works for all values.				
51								
52	Answer			**<<- Put your formula into this cell**				
53								
54	**Status of your answer**							
55	Please complete the answer.							
56								

[See summary of answer status of all exercises].

Conditional counting

Our Visual Basic course shows how to automate routine work and save the time and trouble of doing the same thing over and over

Overview
This exercise involves counting conditionally. The best formulae to use are either SUMPRODUCT or a combination of COUNTIF and IF.

Degree of difficulty (1=easy, 10=extremely difficult)
5

The exercise
In cell L20 put a formula that gives the number of times the departmental performances were between the levels selected in the blue cells on row 20. For example, if the user selects B and D (or D and B) then the formula should return the number of times performances were B,C or

	D	E	F	G	H	I	J	K	L
17	Month:			Oct-09	Nov-09	Dec-09	Jan-10	Feb-10	Mar-10
18	Dept. Performance level:			A	B	C	B	D	C
19									
20	Number of times performance was betweer				A	is		<- Your answer	
21			and (including)		B				
22	**Calculations**								
23									
24									
25									
26									
27	**Status of your answer**								
28	Please complete the answer.								
29									
30									

[See summary of answer status of all exercises].

Back to contents

Back to top

Tracking cashflows

The exercise below is about finding the size and "direction" of cash flows in two accounts. To complete the exercise you will need to design some formulae to calculate: 1) Whether cash is going into or out of accounts, 2) how the cash flow is "split" between the accounts, and 3) the amount involved.

Degree of difficulty (1=easy, 10=extremely difficult)

2

Look at the spreadsheet below. The spreadsheet keeps track of cash flows and balance in two accounts: Account #1 and account #2. Both accounts can be in credit or debit.

The initial balances in the two accounts are shown in cells G35 and G36. There is a limit on how much in debit account #1 can be. That limit is shown (as a negative number) in cell G33.

Cell G38 defines a cash flow. The cash flow can be positive (i.e. incoming) or it can be negative (i.e. going out). If the cash is coming in then it needs to be deposited in one or both of Accounts #1 and #2.

This is a rule that defines what needs to be done with a positive cash flow. The cash flow is first applied to reducing Account #1's debit balance. If the cash flow is greater than the debit balance then only enough is deposited into that account to set the balance to zero. Any residual amount from the cash flow is deposited into Account #2.

So, for example, if Account #1 has a debit balance of (17,000) and a cash flow of 20,000 occurs then 17,000 is deposited into Account #1 and 3,000 into Account #2. If, on the other hand, a cash flow of 6,000 occurs then all of that amount is deposited into Account #1.

This is a rule that defines what needs to be done with a negative cash flow. A withdrawal is made from Account #1 equal to the amount of the cash flow. However, Account #1 cannot go into debit to more than the amount shown in cell G33.

So if there is insufficient capacity in Account #1 to pay the entire negative cash flow the residual amount must be withdrawn from Account #2.

	D	E	F	G	H	I	J	K	L	M	N
33	Facility limit Account #1			(30,000)		**Calculations**					
34						Extra workspace isn't really required for this question -					
35	Opening acc. #1 balance			(17,000)		but here's some anyway.					
36	Opening acc. #2 balance			-							
37											
38	Cash surplus / (need)			20,000							
39											
40	Acc. #1 deposit / (withdrawal)				Please complete the answer.						
41											
42	Acc. #2 deposit / (withdrawal)				Please complete the answer.						
43											
44	Closing acc. #1 balance				Please complete the answer.						
45											
46	Closing acc. #2 balance				Please complete the answer.						
47											

[See summary of answer status of all exercises].

Back to contents

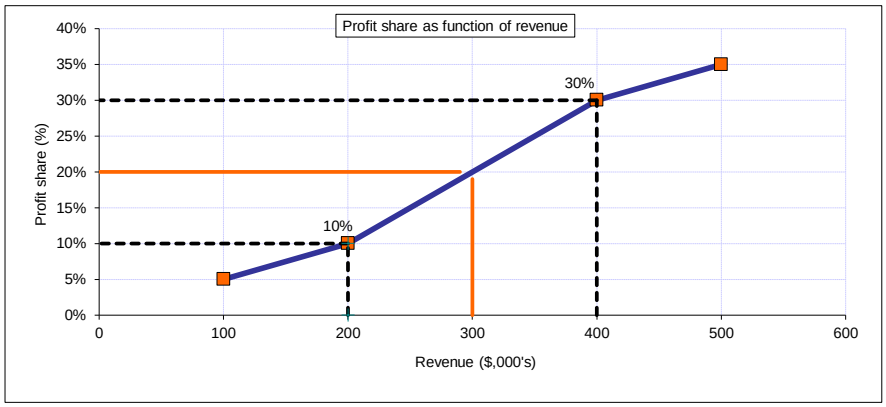
Back to top

Interpolating

Our Visual Basic course shows how to extend or change Microsoft Office products' functionality

Overview

Interpolating involves "filling in the gaps" in tabular or graphic data. The chart below provides an illustration. The chart



Four points (shown by the coloured squares in the chart above) define particular revenues and profit shares. For a revenue of 100 (and below) the profit share is 5%, for a revenue of 200 it is 10%, for 400 it is 30% and for 500 (and above) it is 35%.

For revenues other than 100, 200, 400 and 500 a "sliding scale applies". For example, if the revenue is 300 (i.e. midway between 200 and 400), then the profit share is midway between 10% and 30%. This "midway" scenario is

Our Visual Basic course shows how to extend or change Microsoft Office products' functionality

Degree of difficulty (1=easy, 10=extremely difficult)

8

The exercise

You need to calculate the percentange share for a number of revenues. Your answer (i.e. formulae calculating percentages) needs to be put into cells $G$50:$K$50 in the spreadsheet below. The table defining revenues and associated percentage shares is in cells $G$44:$J$45.

You will probably need to do some intermediate or 'working' calculations. Those calculations can be done below in the area with a light yellow background in cells $E$53:$K$60.

	E	F	G	H	I	J	K	L	M
43	**Revenue share table**								
44	Profit $,000's		<= 100	200	400	>= 500			
45	Revenue share (%)		5.00%	10.00%	30.00%	35.00%			
46									
47	**Forecast profit and revenue share**								
48	Period ending		31/12/10	31/12/10	31/12/10	31/12/10	31/12/10		
49	Profit $,000's		50	125	210	450	550		
50	Revenue Share (%)							**<- Final formulae go here**	
51									
52	**Calculations**								
53								You can use the area on the left for intermediate / working calculations. Your final answers should go into row 50 in cells G50:K50.	
54									
55									
56									
57									
58									
59									
60									
61									
62	**Status of your answer**								
63	Please complete the answer.								
64									

[See summary of answer status of all exercises].

Back to contents

Back to top

Complex lookups

You are modelling the operations of a business. The operations will be impacted by the business's dependence on a critical resource. That resource is expected to be unavailable over certain periods. Those periods are described in the "Interruption schedule" below.

There may be zero, one or more interruption periods in any month. Interruption periods may span month ends and month beginnings. An interruption period may be as short as one day or as long as several months. An interruption period may already be in effect at the start of the selected month and another (or the same) may persist past the end of that month.

Degree of difficulty (1=easy, 10=extremely difficult)
10

The exercise
Calculate the number of productive (i.e. uninterrupted) days in the selected month. Your answer should work even if the information in the interruption schedule is changed (i.e. you shouldn't "hard-code" your answer.)

	D	E	F	G	H	I	J	K	L	M
22	Between		1-Jul-10					*Interruption schedule (inclusive*		
23	... and		31-Jul-10					*start and end dates)*		
24	... the number of un-							*Start date*	*End date*	
25	interrupted days is:			**<- Your answer**				30-Dec-09	2-Jan-10	
26								5-Jan-10	6-Jan-10	
27								31-Jan-10	31-Jan-10	
28								27-Feb-10	2-Mar-10	
29								2-May-10	1-Jul-10	
30	**Calculations**									
31										
32										
33										
34										
35										
36	**Status of your answer**									
37	Please complete the answer.									
38										

[See summary of answer status of all exercises].

Back to contents

Back to top

Status of Exercises

Following is the completion status of the exercises in this section.

Participants on our courses have the opportunity to do on-line pre and post-course self-evaluations.

Exercise	*Degree of difficulty (**	*Completion status*
Compound IF functions	3	Not attempted
Conditional counting	5	Not attempted
Tracking cashflows	2	Not attempted
Interpolating	7	Not attempted
Complex lookups	10	Not attempted

* = Degree of difficulty on a scale from 1 to 10 - 1 is easy and 10 is very difficult.

Back to contents

Back to top

Chapter 5 - Next steps

Our modelling and spreadsheet skills courses can be done in one or two day modules.

Congratulations on having made it up to here. We hope that by now you will have:

- Looked at examples of the kinds of things you can do with spreadsheets (in Chapter 1)
- Reviewed some of the important spreadsheet functions (in Chapter 2)
- Studied how the "building blocks" in Chapter 2 can be combined to build "mini-applications" (in Chapter 3)
- Attempted one or more of the exercises listed in Chapter 4.

So what's next? These are some possibilities:

Do a free on-line self-evaluation of the skills covered in this guide.

Look at workshops we offer that cover in more depth the concepts touched on in this guide.

Submit suggestions for improving or feedback about this guide.

Contents

Index

Back to contents Back to top