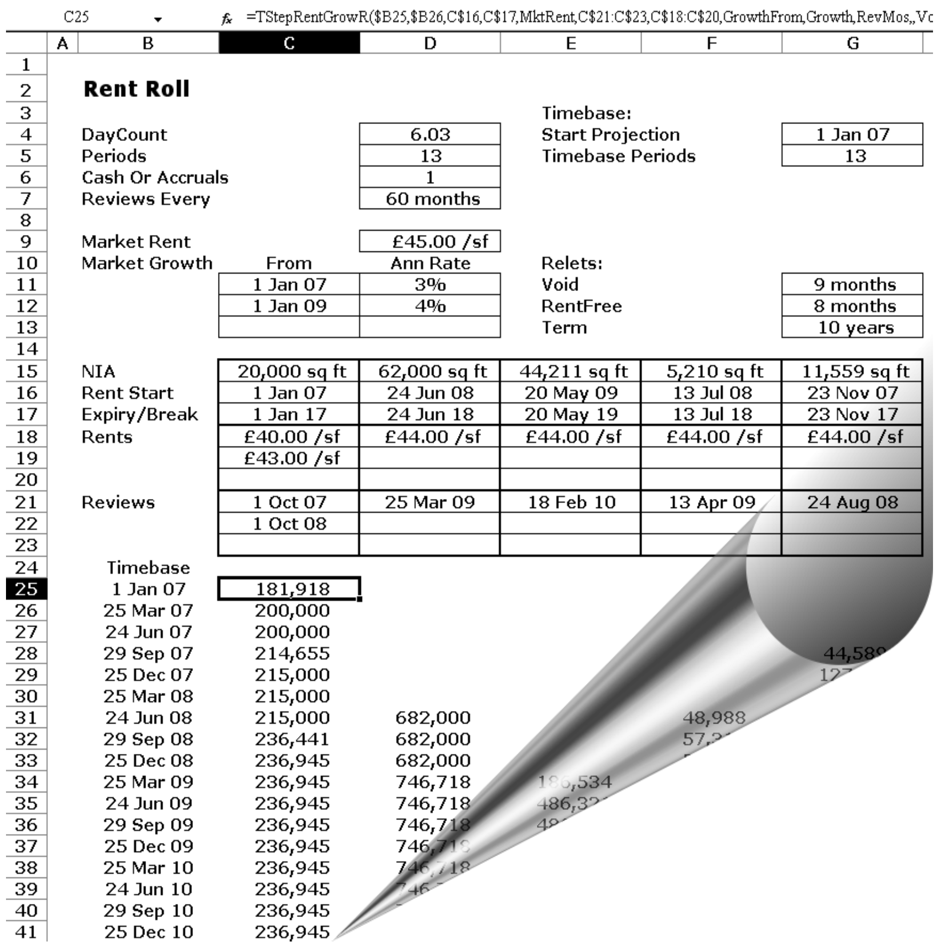
Introduction to Property Cash Flow Modelling Using Business Functions



C25 =TStepRentGrowR($B25,$B26,C$16,C$17,MktRent,C$21:C$23,C$18:C$20,GrowthFrom,Growth,RevMos,,Vc

	A	B	C	D	E	F	G
1							
2		**Rent Roll**					
3					Timebase:		
4		DayCount		6.03	Start Projection		1 Jan 07
5		Periods		13	Timebase Periods		13
6		Cash Or Accruals		1			
7		Reviews Every		60 months			
8							
9		Market Rent		£45.00 /sf			
10		Market Growth	From	Ann Rate	Relets:		
11			1 Jan 07	3%	Void		9 months
12			1 Jan 09	4%	RentFree		8 months
13					Term		10 years
14							
15		NIA	20,000 sq ft	62,000 sq ft	44,211 sq ft	5,210 sq ft	11,559 sq ft
16		Rent Start	1 Jan 07	24 Jun 08	20 May 09	13 Jul 08	23 Nov 07
17		Expiry/Break	1 Jan 17	24 Jun 18	20 May 19	13 Jul 18	23 Nov 17
18		Rents	£40.00 /sf	£44.00 /sf	£44.00 /sf	£44.00 /sf	£44.00 /sf
19			£43.00 /sf				
20							
21		Reviews	1 Oct 07	25 Mar 09	18 Feb 10	13 Apr 09	24 Aug 08
22			1 Oct 08				
23							
24		Timebase					
25		1 Jan 07	181,918				
26		25 Mar 07	200,000				
27		24 Jun 07	200,000				
28		29 Sep 07	214,655				44,58
29		25 Dec 07	215,000				12
30		25 Mar 08	215,000				
31		24 Jun 08	215,000	682,000		48,988	
32		29 Sep 08	236,441	682,000		57,3	
33		25 Dec 08	236,945	682,000			
34		25 Mar 09	236,945	746,718	186,534		
35		24 Jun 09	236,945	746,718	486,32		
36		29 Sep 09	236,945	746,718	48		
37		25 Dec 09	236,945	746,71			
38		25 Mar 10	236,945	746,718			
39		24 Jun 10	236,945	746			
40		29 Sep 10	236,945				
41		25 Dec 10	236,945				

Business Functions

Oct 2005

Contents

What this booklet is about

The aim of this guide is to introduce you to financial modelling with two particular fields of emphasis. First, clearly, is the use of Business Functions, which I think can enable you to achieve high standards of financial modelling, Secondly, it focuses on an area I am qualified to write on, which is property, cash flows and loans. At the end of it I hope you can understand what financial modelling really is, what is generally best practise, and how to use Business Functions for real estate, cash flows and loans. There are companion workbooks and further resources detailed in the Appendix.

Why Excel?

On question to consider is why one should use Excel at all for property and loans, because there are in existence many financial packages that cover property valuation and forecasting, cash flows, consolidation and budgeting. The simple answer is that most businesses use Excel for this area of activity, whether its ‘right’ to or not, and therefore we need to use Excel in the best way possible to achieve the goal. The more thoughtful answer is that Excel offers the flexibility in calculation and presentation that most people consider invaluable. Excel is, after all, a numeric scratchpad, albeit a sophisticated one, and as such is capable of doing anything you ask.

The problem is that it doesn't do some things very well, but that doesn't stop people trying because they crave the aforementioned flexibility. The philosophy of Business functions is to swim with the tide, and not against it, and provide you with more tools to extend Excel's capability, in particular make more use of functions, which are a very robust way of packaging modelling functionality. One day, maybe Microsoft itself will build in more structure and functionality for financial modelling, but recently Excel's basic functionality has been fairly static from release to release, probably because no-one really agrees on how you could make it better and eliminate the deficiencies of the spreadsheet.

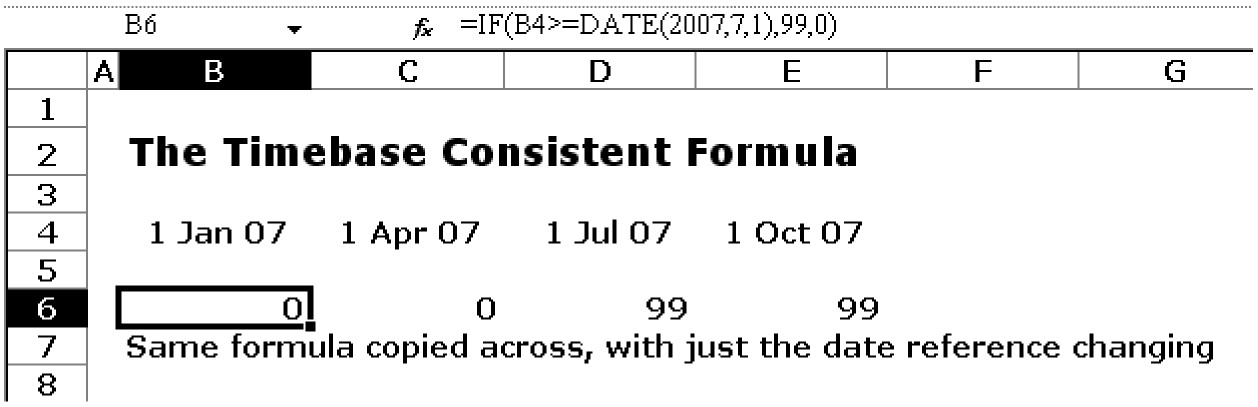
Philosophy

One idea that encapsulates everything I could say about this to 'think like a programmer'. Most people would say that they have no idea how programmers think, but I would insist that by using a spreadsheet you are, in effect, a programmer, so you had better think like one or you will make basic mistakes. Do not be afraid, for instance, to dabble in programming languages like Visual Basic, PHP , Pascal or C++. I don't think you should use these in your day job, because its just too technical to fill one’s head with all that detail, but the exercise of tinkering will give you some of the data structuring and program flow discipline that you need to most efficiently use Excel. The 'best practise' ideas discussed here really originate in a roundabout way from classical programming. The other thing about thinking like a programmer is that you accept that errors *do* happen and are constantly evaluating how they could be reduced.

Timebase consistent formulae

My pet technique is the timebase consistent formula. Amongst those who write on the subject of cash flow modelling there is a consensus that this a good thing, and also that its very difficult to do. But first of all, what do we mean by a timebase consistent formula? A timebase consistent formula is one that is the same across the entire timebase, allowing for the fact that the time references in it might be allowed to change to reflect the current period. The fact remains that that the fundamental logic remains the same across the entire time horizon of the model. Why do this, especially if its difficult? The reason is, to turn your spreadsheet into a *model* and not just a *grid of numbers*.

What we are seeking is a proper model is that we want to re-use this model for different situations, and we want to be able to adjust it year on year for the latest data and assumptions. If we just put numbers in cells on a time grid we will always be shifting things around and not knowing where a certain number came from. If we can turn the spreadsheet into a proper functioning model we can drive it just off the assumptions that we can keep in a box or sheet labeled 'assumptions', and we will all know where we are.



```
B6        fx  =IF(B4>=DATE(2007,7,1),99,0)

The Timebase Consistent Formula

 1 Jan 07    1 Apr 07    1 Jul 07    1 Oct 07

        0           0          99          99
Same formula copied across, with just the date reference changing
```

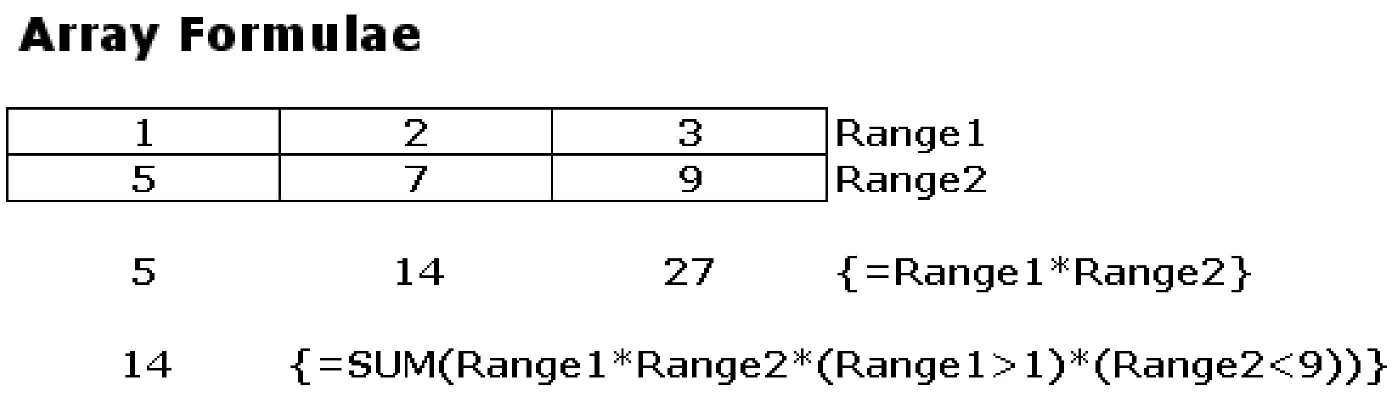
The real heights of modelling flexibility are reached if we can say that the timebase consistent formulae we have used across the time base are actually *independent* of that timebase, In other words:- you can make the timebase what you like, the formula will cope with the projection. The so-called *projections functions* within Business Functions have this as their goal, and, thanks to the date arithmetic and some pretty painful testing, they succeed. You can change the timebase and the Business Function will respond correctly.

Standard Tools and Techniques That Excel Places at Your Disposal

Before we start modelling with Business Functions, it’s worth looking at what Excel offers you as its basic repertoire of tools and techniques, and review which ones are more useful than others. Along the way we will see that there is debate about the usefulness of some, and I shall not withhold my own opinion, which you can take for what it is worth. It’s taken as a given that Excel is multi-page spreadsheet with automatically recalculating formulae. Now lets take a look at those features.

Array Formulae

Normally a formula exists in a cell returns a single number and quite often we copy these over a a wide area of the spreadsheet. However, each formula just returns one number. There is no reason why a formula can't return more than one number, its just that the language we normally use for the formula (A1*A2 etc) usually predisposes us to the idea that just one number will be returned. Also, we have got



Array Formulae

1	2	3	Range1
5	7	9	Range2

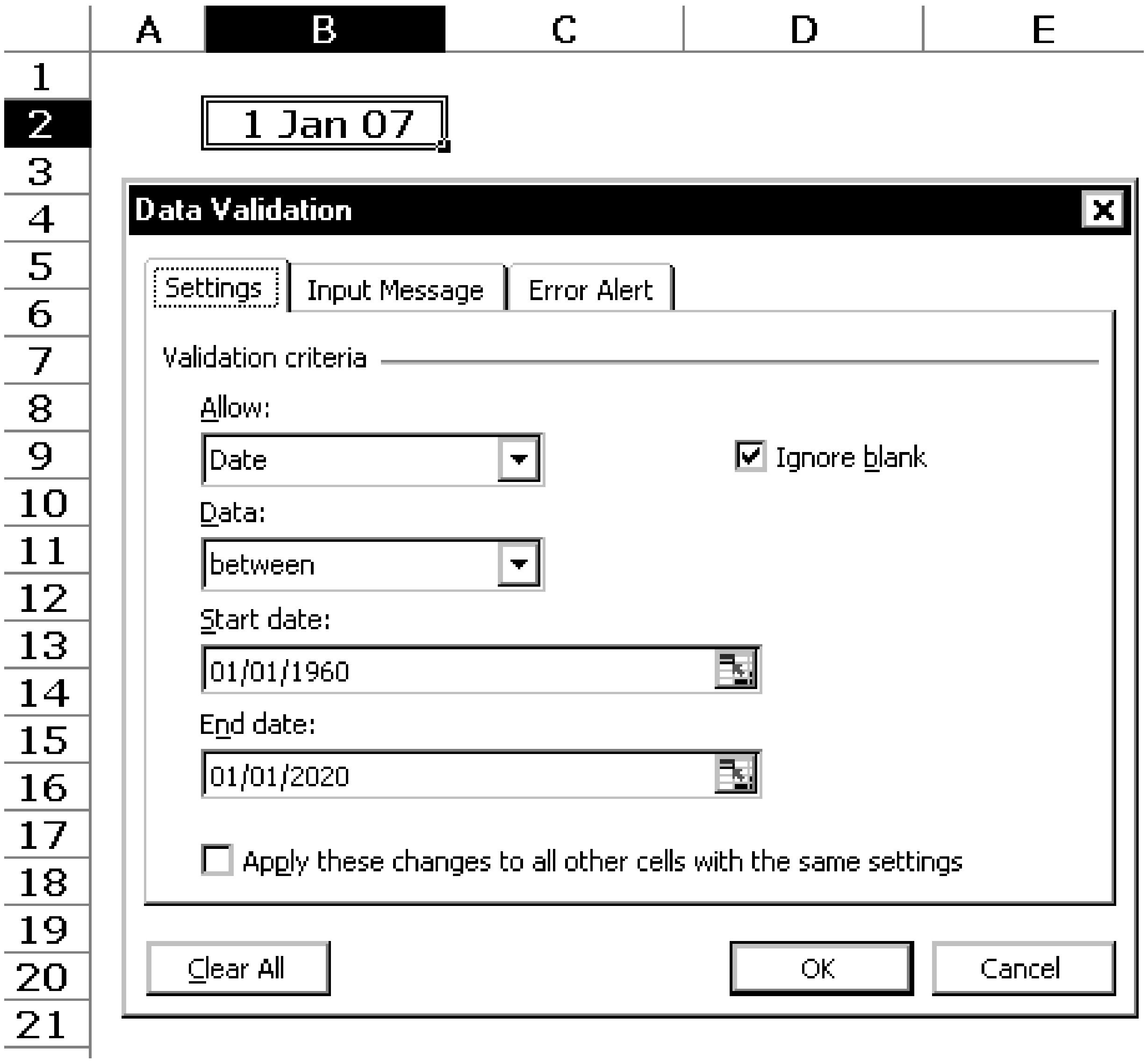
```
5        14        27      {=Range1*Range2}

14    {=SUM(Range1*Range2*(Range1>1)*(Range2<9))}
```

used to the idea that a formula goes in one cell, when in fact Excel allows you to put a formula - the same formula - into a range of cells. This turns out to a powerful technique, because it sets you free from the one result restriction, and gives you what programmers have always had - the ability to return an array.

I should say that array formulae do have their detractors, who say that they can slow a spreadsheet down and are just too complicated. I have never seen evidence for the first assertion, although I can imagine how the dependency tree (Excel's internal map of the order things get calculated) could have to work harder. My own tests lead me to believe that, properly applied (i.e. not using them when you just don't need them) they speed things up. The charge that they are too complicated is one I have sympathy with, because it seems our brains start to get nervous about arrays of numbers being flung around in calculations. The best thing about array formulae is that they are computationally pure - they are not a hack, a fudge or a feature. Later on we should see some examples of array formulae and later still BF's array functions, but for now they are on the 'highly useful' list.

Data Validation

Data validation is a great technique for stopping the user inputting invalid entries into a cell. You can look up how it works in the Excel Help, or use Business Functions 'Validation Standards' under the Utilities menu. With Data Validation you can ensure that only a date is entered where a date should be, and even *what* that date could be. Essentially, if you can write a formula for what is valid and what isn't, you can incorporate this test into the data validation for a cell.



Allow: Date

Data: between

Start date: 01/01/1960

End date: 01/01/2020

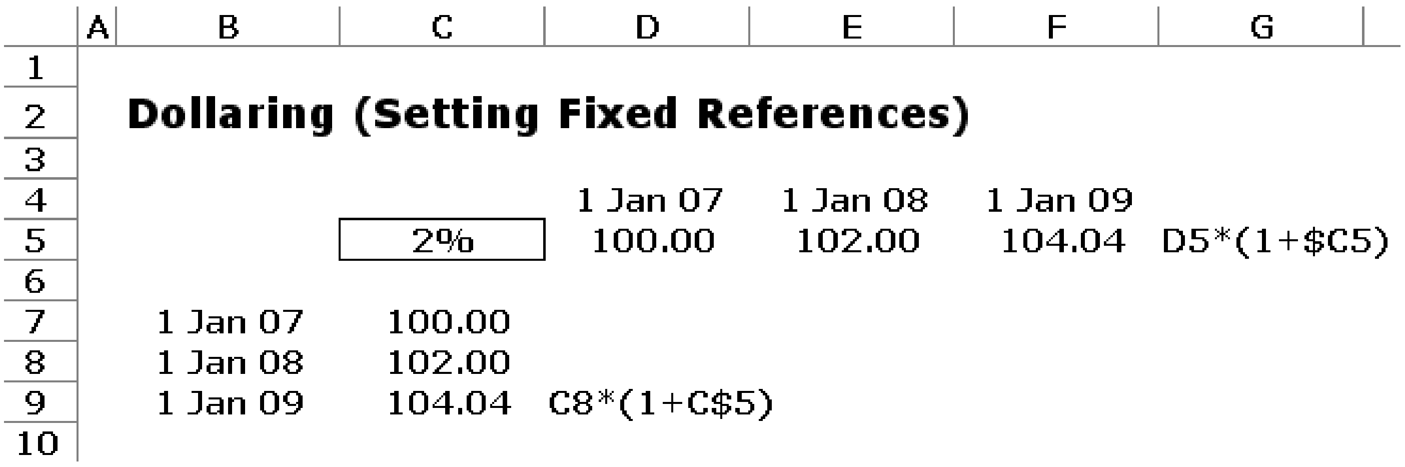
☑ Ignore blank

☐ Apply these changes to all other cells with the same settings

There are shortcomings with DV (you can get around it), but it goes a long way to preventing invalid data entry, and the only reason people don't use it everywhere is they don't know about it or can't be bothered. So use it as much as you can, and the BF validation standards aren't a bad starting point.

Dollaring

You need to understand how relative and absolute references work because you are frequently copying formulae across a time range, and you need to specify which cell references stay fixed and which ones are allowed to move with the copy. Inserting a dollar before the letter (e.g. $A1) or number (e.g. A$1) of a cell reference locks respectively the column or the row. Double dollaring (e.g. A1) locks the reference in both directions. You can cycle through the various dollaring options by pressing F4 repeatedly whilst you are editing a cell reference.



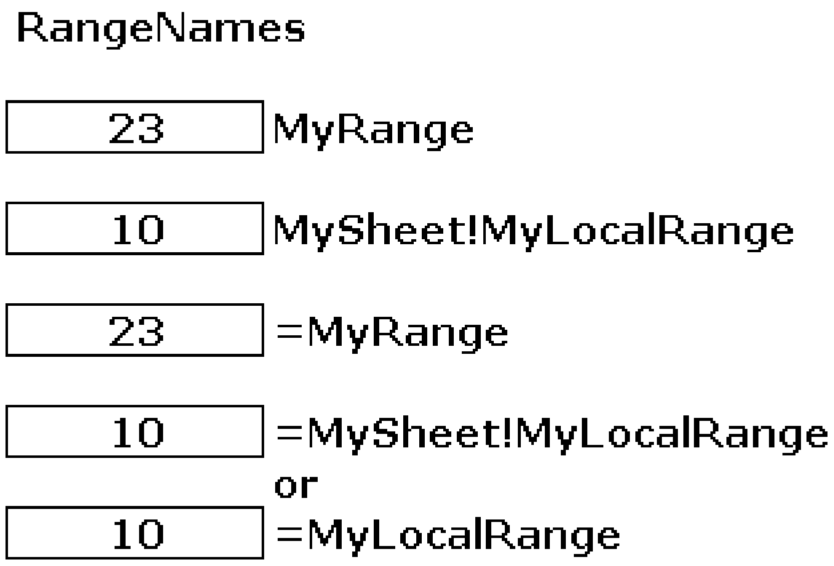
	A	B	C	D	E	F	G
1							
2		**Dollaring (Setting Fixed References)**					
3							
4				1 Jan 07	1 Jan 08	1 Jan 09	
5			2%	100.00	102.00	104.04	D5*(1+$C5)
6							
7		1 Jan 07	100.00				
8		1 Jan 08	102.00				
9		1 Jan 09	104.04	C8*(1+C$5)			
10							

Range Names

Excel allows you, like Lotus 1-2-3 before it, to name certain cells or ranges, so that in future you can simply refer to them by their range name rather than the cell reference. This has to be *a good thing*, because it changes a formula from meaningless cell references to something that, depending on your choice of name, actually reads in English. In general, be free to be verbose in the name you choose, we are trying to impart information to the hapless user here, which might be you.

There are one or two disadvantages of range names which you should be aware of. Firstly, unlike a cell reference, a range name doesn't tell you *where* in the work-book the blessed name is, and you end up looking around for it. Secondly, rangename-itis can set in, where you start rangenaming everything, and if this happens you should learn to relax a bit. Cell references are not bad, they are just a bit cryptic, and if you don't do your names until later, you can always apply a name to all the formulae in the workbook with Insert/Name/Apply. The rule I use is that if you are going to double-dollar (absolutely refer) a cell reference, you should probably be giving the source cell a range name, which by definition are absolute references.

One rather useful and little used feature of range names is *worksheet specific, or local rangenames*. These are range names that refer to a cell reference on a particular worksheet and you assign them by typing the sheetname, followed by a bang (exclamation mark) and your name. So the sheet specific range name of MyName is MySheet!MyName. Then, you can refer to the name using its full sheet specifier (MySheet!MyName), or, if your formula is local to the sheet where the name is, by just the local reference (MyName).



```
RangeNames

23    MyRange

10    MySheet!MyLocalRange

23    =MyRange

10    =MySheet!MyLocalRange
      or
10    =MyLocalRange
```

Excel's rule for looking up names in formulae is that it first of all looks for a local version of the name (on that worksheet), then it looks globally in the workbook. I believe that after that it starts to at other workbooks in memory, which is a horrible thought and you shouldn't do this intentionally. Also, a local range name is simply a normal name with its scope restricted - so you can’t have a local and global version of a name co-existing on the same sheet. All this gives us a quite robust range naming system provided we can work our way around it.

A very nice consequence of local range names is that if you have multiple instances of a type of object that you describe on a worksheet like a tenant or a loan, you can create a new worksheet and when you copy the old worksheet into the new worksheet the formulae in the new worksheet refer to the *local* versions of the range names they use. Not only that, but Excel creates the local range names for you. This is the only way that Excel will automatically create names for you, and rather points you at a technique for multiple worksheet models - that of lots of identical sheets dealing with different customers or tenants.

VBA Functions

You can write your own function library using VBA (Visual Basic for Applications). Its not hard at all and most advanced Excel textbooks will teach you how. There are problems with VBA in my view, which is why Business Functions uses VBA to the absolute minimum, preferring a separate compiled C++ library.

I have often encountered reliability problems with VBA, but the most concrete drawback is is is slow, several times slower than C++ and much slower that native Excel functions. The real performance hit seems to be in the call to the function itself, but looping through cell ranges in an un-optimised, interpreted language like Visual Basic is never going to be quick.

Advanced users may be interested to know that all the examples in this booklet have been done using VBA functions (code provided in companion workbooks) as well as Business Functions, but always with the restrictions that you will encounter later in this booklet, which is that they are not truly independent of the timebase and can't really cope with time properly. The reason for this is that proper date arithmetic would have made them too slow. As it is, we experienced great performance improvements in VBA by making them array functions, because that reduced the number of function calls to just one for a single line item across the timebase.

So, should you use them? My answer is no, and any VBA functions we had in Business Functions we have removed because of performance and reliability issues. These reasons are subjective, however, and you may want to experiment. If you do, our advice is make them array functions wherever possible and also that the variant data type is really rather useful. You may also consult the VBA module in the workbooks that accompany this booklet.

Multiple Worksheets

I mention this because its pretty well standard that you have a number of sheets in your model, not just one. However, there are some authors that argue these should be kept to minimum, because its hard to trace computation flow across worksheets. There are some who go further and advise only ever having one (presumably enormous) worksheet of calculations. This is overly draconian and you will need several worksheets in your model. My own guidelines for this are:

- You need a sheet called 'Globals' where you keep global variables that get used everywhere but which you probably don't want right up front for all to see.
- Key variables should often (not always) live on a sheet called Assumptions.

- You probably need a new sheet at least as often as whenever the timebase is different (eg annual on one sheet, quarterly on another).
- The model's structure should be more or less be the same as the paper output. If you have several sheets, say for rent, service charge, inducements, etc, my advice is to have a schedule for each page and form a landscape A4 'book' - and the worksheet should reflect this, in the same order as the printout. That way, if you have the printout, you can successfully navigate the model. The 'single calculation sheet' is in my view a disaster for working out what's going on - there's something very uncomfortable about trawling the depths of cell CF1847 - I prefer something more like Fitout!H27 - but maybe that's just me!

Pivot Tables and Excel's Database functionality

The day might come where I see the light on these, but for now I advise just saying no. Pivot Tables are for providing different views of data, for slicing and dicing in different ways, and they are part of what I would call Excel's database functionality.

Well, Excel's database functionality is, in my opinion, not really good news. Excel is not a database. A database, in the modern world, must mean a collection of tables that are related (relational) and indexed to make lookups super-fast. Excel doesn't have this functionality. Excel, purely by virtue of its rows and columns structure, is a list manager. This is useful when you have a list of tenants, customers or loans. But you can't relate one list to another (say tenants to buildings), except by doing it yourself, and the lookups you do will be unindexed which will mean they get slow. Once you get past 50 or 100 items that you want to sort etc I think you should use a desktop database like Microsoft Access.

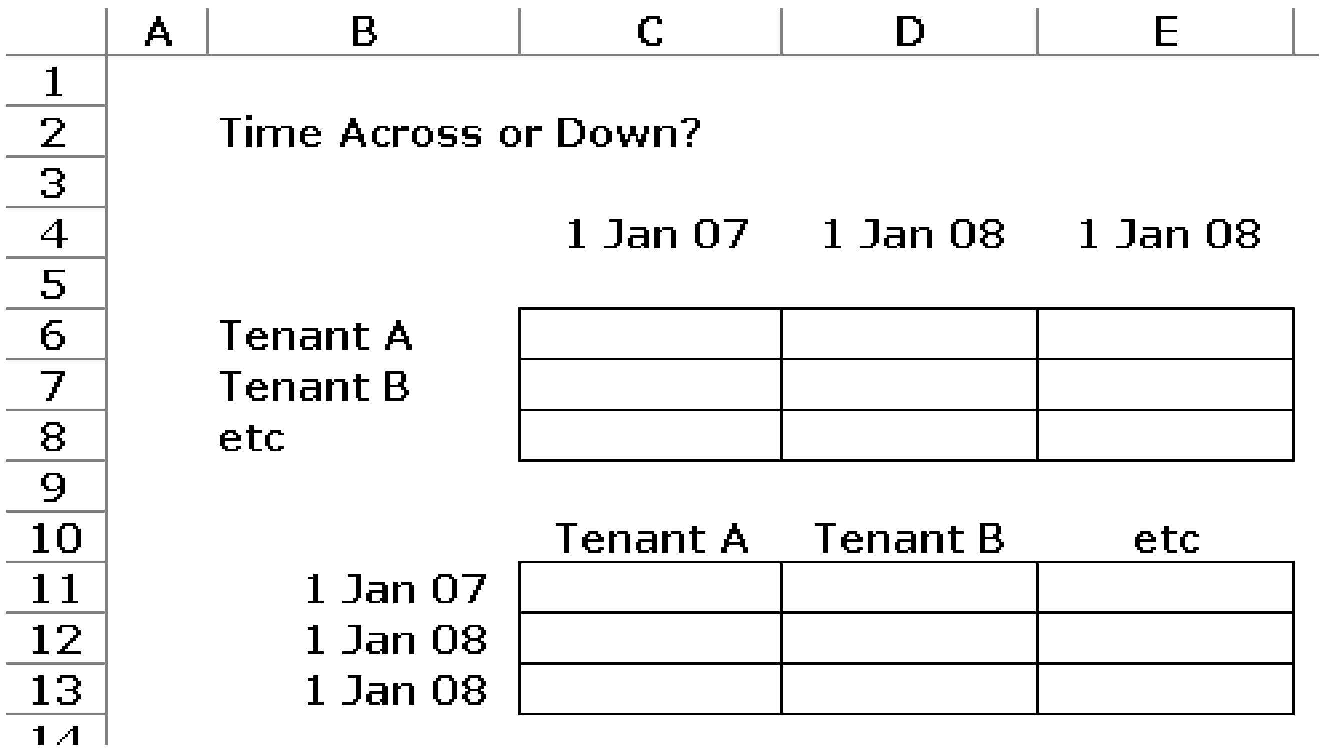
So getting back to Pivot Tables, they are tarred with the accusation that they are part of Excel's database functionality, which in my book is a problem. Also, they allow you, the user or anyone to change their view of the data. This does not sound healthy. With spreadsheets we are always looking to reduce the source of errors, and not finding things where you expect them is a prime cause of error, causing you to pick the wrong interest rate, wrong rent or whatever. So Excel's database functionality is, in the wrong hands, going to break your models - marvelous, that's just what we needed!

Range Context Formula

These I have to confess I am still in two minds about. They are very clever, but, well ...I just don't know. Anyhow, how they work is like this. You have range names Revenues and Costs that refer to two rows in the spreadsheet. If you type into a cell ‘ =Revenues-Costs’, excel does something rather clairvoyant. It looks up the item in the row or column that corresponds a cell in Costs and deducts it form a corresponding cell in Costs. So although it looks like you have tried to deduct one array from another, Excel knows you didn't mean that and therefore selects the most appropriate item in each range to do the calculation.

This is very ingenious, but to me it breaks the rules of Excel. As a 1 dimensional array Revenues is legitimate. For Excel to second guess that you just want one item out of that array is a darn right liberty. If a formula doesn't make sense Excel should just tell you, not try to be clever. To extend this, as some have done, into a mainstream technique is asking for trouble and it finds it, in the shape of MAX() and MIN() functions (e.g. (MIN(Revenues,Costs)) because at this point Excels extra-terrestrial intelligence runs out and reverts to normal behaviour. My view is that, if the technique doesn't work across the board, it shouldn't be used - but its a very good technique with only one or two rough edges. Another way of achieving the same thing without range context formulae is an array formula like ({=Revenues-Costs}, which in some ways better reflects what you are trying to do, which is deduct two arrays, although it suffers some of the same problems.

Time across or Down



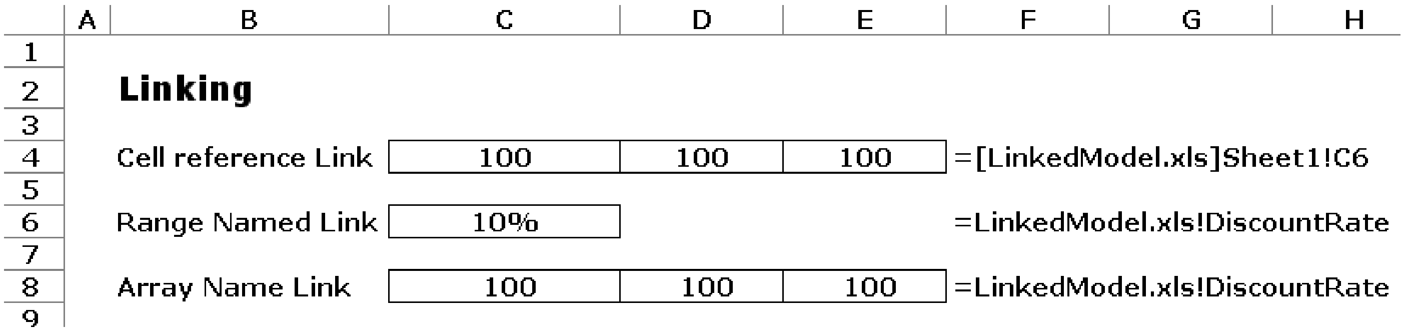
	A	B	C	D	E
1					
2		Time Across or Down?			
3					
4			1 Jan 07	1 Jan 08	1 Jan 08
5					
6		Tenant A			
7		Tenant B			
8		etc			
9					
10			Tenant A	Tenant B	etc
11		1 Jan 07			
12		1 Jan 08			
13		1 Jan 08			

The debate rages about whether a cash flow should go across or down. I think it depends and either is right in certain circumstances. Timebase across is good where there are lots of lines of rent, costs etc. Its not that great when you have lots of time periods or times because scrolling across is not a very comfortable thing to do. In these cases, its better to go down the page with your times, provided you don't have too many line items.

If in doubt, most people prefer across the page, which is why a particularly good business plan format is sideways A4 or A3, with time across the top. A good valuation format is A4 portrait with time down the side.

Workbook Linking

As a rule, this is a very error-prone area.. Hotlinking, or dynamic linking is dangerous because you don't have control over the source spreadsheet. It might not even be *your* spreadsheet and could be changing all the time. Also, inserting rows and columns in the source spreadsheet can easily destroy the link. If you must hotlink, its much better to link by referring to rangenames (eg SourceSheet.xls!SourceRangeName). This technique also works just fine for linking to a range of sveral cells, because you can just array-enter the link (eg {=SourceSheet.xls!SourceRangeName}).



	A	B	C	D	E	F	G	H
1								
2		**Linking**						
3								
4		Cell reference Link	100	100	100	=[LinkedModel.xls]Sheet1!C6		
5								
6		Range Named Link	10%			=LinkedModel.xls!DiscountRate		
7								
8		Array Name Link	100	100	100	=LinkedModel.xls!DiscountRate		
9								

I should say that we have an add-in on the Business Functions website called Cashflow Administrator that we think is a better way to link spreadsheets together, because its not a live link, but a documented, audit-trailed data transfer utility specifically for cashflow data.

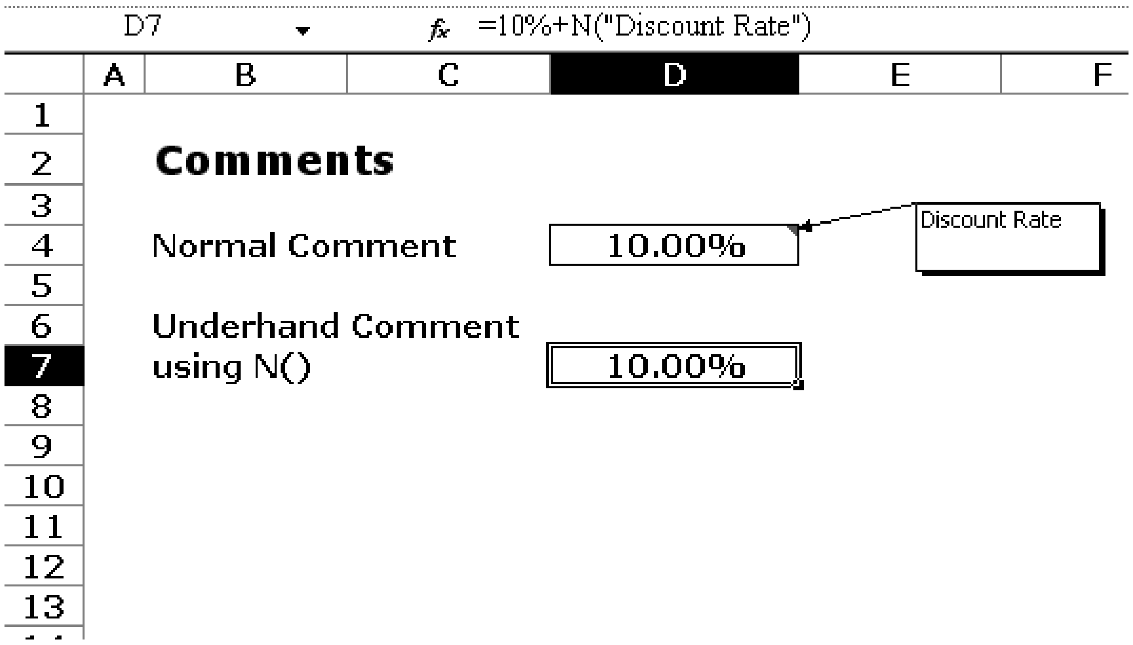
Little Hacks

These little hacks are small, nifty workarounds to common issues. There are so many little tricks in Excel that you can't remember them all, but I think you would find the following ones useful.

Adding a comment to a formula

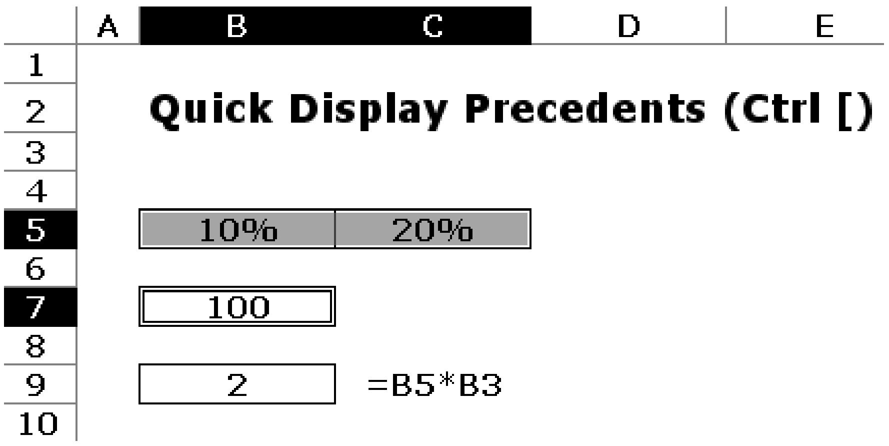
Excels standard comments are for many people rather unsatisfactory. It’s hard to pin down just what the problem with them is. They seem to misbehave and pop up when you don't want them and they have a habit of getting very out of date, probably because they are a bit of pain to edit. A fundamental flaw in them is that the option about whether and how to display them is left to the settings on the users machine, rather than being a property of the spreadsheet itself.

Time after time I find that once Excel strays out of its basic formula paradigm in this way it starts to get clumsy. So one way of putting your comment right with the formula is to include it as part of the formula, but in a totally inert (ie doesn't do anything) way. The function N() is actually supposed to transform something into a number, but used with text it just returns zero. So you can happily go: =(1+r)^(1/4)+N("This is the adjustment to a Quarterly discount rate") and you've added a comment in a different way, a way which more people are likely to read incidentally, because they will think "what on earth's that N() function doing there?" and you therefore instantly have their attention, which is what you want.


D7 =10%+N("Discount Rate")

	A	B	C	D	E	F
1						
2		Comments				
3						
4		Normal Comment		10.00%		
5						
6		Underhand Comment				
7		using N()		10.00%		
8						
9						
10						
11						
12						
13						

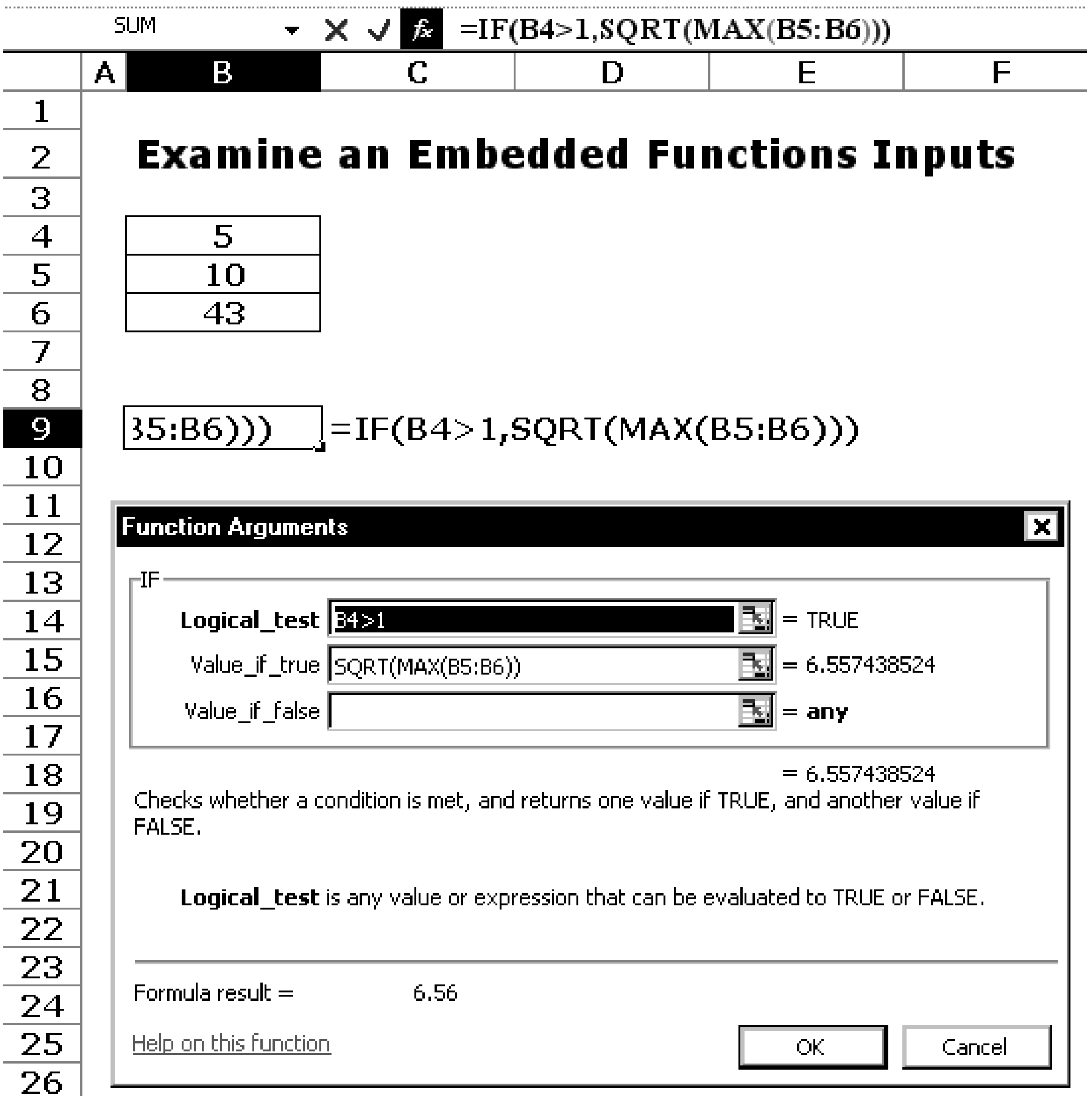
View Precedents and Dependents short cuts


	A	B	C	D	E
1					
2		Quick Display Precedents (Ctrl [)			
3					
4					
5		10%	20%		
6					
7		100			
8					
9		2	=B5*B3		
10					

I have to confess I never remember any keyboard shortcuts for anything, I am the sort of person that never uses the number keyboard because he's forgotten where it is (somewhere in the murky region on the right of the keyboard where a coffee cup lurks). And as for toolbars, well, I can't work out the icons. But Ctrl-[and Ctrl-] do a very useful job by showing you, in a flash, what the inputs of the formula you are on are, and what figures are affected by that formula. No auditing toolbars or menus - just a keystroke.

Evaluate a function embedded in the depths of a long formula

A lot of useful functionality about existing formulae is buried in the Insert Excel menu, which I always find confusing and slightly scary because it feels like you might accidentally insert something. To find about range names for instance you use Insert/Name/Define which doesn't necessarily do any inserting at all. The same is true of functions. To insert a function (even a Business Function) you can use Insert/Function (Alt-I-F). But where Alt-I-F comes into its own is where you are trying to evaluate an existing function.


SUM =IF(B4>1,SQRT(MAX(B5:B6)))

Examine an Embedded Functions Inputs

4	5
5	10
6	43

9: 35:B6))) =IF(B4>1,SQRT(MAX(B5:B6)))

Function Arguments

IF

Logical_test B4>1 = TRUE
Value_if_true SQRT(MAX(B5:B6)) = 6.557438524
Value_if_false = any

= 6.557438524

Checks whether a condition is met, and returns one value if TRUE, and another value if FALSE.

Logical_test is any value or expression that can be evaluated to TRUE or FALSE.

Formula result = 6.56

Help on this function

Just position the cursor over the function name as it sits in the formula bar and go Alt-I-F and Excels extremely capable function wizard will pop all the inputs to that function (fully evaluated) and the output of that function. Where you have several functions in a formula and where you need to know how each component of the formula evaluates, this is the best and probably only way of debugging.

Excels little known functions

I always recommend that you have Excel's analysis toolpak installed (Tools/Addins) because a lot of Excels best functions reside here and the help forums on the web are filled with 'just look in the analysis toolpak' responses to queries. Some of the following are in the toolpak and some are not, but you should always have it installed.

EDATE is a great way to increment a date by a specified number of months. If you use Business Functions a lot you will probably prefer *DpM*, but EDATE is very capable at what it does. EOMONTH is a handy way to get the date of the end of the current month. One function which rather amusingly seems to have disappeared from Excels current documentation is DATEDIF which determines the time difference between two dates. BF users will use *DiffM* or *DiffY* in preference, but DATEDIF is useful if you don't have anything else. No-one knows why its slipped into recluse in the Excel help file - I think its because Excels developers have realised what a bag of worms dealing with date differences is, and would rather not go any further in that direction. At Business Functions of course, we embrace that same bag of creepy-crawlies with reckless abandon, which is why we offer dozens of date difference functions!

Some very powerful functions for dealing with ranges are INDIRECT, OFFSET and INDEX, which we won't go into here, but which do their stated jobs in rock-solid fashion.

Introduction To Business Functions

Up to now everything in this booklet has been about Excel in general and our individual view of it. Now lets come to Business Functions, BF came about because of the need for business models that were accurate, fast, flexible, understandable, transparent and small. This arose out of the various demands of the users of financial information:

- Management expect accuracy, speed and need flexibility.
- Analysts (the users) need models that are understandable, fast, flexible, transparent and accurate.

- Third parties like banks and advisors expect accuracy and transparency, and need a fairly small model.

As these pressures bear down on the standard version of Excel, things start to give. Model size is usually the first to give, as the demand for accuracy requires that short cuts and fudges just won't do, and masses of intermediate calculations on worksheets are required. I well remember how calculating a simple real estate rent that reviewed to market required several columns of intermediate calculations and still lacked flexibility. That special form of property rent, retail turnover rent, had Excel creaking. Even today, my understanding of the current state of the art is that the problem of modelling partial periods (eg amounts changing or starting part way through a year or quarter) is *not* really solved, and as a result, a large of number of analysts don't do it. That's a bit like your mortgage company rounding your total interest bill to nearest month - rather a large number!

Despite the magnitude of the problem, the basic calculations we are struggling with are basically repetitive:- rents that escalate, interest, spreading costs over time. We need to access standard formulae, or what might more generically be called logic, from a system that is highly optimised and fast, because these calculations happen thousands of times a second, every time someone changes an input cell. To get the speed you need a compiled language like C++, because you can subject code written in this type of language to specialised optimising compilers that can increase the speed of execution several times over. In a recent release we increased execution sped by nearly 40% just by changing the brand of compiler - that’s the kind of thing you can expect. Although you can write custom functions in Visual Basic for Applications, you cannot get the required speed on today’s computers.

So the only way to reliably package a set of logic is in a function, and the only way of getting it fast enough, is the use of the language C++. Hence the need for something like Business Functions, which has at its core a set of projections functions that do the forecasting work that presented such difficulty. Of all the spreadsheet packages out there, we are lucky and Microsoft are to be congratulated that Excel is the most open and accessible of the programs to work with, and as a result companies like Bloomberg have been writing mission-critical C++ add-ins for Excel for many years.

The design of Business Functions was mapped out in 2001 and it became apparent early on that the really difficult nut to crack was partial periods and how time is calculated, which can be summed up under the banner of *daycount*. The way loan interest, bond interest and rent is calculated is quite different in respect to how they deal with parts of a year, and a new methodology had to be created for use in the functions to cope with everything we could throw at it. The second issue to appear was that of accruals and cash. Treasury and the analysts would always want cash, so that interest and rent would only appear in a monthly budget every 3 months, whereas the accountant expected a monthly (or any other basis of) accrual, or in other words a continuous flow as opposed to a periodic discrete set of payments. Then there was a myriad of other desires that arose from people's wish list and often with their root in the calculation of time. At the same time many general utility functions were added, because you only really want to have one or two add-ins loaded into Excel or it gets confusing, and fitting them in to BF's documentation system and structure all made complete sense. One of the great strengths of the library is that calls itself an enormous amount of the time, so a bug-fix in an interest function probably fixes a rent function too.

The result of the last 4 and a half years development is a library which is robust and stable and capable of handling the complexities of modelling projections, which is at the heart of every cash flow analysis.

Starting Out: Modelling One-Off Payments

To start with the easiest example, let's consider modelling some discrete, one off payments that need to dropped into a cash flow that is organised by quarterly budget periods. There are no partial periods to worry about, because either a payment occurs in a budget period or it doesn't and it occurs in another one. We have two lists in two ranges on an Excel spreadsheet: *Payments* and *PaymentDates*. Let's see ho we would do this in an Excel formula. The obvious place to start is Excel's SUMIF function but there's a problem. We need to check if the *PaymentDate* is within (ie between the start and the end of the budget period) and that amounts to two criteria for the function, and SUMIF only takes one. So what we find ourselves having to do is a very advanced technique: using SUM, array entered with two boolean Control Arrays as follows:

```
    {=SUM(Payments*(E17<=PaymentDates)*(F17>PaymentDates))}
```
[1]

This is quite elegant but no-one is going to understand what you have done, particularly since it’s a rather unusual type of array formula. It is not a single array formula applied to whole timebase, which you might expect, instead it is an array formula for a single cell that you copy into each cell in the timebase. Come to think of it, I have difficulty understanding why it needs to be an array formula, but it just does. Gosh, this Excel business is quite complex!

Lets do it in BF. The formula we needed was just:

```
    =MkPmts(E17,F17,PaymentDates,Payments)
```

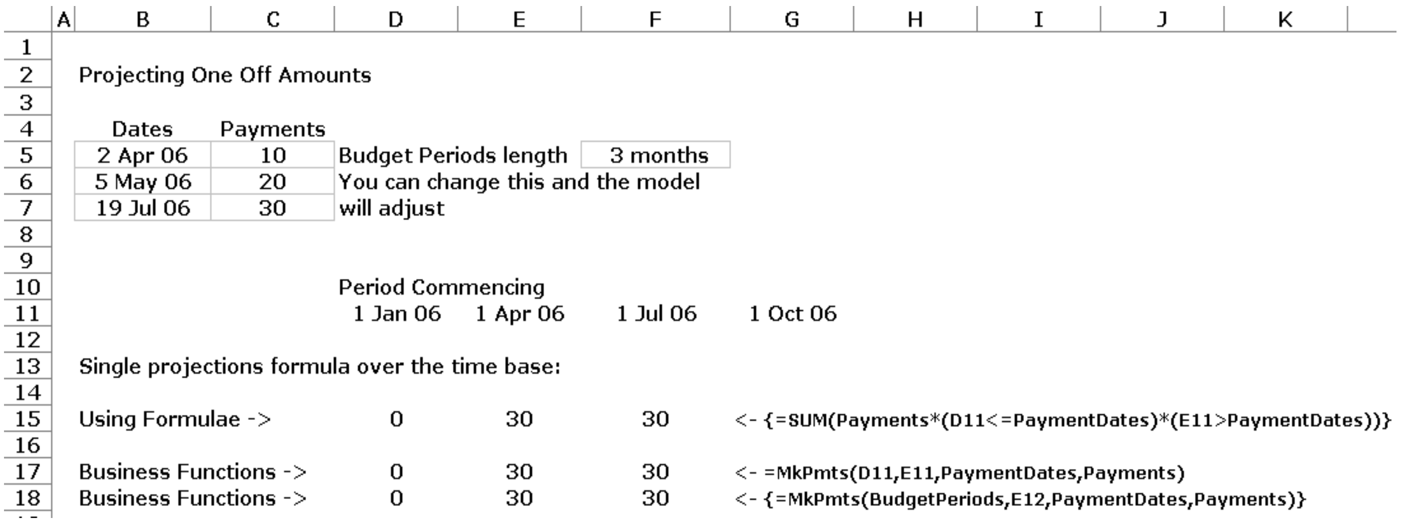
[1] If you don’t understand array formulae (and neither did I until surprisingly recently), skip ahead to the Array Formula section. Basically, they are formula that return an array not just a single number, and you edit them with a strange *Control-Shift-Enter* key sequence.

Distribute Payments on each PaymentDate in the timeperiod that starts at the date in cell E17 and finishes in the cell at F17

This is succinct and you can understand what it is - it is a 'Make Payments' function, and it's a normal formula, not an array formula. It couldn't really be simpler. If you want the array form of it it is just:

```
{=MkPmts(BudgetPeriods,F18,PaymentDates,Payments)}
```

Distribute Payments on each PaymentDate in the timeperiods that are defined by the range at BudgetPeriods



	A	B	C	D	E	F	G	H	I	J	K
1											
2		Projecting One Off Amounts									
3											
4		Dates	Payments								
5		2 Apr 06	10	Budget Periods length		3 months					
6		5 May 06	20	You can change this and the model							
7		19 Jul 06	30	will adjust							
8											
9											
10				Period Commencing							
11				1 Jan 06	1 Apr 06	1 Jul 06	1 Oct 06				
12											
13		Single projections formula over the time base:									
14											
15		Using Formulae ->		0	30	30	<- {=SUM(Payments*(D11<=PaymentDates)*(E11>PaymentDates))}				
16											
17		Business Functions ->		0	30	30	<- =MkPmts(D11,E11,PaymentDates,Payments)				
18		Business Functions ->		0	30	30	<- {=MkPmts(BudgetPeriods,E12,PaymentDates,Payments)}				

Using the Time and Base Variables

The previous example introduces some features of Business Functions. Nearly all projections functions start with a *Time* argument followed by a *Base* argument, which can be used in a variety of ways as follows:

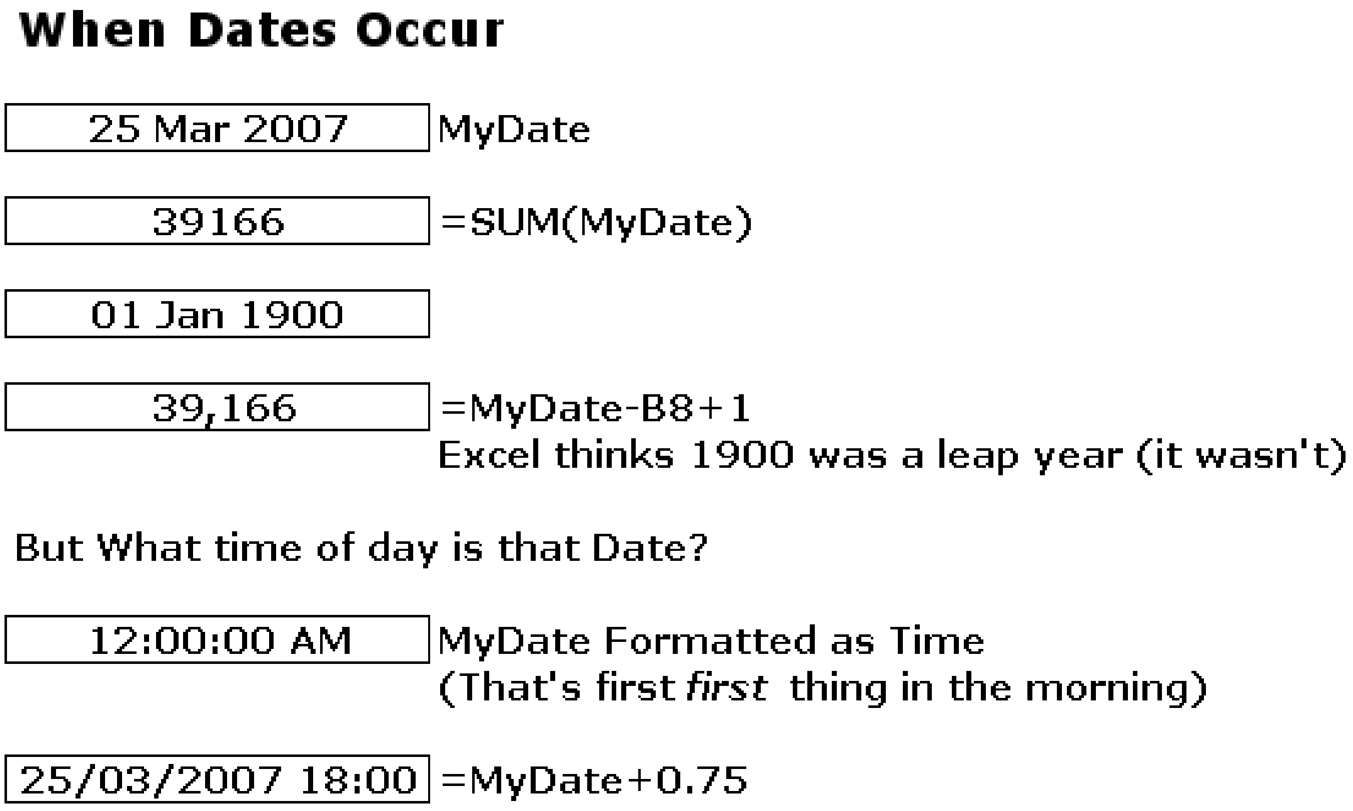
- You can use *Time* to be the start of the time period and *Base* to be the length of that time period in months. So *Time*=1/1/2006 and *Base* of 3 means the first calendar quarter of 2006, and all payments from 1st Jan to 31st March inclusive would get included in a *MkPmts* function call.

- You can use *Base*, as we did above, to be the *end* of the time period, so *Time*=1/1/2006 and *Base*=1/4/2006 also means the first quarter of 2006. Ah wait, I can hear you say, is the 1st of April really the end of the first quarter, or should it be 31st March? We shall go into this in the next section which deals with when dates actually occur.

- Finally, *all* functions that have *Time* and *Base* as their first two arguments can be used as array functions, and in fact they are called *Auto-Array* functions in the BF vernacular. This means that you can effectively use just one function call across the whole timebase. The function will return an array which will go across the whole timebase in one fell swoop, and your model will be a good deal faster. On the other hand, you need to understand a bit about array functions, in particular that you enter or edit them with *Ctrl-Shift-Enter*.

When Does a Date Occur?

This is terrific question, and one you are probably incredulous that we are asking. But it's worth reading this section because a great many errors occur, not just with BF but with Excel and life generally, because people misunderstand when critical times occur or are deemed to have occurred For example, take a property lease that starts on 1st Jan, does that mean you pay rent for 1st Jan? *Yes*, of course it does. But lets say a loan is repaid on 1st of Jan, is interest payable for the first? *No!* And let us say that you type a date into a cell in Excel. Excel translates that into a date and time for its own use. What time does that represent? Midnight? Midday?



When Dates Occur

```
25 Mar 2007          MyDate

39166                =SUM(MyDate)

01 Jan 1900

39,166               =MyDate-B8+1
                     Excel thinks 1900 was a leap year (it wasn't)

But What time of day is that Date?

12:00:00 AM          MyDate Formatted as Time
                     (That's first first thing in the morning)

25/03/2007 18:00     =MyDate+0.75
```

The answer to these questions and the rule to remember in all your spreadsheeting is that the date occurs at a tiny instant past midnight at the start of the day in question. You will see this if you reformat a date as a time, you will get 00:00 AM. And it explains why the aforementioned rent was payable on that day and interest was not. Interest is payable up until the day before repayment, because repayment is deemed to occur in that nanosecond after midnight on the day in question.

Another way of explaining this rule is that to say that by the date 'occurring' at a nanosecond past midnight, what ends up happening is that you include the first day of a time period, but exclude the last.

This works great for loans but there is something to watch out for in the case of real estate leases. We frequently say that a lease expires on a certain day, when what we mean is that it expires at the *end* of that day - ie the *start* of the next day. So for property leases an expiry of 25th March 2010 actually should be represented in Business Functions as 26th March 2010. If this seems too hard to remember, there is a *ProjMode* option which effectively does the adjustment for so that you can put in the unadjusted date of 25th March, but its probably better to remember and do the adjustment yourself. More about *ProjMode* later.

Moving on: Spreading an Amount Over Time

We have dealt with discrete items of expense or income that you want to drop into a budget period in a cash flow. Now lets look at spreading a total capital cost over time. This is a very common thing you want to do, and is used for things you might consider less often, like depreciation. Along the way we will come across some initially tough concepts of *DayCount*, *Periods*, *cash* and *accruals*.

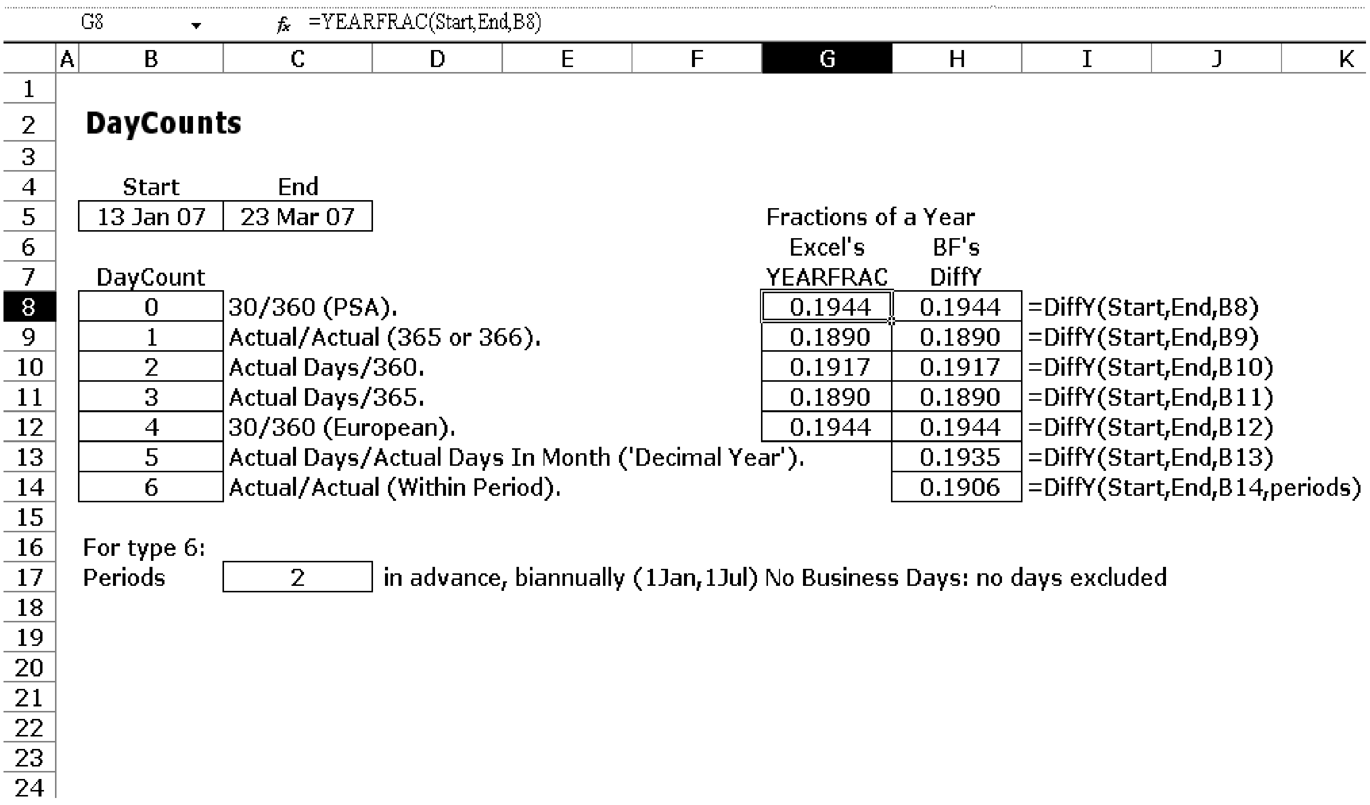
So lets spread £100 uniformly over a year from Feb 2006, with our budget in calendar quarters. We can deduce that the average rate of spend over the year is £100 per annum, so that for complete budget quarters we would expect to pay £25. But what about the first and last periods? They are *partial periods*[1], and what's worse, they are of different length depending on whether it's a leap year. A lot of people give up at this very early stage and round everything to the nearest quarter, so it would be 4 lots of £25 and it would be, basically, wrong. In order to understand how much to put in each period we need to understand *Daycount*, and its sister variable, *Periods* and *ProjMode*.

The Daycount, Periods and ProjMode Variables

These three variables are interlinked and deal with how time differences are measured (*DayCount* and *Periods*) and when amounts are to be paid (*Periods* and *ProjMode*).

[1] A partial period is just that: part of a time period, or budget period, over which a rate of something flows, and you are usually concerned with how much of a year or a period that partial period represents.

DayCount is familiar to in its basic form to lots of us. In essence, there is more than one way of determining the fractional part of a year or a period between two dates. If this seems strange, consider a property lease and compare it to interest on a loan. Interest on a loan is usually charged on an Actuals/365 basis, so a typical quarters interest might be 90, 91 or 92 days divided by 365. Rent, on the other hand is always the same in each complete quarter, so it's method of daycount is quite different, as we shall see later. Furthermore, things like Bonds have a different interest regime again, using such conventions as 30/360. In fact, Business Functions has 11 types of daycount (compared to Excel's 5).



G8 =YEARFRAC(Start,End,B8)

DayCounts

Start	End
13 Jan 07	23 Mar 07

Fractions of a Year

DayCount		Excel's YEARFRAC	BF's DiffY	
0	30/360 (PSA).	0.1944	0.1944	=DiffY(Start,End,B8)
1	Actual/Actual (365 or 366).	0.1890	0.1890	=DiffY(Start,End,B9)
2	Actual Days/360.	0.1917	0.1917	=DiffY(Start,End,B10)
3	Actual Days/365.	0.1890	0.1890	=DiffY(Start,End,B11)
4	30/360 (European).	0.1944	0.1944	=DiffY(Start,End,B12)
5	Actual Days/Actual Days In Month ('Decimal Year').		0.1935	=DiffY(Start,End,B13)
6	Actual/Actual (Within Period).		0.1906	=DiffY(Start,End,B14,periods)

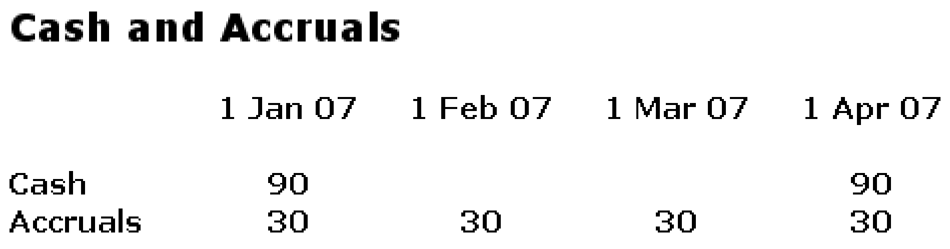
For type 6:
Periods 2 in advance, biannually (1Jan,1Jul) No Business Days: no days excluded

There is one particular type of daycount which is absolutely vital and yet Excel doesn't do it. That daycount type is known as *Actual/Actual in Period*, which is to say that the number of days between two dates is divided by the total number of days in the period, and then, if necessary, divided by the number of periods in a year.

If you are feeling the mists start to rise before you, hang on! Because there is one more nuance of daycount we need to cover before we show how to apply it. That particular peculiarity is that frequently more than one daycount type is used to calculate something and the perfect example is rent. For periods that are complete, the rent is the same in each quarter. Without obviously saying so, this implies Actuals/Actuals in Period daycount for complete periods. partial periods at the start and end of a lease are more a matter of the idiosyncrasies in the lease. Sometimes a partial period is calculated as Actual/365 and sometimes it is Actual/Actual in period (i.e. x/(91 or 92)/4).

Business Functions has all the standard Excel daycount options but you may notice that the Actual/Actual in period daycount option requires that you specify *when* payments occur. In property this can mean the difference between calendar quarters and English Quarter Days. You *need* to specify these to calculate how much rent is due. The second thing we need to remember is how partial periods are specified. All this is embodied in the two variables *DayCount* and *Periods*, usually located at the end of an argument list in a projections function. Periods are specified by indicating the month and the day in *mm.dd* format, so quarterly periods are 1.01,4,01,7,01, and 10,01[1], although these can also be specified by using the preset value of 4 (payments per year). The two daycounts (that for complete periods and that for partial periods) are specified on either side of the decimal point in the *DayCount* variable. So 6.03, which is a typical rental daycount, means that complete rental periods are Actuals/Actuals in Period (and its always a complete period) and partial periods are Actuals/365. Another common one is 6 or 6.06. meaning Actuals/Actual in period for the partial periods. For loans, daycount is usually type 3 (Actuals/365) and for bonds it is usually 6.07 (Actual/Actual in period for complete periods, 30/360 for the partial periods. Got it? You need the two variables *DayCount* and *Periods* to completely specify what it is that you want, but between them these two variables can do *any* type of daycount type you need.

Finally we come to *cash* or *accruals*, one of the neatest features in Business Functions and something you could never do in a standard Excel formula. Again, if we take rent as the example, if you are doing a monthly budget on a *cash* basis, you want the rent to pop in every 3 months, and you want it to be zero in the 2 months between each rental payment date. On the other hand, many management accountants, and financial accountants for that matter, would prefer to budget on an accruals basis, which is to say that the rent is evenly spread throughout every month of the year, even if it only actually gets paid on quarter days. Both methods of projection require the same inputs, so you might hope that there would be a single function input that toggles between the two methods of projection. There is and its called *ProjMode*, which is zero (the default) for accruals and 1 for cash. In order to project on cash basis, you need to know *when* the rent or whatever is paid. That information we already have in the *Periods* variable.



Cash and Accruals

	1 Jan 07	1 Feb 07	1 Mar 07	1 Apr 07
Cash	90			90
Accruals	30	30	30	30

[1] English Quarted days would be 13. Monthky in arrears would be -12, etc. See help file for details.

The only remaining thing we need is to specify in advance and in arrears, and this point your brain might slip into overload, mine certainly did when I was doing all this. Rent and most costs are paid in advance whilst interest is always in arrears, and we said you need to remember that interest is always paid until the end of the day before the loan is repaid and rent starts on the day of the lease. So, given the information in *DayCount*, *Periods* and *ProjMode*, we just need a way to specify whether we are talking about something that is in advance or in arrears. The place that belongs is in the *Periods* variable, and the way it works is that positive periods imply payments in advance and negative periods imply payment in arrears. So *Periods*=4 is quarterly in advance and -4 is quarterly in arrear. For UK property, *Periods*=13 is rent paid in advance.

So that is it, *DayCount*, *Periods* and *ProjMode* are the three variables to control how you deal with time, payment frequency and cash/accruals mode. There are some fantastic complexities I could go into here, such as how to incorporate business days in to a *Periods* variable so that interest is paid only on a business day, or a type of *ProjMode* that allows you to make payments on a cash basis a certain number of days after the end of a payment period (ie a time lag), but all the information should you want to do this is in the Business Functions help file! For now, I would urge you to understand *DayCount*, *Periods* and *ProjMode* since it is the source of a number of misunderstandings. The main thing to remember is, that whatever someone may tell you: time is complicated!

Spreading An Amount Over Time Part 2

In order to spread a single capital amount over time we needed to understand how daycount works and , in turn, how partial periods should be dealt with. So, back to spreading the £100 between 1st Feb on two successive years. If you were doing this in plain Excel without Business Functions you would have to make two compromises:

- You couldn't do it on budget periods, you would have to have as your timebase the exact payment dates, and then you would have to consolidate quarters into years or whatever later.
- You would have to be a lot more rough and ready with dealing with daycount over partial periods.

If you were to accept these two limitations, the following formulae would work quite well for monthly accuracy and Actuals/365 respectively:

```
=DATEDIF(MAX(E$17,FromDate),MIN(F$17,ToDate),"m")*amount/
DATEDIF(FromDate,ToDate,"m")
```

```
=(MIN(F$17,ToDate)-MAX(E$17,FromDate))*amount/(ToDate-
FromDate)
```

Note the use of Excel's rare DATEDIF function for the one that just rounds everything to the nearest month.

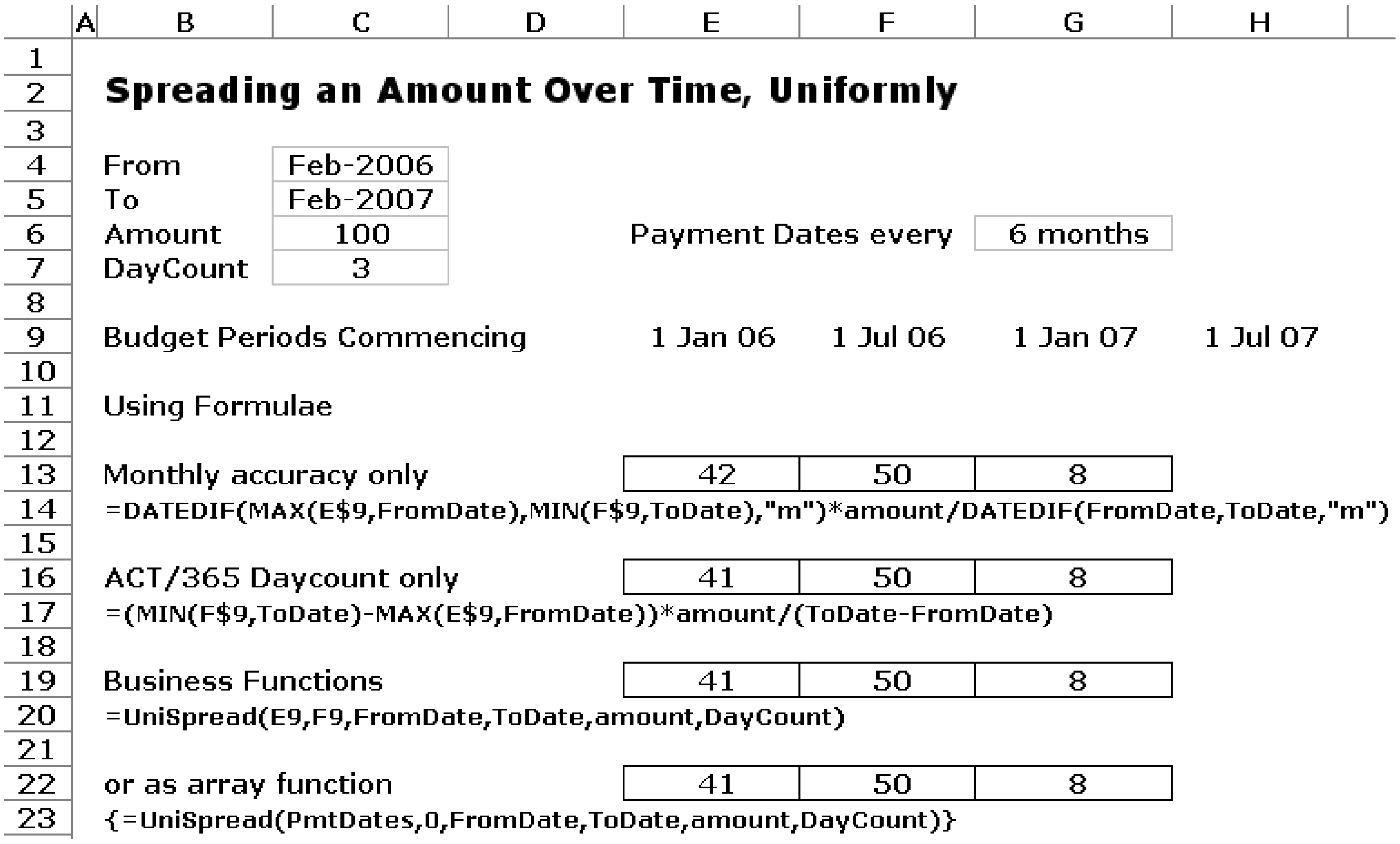
The Business Functions way of handling this would be:

```
=UniSpread(E17,F17,FromDate,ToDate,amount,DayCount)
```

or the array function form:

```
=UniSpread(PmtDates,0,FromDate,ToDate,amount,DayCount)
```

Notice that we didn't even specify *Periods* and *ProjMode*, because when our timebase is the payment dates, as opposed to the budget periods, accruals and cash are the same thing.



	A	B	C	D	E	F	G	H
1								
2		**Spreading an Amount Over Time, Uniformly**						
3								
4		From	Feb-2006					
5		To	Feb-2007					
6		Amount	100		Payment Dates every		6 months	
7		DayCount	3					
8								
9		Budget Periods Commencing			1 Jan 06	1 Jul 06	1 Jan 07	1 Jul 07
10								
11		Using Formulae						
12								
13		Monthly accuracy only			42	50	8	
14		=DATEDIF(MAX(E$9,FromDate),MIN(F$9,ToDate),"m")*amount/DATEDIF(FromDate,ToDate,"m")						
15								
16		ACT/365 Daycount only			41	50	8	
17		=(MIN(F$9,ToDate)-MAX(E$9,FromDate))*amount/(ToDate-FromDate)						
18								
19		Business Functions			41	50	8	
20		=UniSpread(E9,F9,FromDate,ToDate,amount,DayCount)						
21								
22		or as array function			41	50	8	
23		{=UniSpread(PmtDates,0,FromDate,ToDate,amount,DayCount)}						

What Business Functions would enable you do in this case is remove the restriction that the timebase is the payment dates and instead have the timebase as budget period dates, and allow you to project on an *annual* basis for amounts which were paid *quarterly*:

```
=UniSpread(E17,F17,FromDate,ToDate,amount,DayCount,4,1)
```

or for accruals:

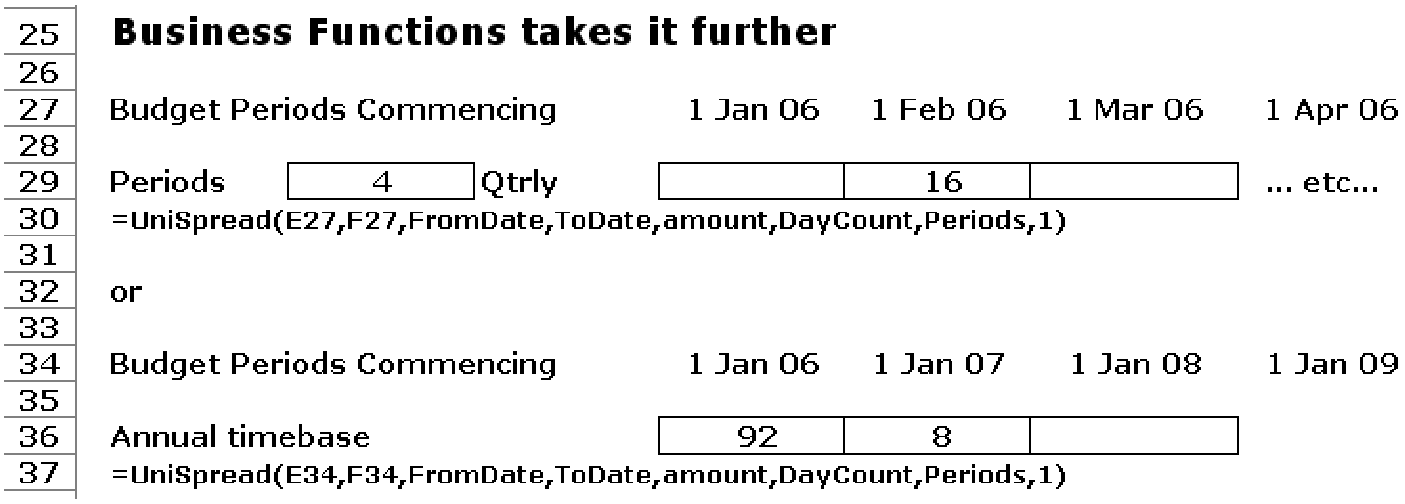
```
=UniSpread(E17,F17,FromDate,ToDate,amount,DayCount,4,0)
```

Let's repeat that: you can project on an annual basis amounts which are paid quarterly. Not only that, you could project on a quarterly basis amounts that are paid annually.

This is powerful stuff, if you can grasp what's going on here. We've managed to separate the definition of the timebase over which you are projecting (the budget periods) from the definition of *what* you are projecting (a capital amount spread uniformly between two dates, with payments dates on the first day of each quarter). This is crucial to moving to the next level of cashflow modelling: everything that relates to the object being calculated is in the *UniSpread* function. What is going on with the timebase is quite different, the function can cope with *any* timebase. I will offer a couple of quotes from respected authors:

> *"Quarterly to Annual. This is one of the thorniest of all modelling issues.. The usual solutions tend to be a bit messy" Jonathan Swan, Operis plc, "Practical Cashflow Modelling"*
>
> *".... to keep things simple we always work to the nearest quarter day" Scott Fawcett, "Designing Flexible Cashflows"*

Where others have to do two versions of a cashflow or have intermediate calculations, *we* can just change a single variable and our whole cashflow changes, from cash to accruals, from quarterly to annual. This is exactly what Business Functions users are used to doing - its just not a problem!



```
25  Business Functions takes it further
26
27  Budget Periods Commencing          1 Jan 06   1 Feb 06   1 Mar 06   1 Apr 06
28
29  Periods   [    4    ] Qtrly    [        ][   16   ][          ]  ... etc...
30  =UniSpread(E27,F27,FromDate,ToDate,amount,DayCount,Periods,1)
31
32  or
33
34  Budget Periods Commencing          1 Jan 06   1 Jan 07   1 Jan 08   1 Jan 09
35
36  Annual timebase                  [   92   ][    8   ][          ]
37  =UniSpread(E34,F34,FromDate,ToDate,amount,DayCount,Periods,1)
```

Other ways of Spreading Amounts Over Time - S Curve etc

We have covered the easiest was of spreading an amount over time. Business Functions has several other functions that spread an amount over time in a more sophisticated way, but the general principle of how you use the function is that same. Refer to the help file for further details, but here are some of the main ones:

```
SCurve ( Time, Base, Start, Finish, Total, [Skew],
[Peakness], [DayCount], [Periods], [ProjMode] )

SCumCurve ( Time, Base, Start, Finish, Total, CumTime,
CumAmt, [InterpType], [Tension], [Bias], [DayCount],
[Periods], [ProjMode] )

FStepSpread ( Time, Base, Total, FromDates, AnnualRates,
[DayCount], [Periods], [ProjMode] )
```

Forecasting an Operating Cost

We have done discrete payments, and spreading an amount over time. The next thing to master is forecasting a recurring or continuous cost. It could be salaries, rates, even interest. Whatever it is, its going to be a rate of cost that accrues throughout the year, perhaps paid periodically. It's a flat amount that just has a start date an end date and an annual rate. We shall choose £900 per year starting in Feb 06 and finishing in Oct 06. The conventional way to do this would involve the same compromises as before:

- The timebase has to be payment dates, not budget periods.
- We shall have to be very crude with partial periods.

The formula I came up for this was either:

```
=DATEDIF(MAX(E$22,FromDate),MAX(E$22,MIN(F$22,RateTo)),"m
")*RatePA/12
```

if monthly accuracy would suffice (it usually doesn't), and

```
=MAX(MIN(F$22,RateTo)-MAX(E$22,FromDate),0)/365*RatePA
```

if Actual/365 daycount was adequate for your needs.

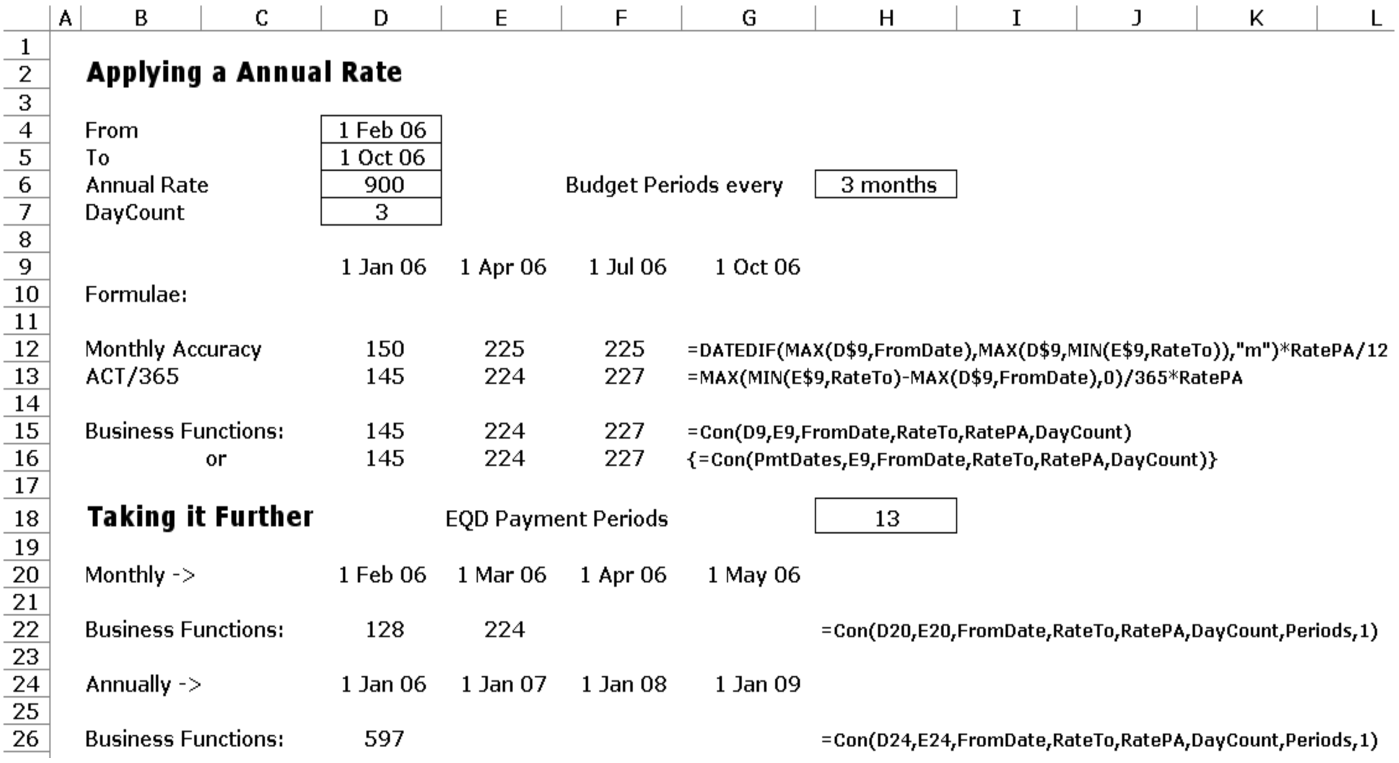
The limitations mentioned are severe, however. The Business Function has no limitations on the timebase and is simply:

```
=Con(E22,F22,FromDate,RateTo,RatePA,D20)
```

or its array form:

```
=Con(PmtDates,0,FromDate,RateTo,RatePA,D20)
```

where *Con* is short for 'constant rate'. Again, we can go further with Business Functions than conventional formulae, because we can adapt to any timebase or payment frequency.



	A	B	C	D	E	F	G	H	I	J	K	L
1												
2		**Applying a Annual Rate**										
3												
4		From		1 Feb 06								
5		To		1 Oct 06								
6		Annual Rate		900		Budget Periods every		3 months				
7		DayCount		3								
8												
9				1 Jan 06	1 Apr 06	1 Jul 06	1 Oct 06					
10		Formulae:										
11												
12		Monthly Accuracy		150	225	225	=DATEDIF(MAX(D$9,FromDate),MAX(D$9,MIN(E$9,RateTo)),"m")*RatePA/12					
13		ACT/365		145	224	227	=MAX(MIN(E$9,RateTo)-MAX(D$9,FromDate),0)/365*RatePA					
14												
15		Business Functions:		145	224	227	=Con(D9,E9,FromDate,RateTo,RatePA,DayCount)					
16			or	145	224	227	{=Con(PmtDates,E9,FromDate,RateTo,RatePA,DayCount)}					
17												
18		**Taking it Further**			EQD Payment Periods			13				
19												
20		Monthly ->		1 Feb 06	1 Mar 06	1 Apr 06	1 May 06					
21												
22		Business Functions:		128	224			=Con(D20,E20,FromDate,RateTo,RatePA,DayCount,Periods,1)				
23												
24		Annually ->		1 Jan 06	1 Jan 07	1 Jan 08	1 Jan 09					
25												
26		Business Functions:		597				=Con(D24,E24,FromDate,RateTo,RatePA,DayCount,Periods,1)				

Recap - Where We Have Got To

We have come a long way in this little booklet and much of this may be new to you, so it's time to recall where we have got and why. We started out looking for ways to package standard calculations and concluded that the function is the best way to extend Excel's financial modelling capabilities. The idea of a projections function came from the need for *timebase-consistent formulas*, because we wanted to build a true model that could respond to different timebases and start dates. We noted that this was going to be hard, but in our first 'Make Payments' example, we were able to use Excel formulas as well as a Business Function to achieve the same result. This was because we were just looking at popping discrete payments into a set of budget periods.

When it came to spreading an amount over time we had our first encounter with the beast we call *daycount*, and saw how a time difference is very much in the eye of the beholder. Although we did a noble effort in native Excel formulas, we had to fudge the daycount issue and assume that the timebase was in fact the payment dates, and we just couldn't cope with having the timebase as budget periods.

We found that the Business Function *UniSpread* not only exhaustively dealt with the daycount problem, it could also cope with varying budget periods, varying payment dates and could even do cash and accruals. In short, the cashflow object managed by the function had, at last, become truly independent of the timebase and the era of true modelling was upon us.

With the bit between our teeth, we quickly sorted out how an constant rate operating cost could be done with Business Function *Con*. Just to show that

Excel's native formulas are not completely dead and buried, we did the same as we did in the *UniSpread* example, whilst noting that we were suffering the same restrictions.

We have completed the 3 key types of projection function: discrete payments, the spreading an amount over time, and the application of a constant rate over time. From hereon we are going to cover a few slightly more challenging examples in a similar fashion, and we are going to start with a type of projection that is highly relevant to a large body of Business Functions users, rent. But before we do that, we are going to review the date functions in Excel and Business Functions, since this is also useful revision.

Date Functions

At the core of a Business Functions projection function is date arithmetic, as it works out how to apply a rate, rent or amount. The raw date functions are often useful by themselves. But, I hear you say, doesn't Excel have some rather good date functions? Excel does indeed have a small number of date functions, and we shall look at them as we discuss some common tasks.

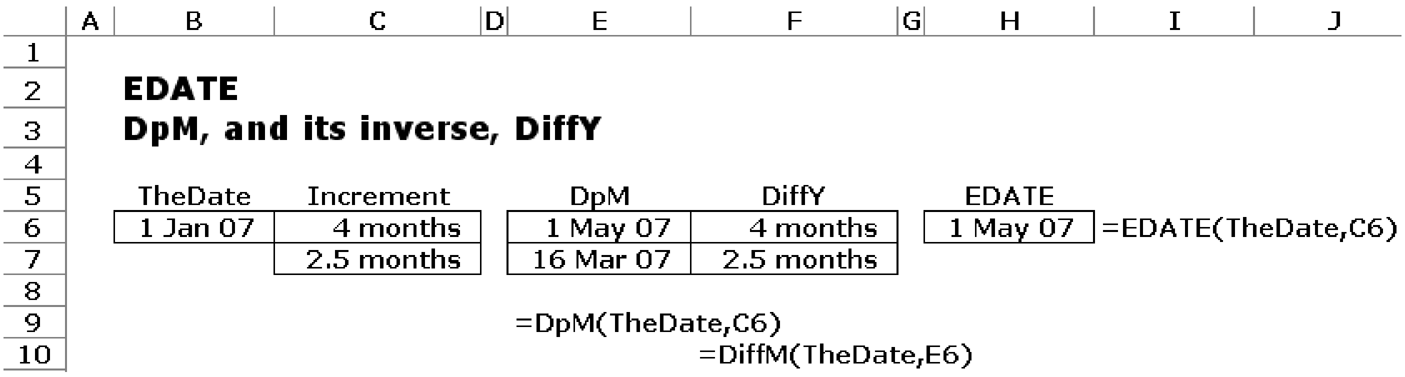
One extremely common task is determining the time difference between two dates, As has been mentioned elsewhere, there are different algorithms for determining time difference between two dates that fall under the broad area of daycount - it is not just the difference in days. Excel has two functions in this area: DATEDIF and YEARFRAC. DATEDIF is really a very useful function but you will find that it is largely undocumented in the Excel help files. No matter though, because there is lots of information about it on the internet. DATEDIF works out the difference between two dates in terms of whole numbers of the unit that you specify - days, months or years. So its obvious deficiency is that it can't cope with a date difference that involves partial months or years, because it only deals in whole numbers it kind of sidesteps neatly the issue of daycount . In truth, its only working out the number of anniversaries (or 'monthly anniversaries' if that makes sense) between two dates.

YEARFRAC does cope with Excel's 6 daycount types and returns the fraction of the year. The only problem is that it does not do Actual/Actual in Period - in other words it doesn't know or care about, say, English Quarter Days, so working out the fractional years of rent between 2 dates won't work properly. It's OK for small partial periods of less that a payment period, but that is only part of what you normally would normally want.

The closest Business Functions counterparts to DATEDIF and YEARFRAC are *DiffY*, *DiffM* and *DiffD*. These work out the fractional date difference in years, months and days respectively. They accept (optionally, its not mandatory) a *Daycount* and a *Periods* variable. This means that if you want to work out the number of quarters rent between 2 dates you can do it exactly as your property manager would, counting the whole periods and then choosing Act/365 or Act/Act in Period for the partial periods, which leads you to a daycount of 6.03 or 6.06[1]. You can directly specify any payments dates you like for *Periods* (eg 13=English Quarter Days). The Business Functions date difference functions work the same way as any date calculation in a projections function - it all uses the same code.

Another common task is incrementing a date, and here Business Functions extends Excel quite a bit. The basic Excel way to do this is to use EDATE, which increments a date a whole number of months. Again, because the Excel function deals with only integer months and works in a kind of 'anniversary' way, it doesn't need to worry about daycount - neat. So for creating a timebase you can go EDATE(ThePreviousDate,NumberOfMonths), and if you set out 1 years worth of payment dates (eg English Quarter Days) you can just use EDATE to project these forward by a year.

Business Functions nearest equivalent to EDATE is *DpY*, *DpM*, and *DpD*. These functions *do* cope with fractions of a month and therefore can optionally make use of *Daycount* and *Periods* variables. In fact, the Business Functions *guarantee* that too the extent possible, they will do the opposite of *DiffY*, *DiffM* and *DiffD*. This is sometimes more complex than it seems because some daycount types are not obviously reversible, so Business Functions will scan the vicinity around the answer to provide the best answer consistent with the daycount and periods supplied.



	A	B	C	D	E	F	G	H	I	J
1										
2		**EDATE**								
3		**DpM, and its inverse, DiffY**								
4										
5		TheDate	Increment		DpM	DiffY		EDATE		
6		1 Jan 07	4 months		1 May 07	4 months		1 May 07	=EDATE(TheDate,C6)	
7			2.5 months		16 Mar 07	2.5 months				
8										
9					=DpM(TheDate,C6)					
10						=DiffM(TheDate,E6)				

Business Functions has a great many date functions and one particular variant of a date increment function that is very useful is *NextDateSeq*, or 'return the next date in a sequence'. You can simple go *NextDateSeq*(ThePreviousDate,13) to create a timebase of English Quarter Dates. Not only that, but if you want that timebase to include one or two specific dates, like a rent start of expiry, you could these dates in a range and supply that as a third argument to the function.

[1] Refer to help file to see how we are yusing compound daycount here to specify the daycount for entire periods and partial periods.

One very simple Business Function is *Occurs*, which simply tells if a particular payment date occurs in a time period.

Another very useful Business Function is *MostImminent* which turns up in a few real estate applications, when you want to find the most closest date coming up after a certain. There are several more date functions of note, particularly for property, and they are constantly getting added to, but you are probably better off exploring the help file from here.

Forecasting Rent

Forecasting rent turns out to remarkably difficult in native Excel, even if, on the face of it, it appears easy. Rental Growth, stepped rents, expiries, relets and partial periods each add their own, sometimes intractable, complications. One book that is an interesting read as regards the native Excel way of doing things is Scott Fawcett's "Designing Flexible Cash Flows", published through the Estates Gazette, and it scopes out the main problems that you encounter.

The type of rent we are going to model here is a stepped rent that after one or more steps reviews to market, and after a void it relets. Simple? Let us see!

True to our past form in this booklet, we are going to do it the native Excel way first, but we are going to have to make some assumptions that are really rather a pain, but never mind:

- We are not going to deal with partial periods in native Excel.
- The timebase has to be the rental payment dates.

We will need some helper columns, Firstly, because we have allowed for multiple growth rates through a tidy 'growth from' mechanism, we will need to break that out into the *growth at each rental date*. We use a lookup for that, being careful to trim out it down for blank entries in the growth table:

```
=LOOKUP(B33,OFFSET(GrowthDates,0,0,1,COUNTA(GrowthRates))
,OFFSET(GrowthRates,0,0,1,COUNTA(GrowthRates)))
```

Then, we need to generate the *market* rent at each rental payment date, for which we use a couple of switches and a date function to apply growth.

```
=IF(D32=0,IF(B33<=FirstGrowthDate,MktRent,MktRent*(1+C33)
^(DATEDIF(FirstGrowthDate,B33,"m")/12)),D32*(1+C33)^(1/Pm
tsPerYr))
```

If this is starting to look complicated, it *is*, but do not worry, we will come to the Business Functions way soon. For now, let us see what you are missing out on. The next thing is to do the stepped rent section, which requires a lookup:

```
=LOOKUP(TRUE,B33<Reviews,Rents)/4*(B33>=RentStartDate)*(B
33<MAX(Reviews))
```

Then the market rent section (after the stepped rents have finished), uses Excel's MOD function to determine if a particular payment date is a review date. We are using DATEDIF a lot in this logic and here you see another limitation. Not only do we implicitly round to nearest quarter date, we also round other intermediate calculations to the nearest month. You might be thinking at this point that there is an easier way to do this. We would welcome you to demonstrate this method: apart from specially designed function like a Business Function, we just don't see a better way. Anyway, here’s the market rent section:

```
=IF(B33>=MAX(Reviews),IF(MOD(DATEDIF(MAX(Reviews),B33,"m"
),RevMos)=0,D33/PmtsPerYr,F32),0)*(B33<ExpOrBrk)
```

Finally we do the relet (just one relet) by using a similar formula to the market rent section:

```
=IF(B33>=EDATE(ExpOrBrk,ReletVoid+ReletRentFree),IF(MOD(D
ATEDIF(EDATE(ExpOrBrk,ReletVoid),B33,"m"),RevMos)=0,D33/P
mtsPerYr,IF(DATEDIF(EDATE(ExpOrBrk,ReletVoid+ReletRentFre
e),B33,"m")=0,D33/PmtsPerYr,G32)),0)
```

We add up the stepped rent, market rent and relet rent sections and we have a rent projection that has as a plus point that it uses timebase-consistent formulae, but unfortunately it is not really accurate and does not feel at all robust. At this stage I thought it would be sensible to see how the aforementioned book solved a similar, slightly simpler rent, as a sanity check and to see that we have not made a mountain out of a molehill. I was therefore relieved to find this rental formula:

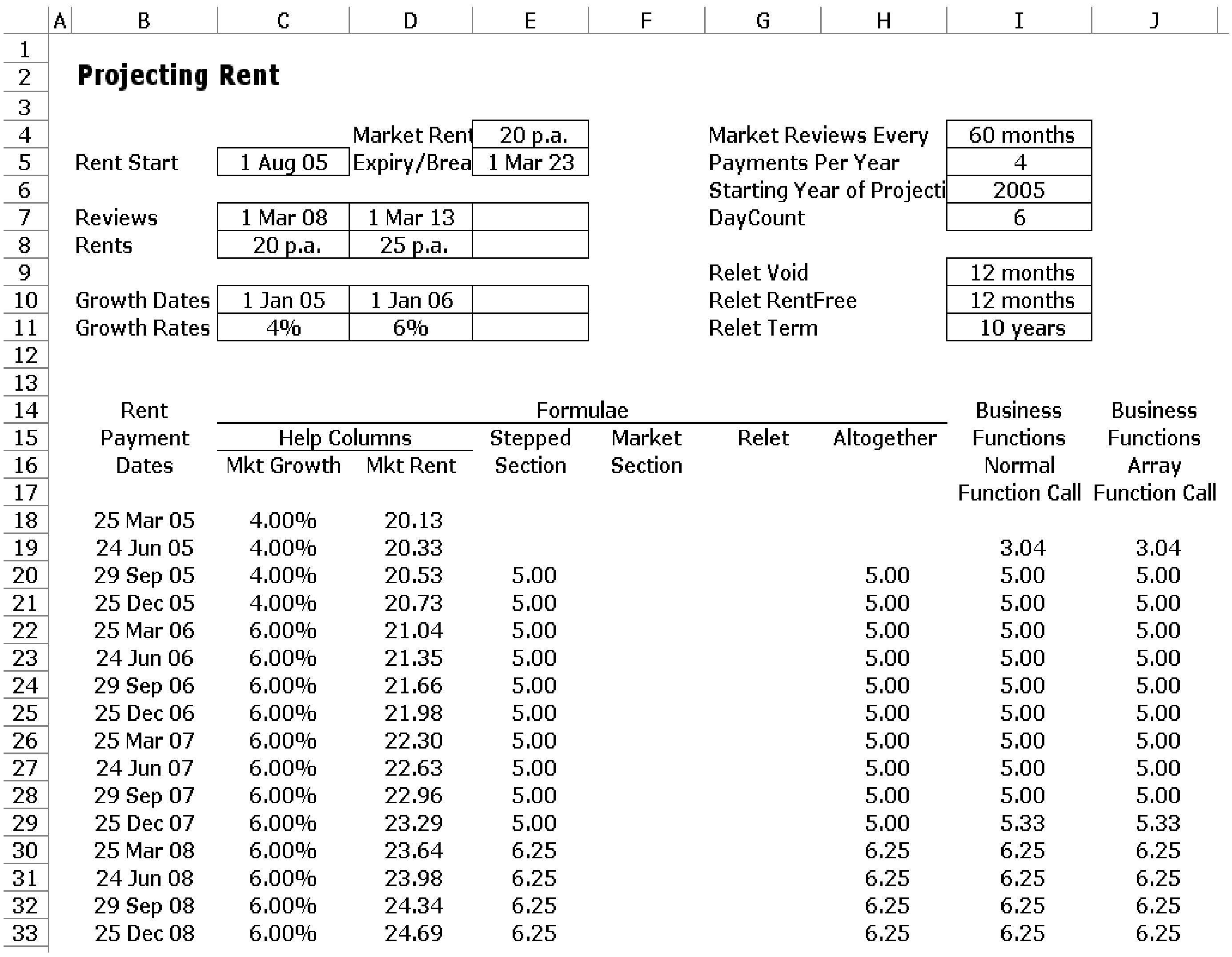
```
IF(M$3=I$4,$D4*(1+$J4)^((M$2-
1)/4)/4,IF(M$3=$G4,0,IF(AND(OR(M$3=$E4,M$3=$F4),$D4*(1+$J
4)^((M$2-1)/4)/4>L4,$D4*(1-J$4)^((M$2-1)/4,L4)))
```

At this point you are probably snowed, I don't blame you - I have a lot of trouble understanding long formulae with embedded IF statements Excel's formulae quickly become impossibly complex and that means they will be hard to understand and to maintain. That wouldn't be so bad but as we have found out they are not even accurate!

The answer, of course, is a function. But before we come to using a Business Function let me tell you that you can achieve notable progress with a VBA function. Its not given here, but it is in the companion workbooks to this booklet that you can download from the internet. VBA can package up a lot of the intermediate calculations so that the function call you need is something like:

```
=a_steprentmkt(PmtDates,RentStartDate,ExpOrBrk,MktRent,Re
views,Rents,GrowthDates,GrowthRates,RevMos,ReletVoid,Rele
tRentFree,ReletTerm,daycount,0)
```

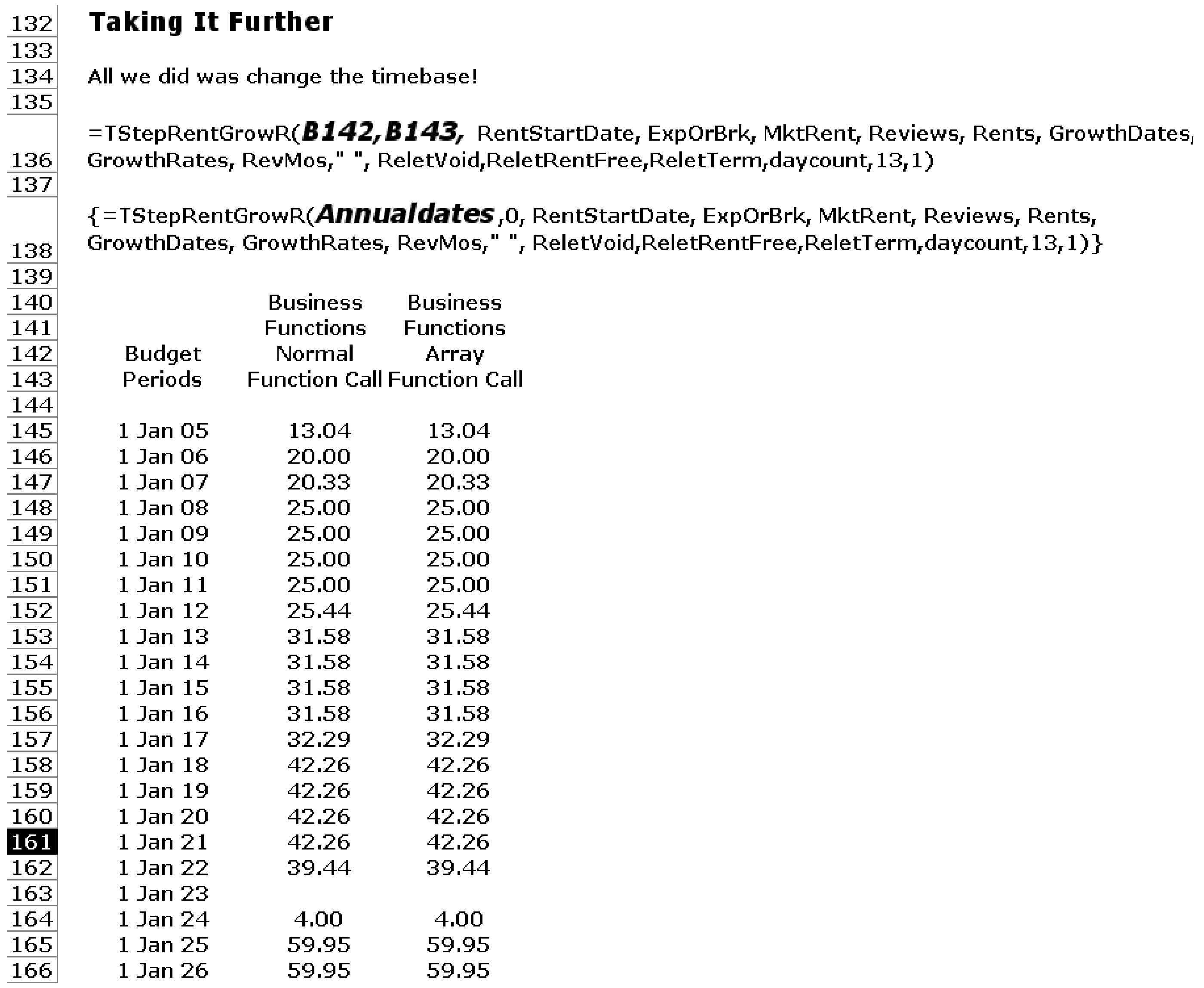
Because it's VBA however, we are limited by performance constraints, and this case, whilst we managed to model partial periods, we had to retain the stipulation that the timebase is the same as the rental payment dates. We also made it an array function, which you may find awkward to use, but was necessary in the name of performance.



	A	B	C	D	E	F	G	H	I	J
1										
2		**Projecting Rent**								
3										
4				Market Rent	20 p.a.		Market Reviews Every		60 months	
5		Rent Start	1 Aug 05	Expiry/Brea	1 Mar 23		Payments Per Year		4	
6							Starting Year of Projecti		2005	
7		Reviews	1 Mar 08	1 Mar 13			DayCount		6	
8		Rents	20 p.a.	25 p.a.						
9							Relet Void		12 months	
10		Growth Dates	1 Jan 05	1 Jan 06			Relet RentFree		12 months	
11		Growth Rates	4%	6%			Relet Term		10 years	
12										
13										
14		Rent			Formulae				Business	Business
15		Payment	Help Columns		Stepped	Market	Relet	Altogether	Functions	Functions
16		Dates	Mkt Growth	Mkt Rent	Section	Section			Normal	Array
17									Function Call	Function Call
18		25 Mar 05	4.00%	20.13						
19		24 Jun 05	4.00%	20.33					3.04	3.04
20		29 Sep 05	4.00%	20.53	5.00			5.00	5.00	5.00
21		25 Dec 05	4.00%	20.73	5.00			5.00	5.00	5.00
22		25 Mar 06	6.00%	21.04	5.00			5.00	5.00	5.00
23		24 Jun 06	6.00%	21.35	5.00			5.00	5.00	5.00
24		29 Sep 06	6.00%	21.66	5.00			5.00	5.00	5.00
25		25 Dec 06	6.00%	21.98	5.00			5.00	5.00	5.00
26		25 Mar 07	6.00%	22.30	5.00			5.00	5.00	5.00
27		24 Jun 07	6.00%	22.63	5.00			5.00	5.00	5.00
28		29 Sep 07	6.00%	22.96	5.00			5.00	5.00	5.00
29		25 Dec 07	6.00%	23.29	5.00			5.00	5.33	5.33
30		25 Mar 08	6.00%	23.64	6.25			6.25	6.25	6.25
31		24 Jun 08	6.00%	23.98	6.25			6.25	6.25	6.25
32		29 Sep 08	6.00%	24.34	6.25			6.25	6.25	6.25
33		25 Dec 08	6.00%	24.69	6.25			6.25	6.25	6.25

Business Functions treats rent as any other projections function, which means it comes as standard with the full daycount arithmetic, which comprehensively deals with partial periods but it also does what we were saying before is so useful: it works *on any timebase*, treating the dates as *budget periods*. In other words, once again the *financial object* you are modelling and the *timebase* are *independent of each other*. So if you want have the timebase as the rental payment dates, fine - but you can also have it as years, in which the function will correctly aggregate the rental payments that occur in that year, or by month, in which case the function will pop the rental payments into the appropriate month. Furthermore, you can run the whole thing on either a cash or accruals basis, so that the numbers are correct for both property management and the accountant. The Business Function also copes with upward only or upwards/downwards review, and discount to market at review. The Business Function call looks something like this:

```
=StepRentMkt(B33,B34,RentStartDate,ExpOrBrk,MktRent,Revie
ws,Rents,GrowthDates,GrowthRates,RevMos,ReletVoid,ReletRe
ntFree,ReletTerm,PmtsPerYr,daycount,0)
```



Taking It Further

All we did was change the timebase!

=TStepRentGrowR(***B142,B143,*** RentStartDate, ExpOrBrk, MktRent, Reviews, Rents, GrowthDates, GrowthRates, RevMos," ", ReletVoid,ReletRentFree,ReletTerm,daycount,13,1)

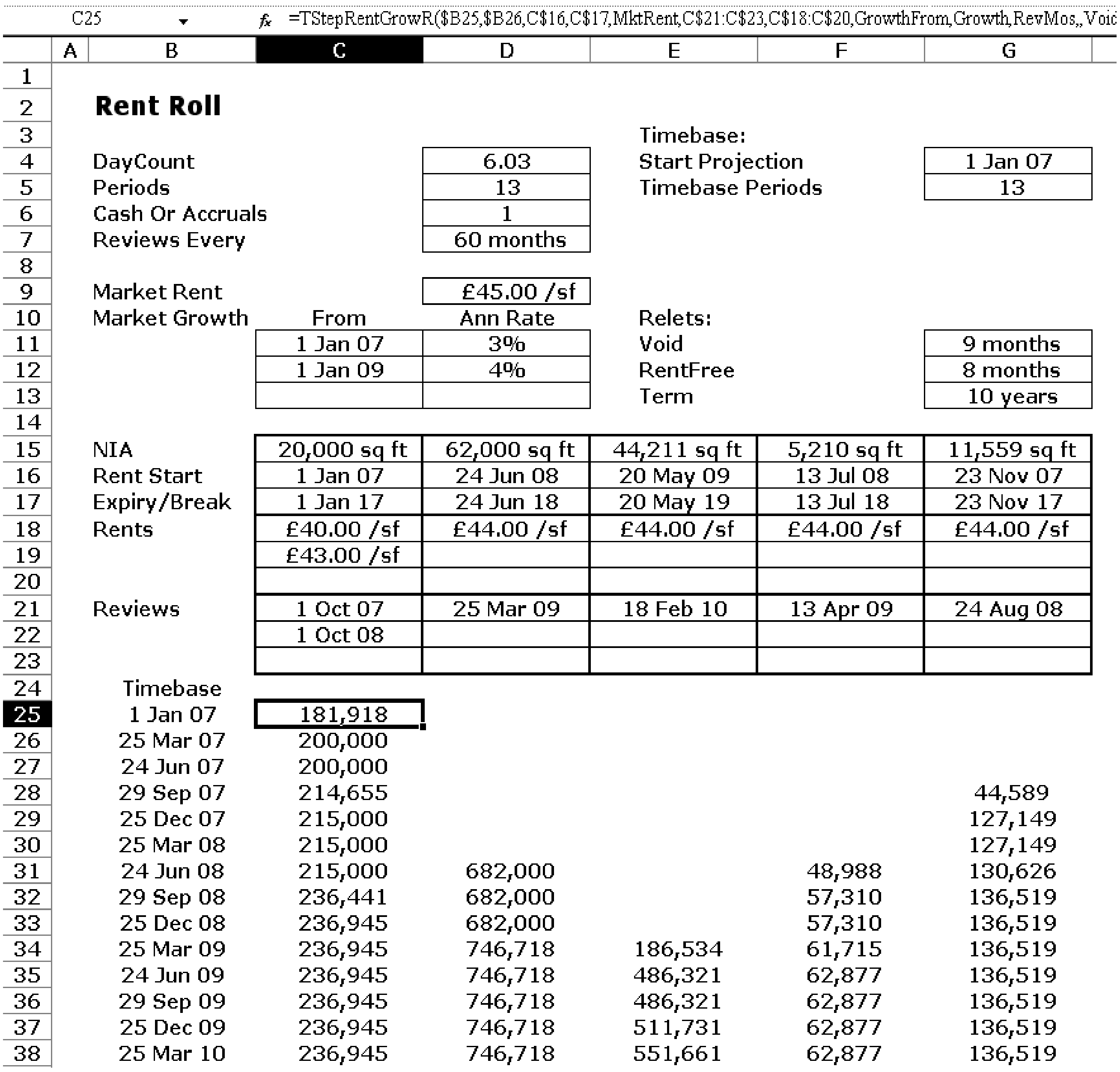
{=TStepRentGrowR(***Annualdates***,0, RentStartDate, ExpOrBrk, MktRent, Reviews, Rents, GrowthDates, GrowthRates, RevMos," ", ReletVoid,ReletRentFree,ReletTerm,daycount,13,1)}

	Budget Periods	Business Functions Normal Function Call	Business Functions Array Function Call
145	1 Jan 05	13.04	13.04
146	1 Jan 06	20.00	20.00
147	1 Jan 07	20.33	20.33
148	1 Jan 08	25.00	25.00
149	1 Jan 09	25.00	25.00
150	1 Jan 10	25.00	25.00
151	1 Jan 11	25.00	25.00
152	1 Jan 12	25.44	25.44
153	1 Jan 13	31.58	31.58
154	1 Jan 14	31.58	31.58
155	1 Jan 15	31.58	31.58
156	1 Jan 16	31.58	31.58
157	1 Jan 17	32.29	32.29
158	1 Jan 18	42.26	42.26
159	1 Jan 19	42.26	42.26
160	1 Jan 20	42.26	42.26
161	1 Jan 21	42.26	42.26
162	1 Jan 22	39.44	39.44
163	1 Jan 23		
164	1 Jan 24	4.00	4.00
165	1 Jan 25	59.95	59.95
166	1 Jan 26	59.95	59.95

Business Functions deals with several types of rent, and you will see several functions in the help file. The variations are due to modelling the following:

- Rents that go up with annual indexation, or a mixture of indexation and review.
- Rents where the reviews are specified as dates *from* a certain rent occurring or dates *up until* a rent runs.
- Rents that don't have any stepped rent section.
- Retail turnover rent.

The thing to remember with the Business Functions rent functions is that they are very much variants on the more basic projections functions in the library. So if you don't need all the functionality a function offers, its worth looking for a simpler one. *FStep*, for example, just does stepped projection, and its not property specific. *AnnGrow* and *ConGrow* do a constantly growing rate, and the simple *Grow* just grows something by a set of (not just one) growth rates. So the library really does work the same everywhere. Its sometimes astounding for people to see that, if they wanted to, they could model rent in *arrears* and interest in *advance*, such is the generality! In fact, we blocked some functions from projecting interest in advance to save some of our users from themselves, but the point is this: There *is* a standard way to model a projection, across the Business Functions library, and if you understand it in one place it *will* work in another. That is why, for example, its worth the time to understand how *DayCount*, *Periods*,*ProjMode*,*Time* and *Base* work, because they crop up everywhere, and always doing the same job in the same way.



C25 =TStepRentGrowR($B25,$B26,C$16,C$17,MktRent,C$21:C$23,C$18:C$20,GrowthFrom,Growth,RevMos,,Voic

	A	B	C	D	E	F	G
1							
2		**Rent Roll**					
3					Timebase:		
4		DayCount		6.03	Start Projection		1 Jan 07
5		Periods		13	Timebase Periods		13
6		Cash Or Accruals		1			
7		Reviews Every		60 months			
8							
9		Market Rent		£45.00 /sf			
10		Market Growth	From	Ann Rate	Relets:		
11			1 Jan 07	3%	Void		9 months
12			1 Jan 09	4%	RentFree		8 months
13					Term		10 years
14							
15		NIA	20,000 sq ft	62,000 sq ft	44,211 sq ft	5,210 sq ft	11,559 sq ft
16		Rent Start	1 Jan 07	24 Jun 08	20 May 09	13 Jul 08	23 Nov 07
17		Expiry/Break	1 Jan 17	24 Jun 18	20 May 19	13 Jul 18	23 Nov 17
18		Rents	£40.00 /sf	£44.00 /sf	£44.00 /sf	£44.00 /sf	£44.00 /sf
19			£43.00 /sf				
20							
21		Reviews	1 Oct 07	25 Mar 09	18 Feb 10	13 Apr 09	24 Aug 08
22			1 Oct 08				
23							
24		Timebase					
25		1 Jan 07	181,918				
26		25 Mar 07	200,000				
27		24 Jun 07	200,000				
28		29 Sep 07	214,655				44,589
29		25 Dec 07	215,000				127,149
30		25 Mar 08	215,000				127,149
31		24 Jun 08	215,000	682,000		48,988	130,626
32		29 Sep 08	236,441	682,000		57,310	136,519
33		25 Dec 08	236,945	682,000		57,310	136,519
34		25 Mar 09	236,945	746,718	186,534	61,715	136,519
35		24 Jun 09	236,945	746,718	486,321	62,877	136,519
36		29 Sep 09	236,945	746,718	486,321	62,877	136,519
37		25 Dec 09	236,945	746,718	511,731	62,877	136,519
38		25 Mar 10	236,945	746,718	551,661	62,877	136,519

Other Property Functions: Rates and Service Charges

Frequently you need to model an entire property's cash flows. I used to do this a lot in relation to properties whose lease we had 'taken over' and become the somewhat reluctant tenant. For this you need to model rates and service charges too.

UK Business Rates is a complex area and Business Functions takes the most direct, simplest approach possible, so ratings advisors may take issue with some of our methodology (we don't do transitional relief). However, the function has within it all the ratings multipliers from 1990/91 to 2005/2006 and basically works by you supplying a rental forecast and an inflation forecast. That is about it. The function can deal with empty rates and by default the rates are paid over 10 months, although you can set this to what you like. It looks like this:

```
UKEmptyRatesFcstR ( Time, Base, StartCost, StartLease,
ExpOrBrk, RatesMultiplier, StartFcst, FcstBase, FcstRate-
ableValues, InflationRates, InflationDates, [RatesHol_s],
[RelVoid_s], [RelTerm_s], [EmptyRatesFrac], [DCRates],
[PrdsRates], [ProjMode], [AbFin], [ROpts] )
```

With a fairly complex function like this its best to start with the spreadsheet example that comes with each function which is part of the basic product. We do experience and expect a few support questions on this type of function, but usually the function works to an acceptable level of accuracy.

More straightforward perhaps is service charges, which really a variation on the familiar theme of a constantly growing annual rate.

It looks like this:

```
ServChargeGrowR ( Time, Base, Start, ExpOrBrk, First-
SCRevDate, ServChargeTotal, GrowthDates, GrowthRates,
SCRevMonths, SCFromDates, SCPercentages, [RelVoid_s],
[RelTerm_s], [DayCount], [Periods], [ProjMode], [AbFin],
[ROpts] )
```

In essence you supply the service charge budget amount (this function applies growth, but there are other functions that use a stepped service charge budget. You also specify the tenant's service charge percentage over time. The rest of it is basically about specifying when the lease starts and how and if you want to handle relets. The really useful thing about this function is that the multiplication of the service charge percentages with the service charge budget amounts does not have to be coincident in time. That turns out to be very hard to do outside of a function, when, say you are budgeting over a 1 year time period with service charges increasing in August when the service charge percentage kicks up in November. Again, what we are trying to do is obtain the same accuracy as the property manager does, with the ability to project it over a timebase that can be flexed as required. As usual, you can project cash or accruals, with any payment periods.

Loans

A loan is as much a book-keeping problem as a modelling one - you have to keep track of balances, interest re-advanced etc, and in general there is a standard calculation you do at each time:

> *Closing Balance = Opening Balance + Advances - Repayment*

If you are rolling up interest it becomes:

> *Closing Balance = Opening Balance + Advances + Interest Advances - Repayment*

Fine and dandy. The problem comes when our loan stops fitting so neatly into our timebase:

- Suppose the interest rate changes part-way through an interest period.
- Suppose you want to budget a quarterly loan on an annual basis.

As usual, we will start with native Excel and progress to the Business Functions way. With native Excel, we more or less have to abandon hope of addressing the preceding two objectives, and indeed we will make the same kind of simplifying assumption we made when we were projecting rent:

- The timebase is the same as the interest periods.
- We will not attempt to deal with interest rate changes during an interest period.

The first thing we need is a look up function to translate our table of 'from' interest rates into a rate per period:

```
=IF(ISNA(LOOKUP(E22,OFFSET(IntDates,0,0,COUNTA(IntDates),
1),OFFSET(IntRates,0,0,COUNTA(IntRates),1))),0,LOOKUP(E22
,OFFSET(IntDates,0,0,COUNTA(IntRates),1),OFFSET(IntRates,
0,0,COUNTA(IntRates),1)))
```

This is cumbersome because it is stripping out blank rows and to be honest its clumsy and not really robust. You might want to resort to a VBA function straightaway here. I did a function in VBA (not a Business Function) that at least made the interest rate lookup less ugly:

```
{=a_safelookup(IntPmtDates,IntDates,IntRates,0)}
```

The interest rate per period is going to be our helper row for the calculation. The next thing is to set the (opening balance + advances - repayment) = Closing Balance grid on the spreadsheet - trivial stuff.

Then for the interest we do a simple ACT/365 interest formula:

```
=-(E$22-D$22)/365*E$33*D29
```

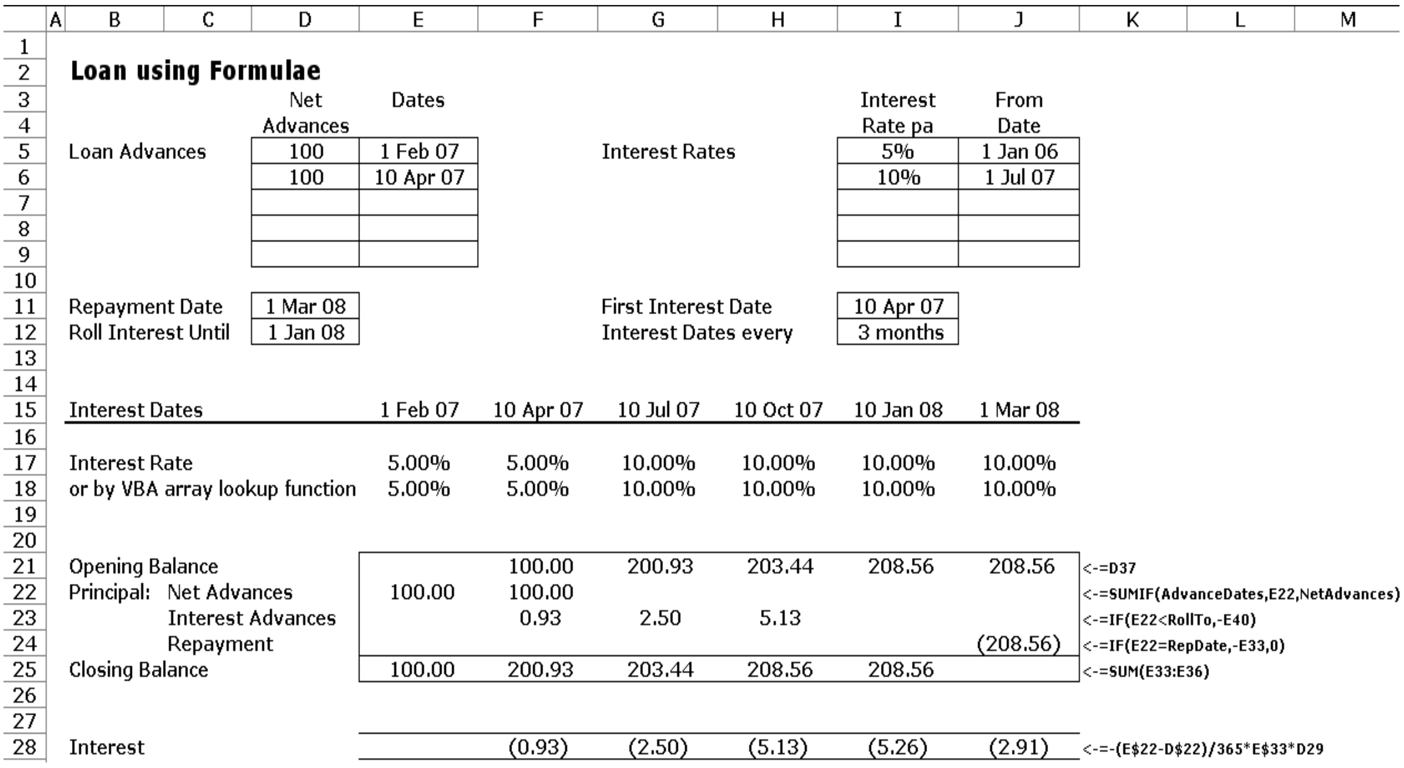
For interest roll-up, we just put in another line of interest advances:

```
=IF(E22<RollTo,-E40)
```

Looking at this again, it seems that if we are prepared to spend the effort at writing a VBA function (in the companion workbooks), we can cope with the interest rate changing part-way through the interest period:

```
=a_avlevels(IntPmtDates,RepDate,IntDates,IntRates,0)
```

So, we have a loan that is pretty accurate. It can cope with changing and stepped interest rates, rolled up interest, and if you don't mind a bit of VBA, we can do the interest rate changing partway through the period.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2		**Loan using Formulae**											
3				Net	Dates				Interest	From			
4				Advances					Rate pa	Date			
5		Loan Advances		100	1 Feb 07		Interest Rates		5%	1 Jan 06			
6				100	10 Apr 07				10%	1 Jul 07			
7													
8													
9													
10													
11		Repayment Date		1 Mar 08			First Interest Date		10 Apr 07				
12		Roll Interest Until		1 Jan 08			Interest Dates every		3 months				
13													
14													
15		Interest Dates			1 Feb 07	10 Apr 07	10 Jul 07	10 Oct 07	10 Jan 08	1 Mar 08			
16													
17		Interest Rate			5.00%	5.00%	10.00%	10.00%	10.00%	10.00%			
18		or by VBA array lookup function			5.00%	5.00%	10.00%	10.00%	10.00%	10.00%			
19													
20													
21		Opening Balance				100.00	200.93	203.44	208.56	208.56	<-=D37		
22		Principal:	Net Advances		100.00	100.00					<-=SUMIF(AdvanceDates,E22,NetAdvances)		
23			Interest Advances			0.93	2.50	5.13			<-=IF(E22<RollTo,-E40)		
24			Repayment							(208.56)	<-=IF(E22=RepDate,-E33,0)		
25		Closing Balance			100.00	200.93	203.44	208.56	208.56		<-=SUM(E33:E36)		
26													
27													
28		Interest				(0.93)	(2.50)	(5.13)	(5.26)	(2.91)	<-=-(E$22-D$22)/365*E$33*D29		

One improvement we can make before demonstrating the Business Function is to package the whole thing in a VBA function, which has been done in the workbook.

This has a lot of advantages, because you can really customise how the loan works then. Writing VBA is really easier and more logical than writing Excel formulae. To keep performance high, you will see that wherever possible the VBA functions we write in the demo workbook are array functions, i.e. they return an array of numbers to span the entire timebase. The way the VBA function works is that you specify, for each row where you use the function, a loan output number that governs whether you get interest, principal or some other parameter returned:

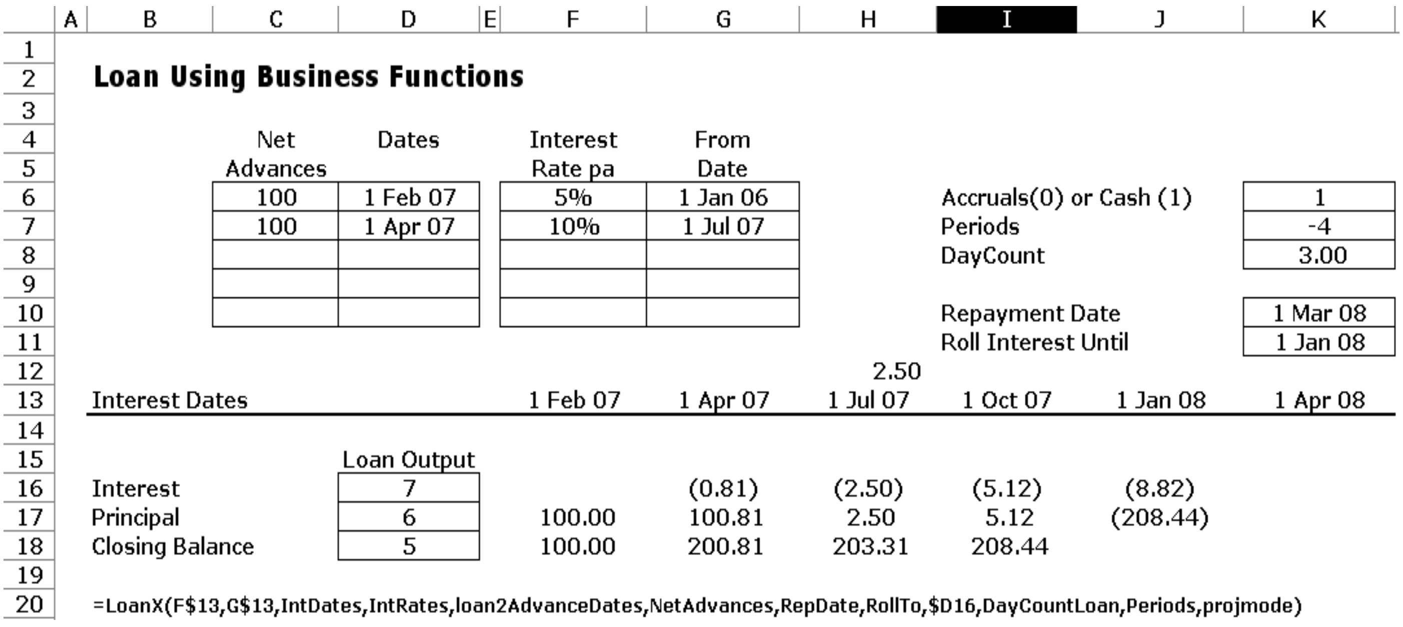
```
=a_loan(IntPmtDates,RepDate,RollTo,AdvanceDates,NetAdvanc
es,IntDates,IntRates,daycountloan,D31)
```

The Business Function, *LoanX*, is a more comprehensive solution again to the VBA version, but it works in a not too dissimilar way. It has no problem with interest rate changes part way through a period, and as usual it can do all combinations of daycount and payment periods. It can also work on cash and accruals, which is particularly useful for loans, where the accounting treatment really insists that loan interest is calculated on an accrued basis. Its also handy for bonds which may use all manner of interest conventions. The formula looks like this:

```
=LoanX(IntPmtDates,0,IntDates,IntRates,AdvanceDates,NetAd
vances,RepDate,RollTo,D40,daycountloan,-
AnnSeq(DateSeqNum(FirstIntDate),12/PmtFreq),projmode)
```

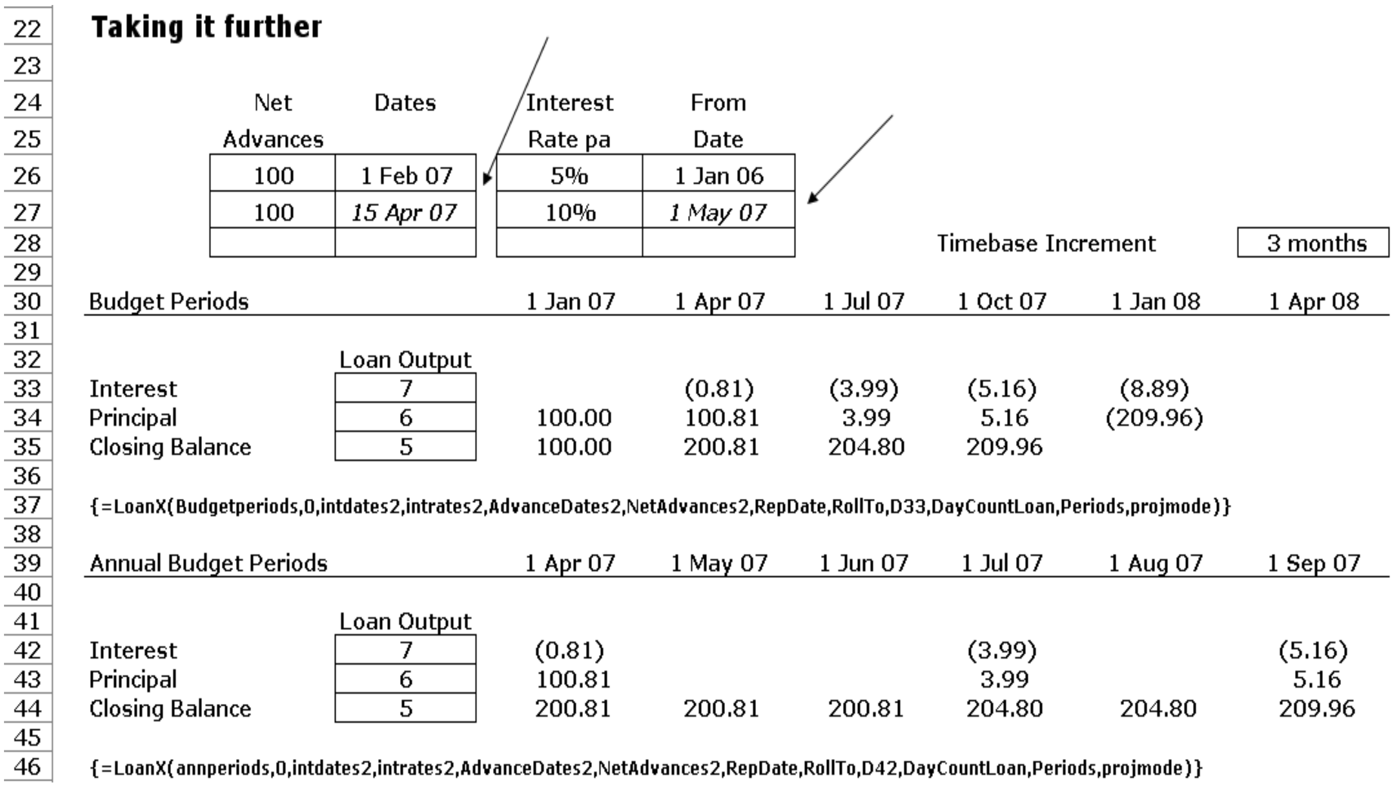
Note that, instead of inputting a *Periods* variable in the form mm.dd (e.g. 1.10,4.10,7.10,10,10), we have created one internally, and we have also made it negative, so that interest periods are in arrears:

```
-AnnSeq(DateSeqNum(FirstIntDate),12/PmtFreq)
```



	A	B	C	D	E	F	G	H	I	J	K
1											
2		**Loan Using Business Functions**									
3											
4			Net	Dates		Interest	From				
5			Advances			Rate pa	Date				
6			100	1 Feb 07		5%	1 Jan 06		Accruals(0) or Cash (1)		1
7			100	1 Apr 07		10%	1 Jul 07		Periods		-4
8									DayCount		3.00
9											
10									Repayment Date		1 Mar 08
11									Roll Interest Until		1 Jan 08
12								2.50			
13		Interest Dates				1 Feb 07	1 Apr 07	1 Jul 07	1 Oct 07	1 Jan 08	1 Apr 08
14											
15				Loan Output							
16		Interest		7			(0.81)	(2.50)	(5.12)	(8.82)	
17		Principal		6		100.00	100.81	2.50	5.12	(208.44)	
18		Closing Balance		5		100.00	200.81	203.31	208.44		
19											
20		=LoanX(F$13,G$13,IntDates,IntRates,loan2AdvanceDates,NetAdvances,RepDate,RollTo,$D16,DayCountLoan,Periods,projmode)									

The great thing about the Business Function, and the reason that treasury groups use it, is that it is truly timebase independent, and has the full range and accuracy of Business Functions date arithmetic.



Taking it further

Net Advances	Dates	Interest Rate pa	From Date
100	1 Feb 07	5%	1 Jan 06
100	*15 Apr 07*	10%	*1 May 07*

Timebase Increment: 3 months

Budget Periods	Loan Output	1 Jan 07	1 Apr 07	1 Jul 07	1 Oct 07	1 Jan 08	1 Apr 08
Interest	7		(0.81)	(3.99)	(5.16)	(8.89)	
Principal	6	100.00	100.81	3.99	5.16	(209.96)	
Closing Balance	5	100.00	200.81	204.80	209.96		

```
{=LoanX(Budgetperiods,0,intdates2,intrates2,AdvanceDates2,NetAdvances2,RepDate,RollTo,D33,DayCountLoan,Periods,projmode)}
```

Annual Budget Periods	Loan Output	1 Apr 07	1 May 07	1 Jun 07	1 Jul 07	1 Aug 07	1 Sep 07
Interest	7	(0.81)			(3.99)		(5.16)
Principal	6	100.81			3.99		5.16
Closing Balance	5	200.81	200.81	200.81	204.80	204.80	209.96

```
{=LoanX(annperiods,0,intdates2,intrates2,AdvanceDates2,NetAdvances2,RepDate,RollTo,D42,DayCountLoan,Periods,projmode)}
```

Conclusion

That concludes the tour of Business Functions projections capabilities. The starting premise is that, in a spreadsheet, you want timebase-consistent formulae that are the same across the timebase, and, if possible, you want the formula to be independent of any timebase that is chosen. because that's very hard, and gets even harder when you consider partial periods, you need a function. And because that function is called thousands of times a second in the workbook, it needs to fast. We started with the simplest case, discrete one-off payments, and progressed through spreading amounts over time, applying constant rates, and arrived at rent and loans, which is roundabout where the current frontier of projections functions is. The approach works because it robustly extends Excel's capability in a consistent, if at times involved, way. This little booklet will probably have raised more questions than it answers, assuming you got this far. Feel free to contact us through the website for further clarification. Until then, good luck with the modelling!

Appendix

About the Author

John Drummond has been financial modelling in spreadsheets since 1983. A Masters level Petroleum Engineer, he also has an MBA from London Business School. He has several years experience in the oil industry, has worked in venture capital with 3i and for nearly 10 years headed up Business Planning and Appraisal at Canary Wharf, Europe's largest commercial office development. He first wrote function libraries for spreadsheets in C++ back in 1992. In 2001 he founded Business Functions Ltd to develop and perfect a rock-solid business function library with cross-sector appeal. Over the last year this has been increasingly adopted by financial analysts worldwide. When not dealing with Business Functions issues he builds models for clients in property, resources and banking.

Downloadable Workbook

The workbook for this booklet is downloadable at:

`http://www.BusinessFunctions.com/pcs05`

Other Internet Resources

For Excel In General (the first two are esssential):

- Excel-L Email List:

  peach.ease.lsoft.com/archives/excel-l.html

- Excel-G Email List

  peach.ease.lsoft.com/archives/excel-g.html

- Dicks Blog

  www.dicks-blog.com/

- Chip Pearson's Website. Lots of information on specific tasks.

  www.CPearson.com

- John McGimpsey's Website

  www.mcgimpsey.com/

- John Walkenbach's Website and Blog. The leading Excel author.

  www.J-Walk.com

- OzGrid Website and Forum

  www.OzGrid.com

- ExcelTip Website and Forum

  www.ExcelTip.com

- MrExcel Website and Forum

  www.MrExcel.com

- Official Excel Developer's Blog (MS). Information on upcoming Office 12.

  blogs.msdn.com/excel/default.aspx

Our new forum for Excel and VBA in property generally

www.PropertyModellers.com

Business Functions

- Website

  http://www.BusinessFunctions.com

- The Help File

Visual Basic for Applications and Microsoft are trademarks of Microsoft Corporation