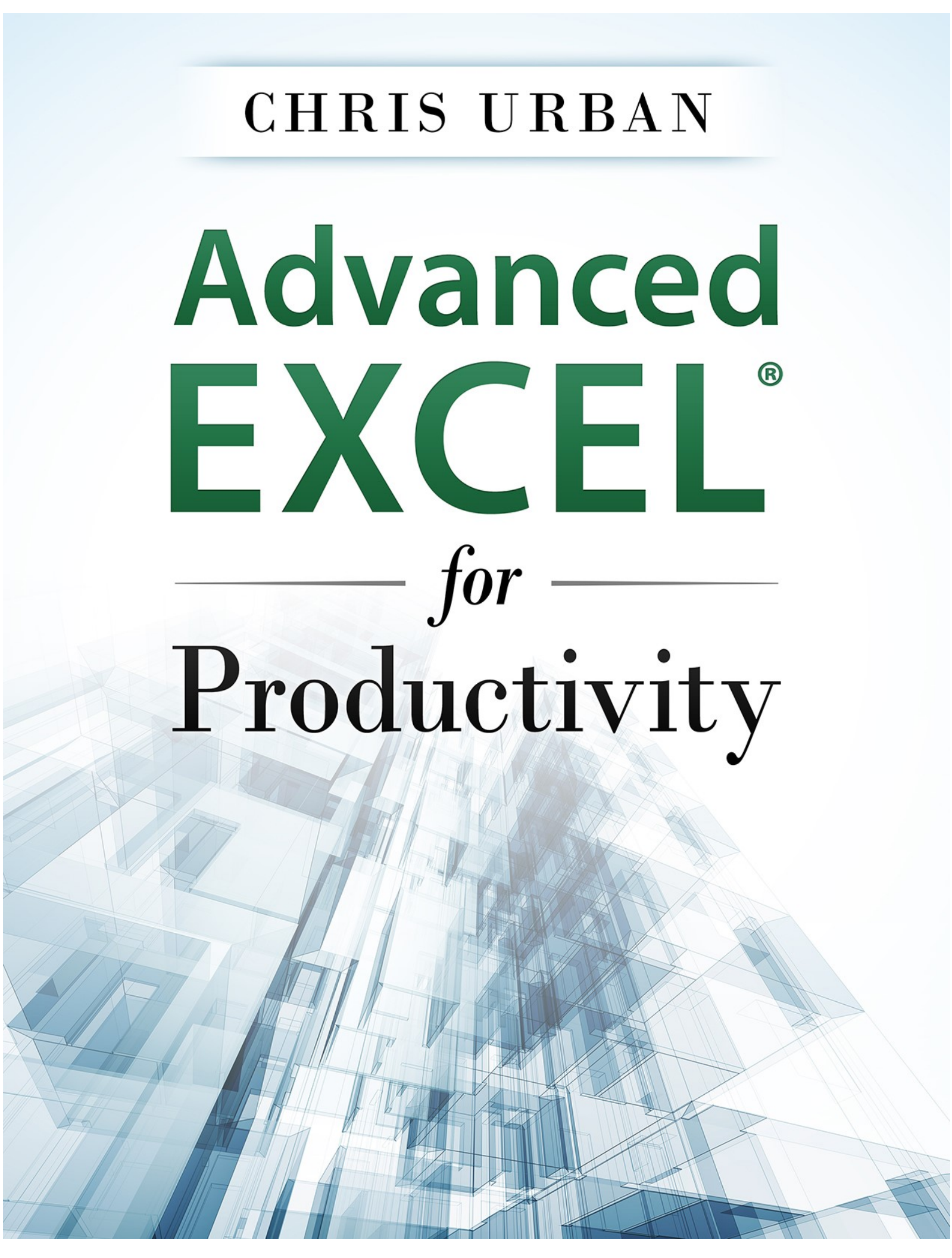
CHRIS URBAN

Advanced EXCEL®

for

Productivity



This is a free PDF version of Advanced Excel for Productivity

Feel free to share this Ebook with your friends and colleagues

To purchase the Paperback book please visit www.AdvancedExcelBook.com or https://www.facebook.com/advancedexcelbook

Advanced Excel for Productivity
Chris Urban

“Productivity is being able to do things that you were never able to do before.”

– Franz Kafka

“While one person hesitates because he feels inferior, the other is busy making mistakes and becoming superior.”

– Henry C. Link

Introduction

Why did I write this book? Why should you learn about Microsoft® Excel® from a physical book rather than websites or YouTube videos? The short answer is: because other Excel books I've come across are too boring, too basic, or too technical. It was time to write a great Excel book.

My goal with this book is to open your eyes to what's possible and to show you best practices using Excel. You'll save time, do more, and impress and intimidate your colleagues with your newfound skills.

We're skipping past the beginner level of: *"click the little C icon to copy..."* We'll also avoid getting too in-depth in any areas that are not useful for the typical day-to-day Excel user. Instead we'll dive right into keyboard shortcuts and never look back.

About Me & Excel

I've been using Excel for over 15 years, nearly every day. It started with basic math and charts in high school, followed by data analysis, statistics, and cash flow modeling in college. The real revelation came during my time in investment banking. That was the first time I learned from power users of Excel, building financial models and beautiful spreadsheets amazingly fast without ever touching the mouse.

I've used Excel for modeling at a hedge fund, running a startup company, and most recently as head of risk management at a multi-billion-dollar financial services company. Along the way, I've seen both some impressive spreadsheets and some terrible ones. I've seen massive and complex models, animated charts, decision trees, pivot tables, vlookups, company reports, embedded SQL, Monte Carlo simulations, and macros that automate all of the above. I want to pass the most useful of these skills on to you.

Software Compatibility – Windows or Mac

The keyboard shortcuts in this book are for the Windows® operating system. Unfortunately, Macs lack most of the keyboard functionality that make Advanced Excel fun – even if you're a die-hard Mac fan, I recommend trying out some of these skills in Windows. The following are only available for Windows (as of the 2016 version of Excel):

- Show trace dependent and trace precedent arrows
- Auto-fit column widths
- Increase or decrease decimals for number formatting
- Insert rows or columns
- VBA editor shortcuts F8 and F9

That said, a lot of great keyboard shortcuts do work in Mac 2016 and newer, so consider upgrading to that version if you're on an older one. Download the Mac version of my keyboard shortcuts reference sheet from www.AdvancedExcelBook.com.

For Windows you'll want Excel version 2007 or newer, but most of what you'll learn works in Excel 2003 (if you have a version from before 2003 – seriously, it's time to upgrade. You're missing out.) Very little has changed from Excel 2007 to 2016 considering the rapid advance of technology during that time! The core of Excel will likely stay the same more than it will change, so rest assured your keyboard shortcuts, formulas, pivot table tricks, and macros should continue to be applicable skills for decades to come.

Website

Go to www.AdvancedExcelBook.com for downloads, including one-page reference sheets, that complement the book very well.

Ebook (PDF) Version of this Book

You're reading the digital PDF version of this book (thanks for saving some trees). The file is fully searchable, and the table of contents items are clickable for easy navigation. This PDF is not compatible with Kindle or other e-readers, but should work well on a tablet or iPad. While the digital format makes this book easy to copy and share with others, please respect Copyright laws and the author's time and effort by purchasing your copy via www.AdvancedExcelBook.com.

Sources

This book has a relatively small number of cited sources. Where I directly referenced someone's work, I've cited those sources in the footnotes.[1] Most things I'll show you are the product of many years of trial and error, self-teaching, and learning from co-workers. Excel itself is my main source: I try a lot of things, and keep what works. The things that work best made it into this book.

[1] Although most footnotes just contain comments, like this one.

Acknowledgements

Writing and publishing a book on advanced Excel is a dream come true for me, and it's impossible to list all the people who have contributed along the way. For everyone I have worked with who gave me difficult problems to solve, or asked me "how do I do X in Excel?" – I want to say Thank You. Your questions and puzzles inspired me to push my own boundaries and constantly learn more. Your enthusiasm for my idea of writing this book kept me motivated throughout this process.

In particular I want to acknowledge my wife **Lori Urban** for helping me fine-tune this book and make it more readable. Thank you also for helping me market and sell this book. And of course for letting me spend countless hours in front of the computer writing it.

Special thanks to my copy editor, **Mariel Kyger, Ph.D.** She's the fastest reader I know and I'm still amazed she found so many errors to correct (any errors on this acknowledgements page are not her fault).

Thank you to **Ian Anderson** for his enthusiastic support of this book, and for contributing ideas such as using scatter plot trendline equations in lieu of single-variable regressions (Chapter 6).

Thank you **Alec Lentz** for contributing some great questions and challenges, especially related to charts and the INDIRECT function.

And of course, thank you to **Microsoft** for creating the single most useful software in world, and thanks to **Google** for helping me find answers when I didn't know something.

1: Quick Start with Advanced Excel

This is a book about using Excel for maximum productivity. My goal is to make you a faster, smarter, more efficient user of Microsoft Excel. If you picked up this book, you've probably found Excel useful in your work so far – but you may not have had formal training with it, or seen the full extent of what is possible.

You are ready for Advanced Excel if you are already pretty comfortable with the essentials. That means you've done all of the following tasks and have no difficulty with them:

- Apply formulas and calculations to data. Drag down and copy formulas with anchored cells
- Enter formulas and calculations purely with the keyboard. That means typing a formula like this, without actually clicking on cells A1 and B1: `=(A1+B1)*2+3`
- Reference ranges of cells, including entire columns or rows
- Use basic functions such as SUM without opening the function wizard
- Create and modify charts, including adding more data to a chart
- Use keyboard shortcuts for the essentials: Cut, Copy, Paste, Bold; Italics; Underline; Save; Undo
- Format cells and make your sheet look attractive

Here are some of the ways you'll soon step up your game with Advanced Excel:

- Significantly speed up your work using keyboard shortcuts (Chapters 2 and 3)
- Know the correct functions to perform common tasks. Use complex formulas, including multiple IF statements and lookups, to work more effectively with numbers (Chapter 4)
- Use named references in your formulas. Deal with Excel errors such as #NUM! and #REF! (Chapter 5)
- Work with large amounts of data, especially with pivot tables. Make nice charts. (Chapter 6)
- Build good Excel models, and detect any mistakes before they become costly. Understand complex worksheets and quickly figure out how someone else's models work. Use trace precedents and trace dependents (Chapter 7)
- Automate repetitive work with macros and VBA (Chapter 8)
- Break any password protection; speed up a slow Excel file; define custom number formats; and use add-ins to make Excel more powerful (Chapter 9)

In the process of building your Excel skillset, I'll tell you about some simple (and often silly) Excel errors of various people in business, academia, and government. These mistakes were well publicized in the press, and caused huge financial and/or reputational damage to those involved. I'll show you best practices to help avoid these kinds of problems!

How to use this book to get the most out of it? Have Excel in front of you while reading so you can **practice the skills you learn here**. They won't stick by just reading; you should follow along with the exercises and screenshots. Some of it might be review for you; I try to move through the more basic sections pretty quickly. I recommend reading the chapters in order. The Chapters 2-4 are simple, but require a lot of repetition to really stick. Chapters 5-8 add complexity and build on the earlier topics. The final chapter is a grab bag of cool tricks. As you practice your new skills, use the three reference sheets that complement this book: keyboard shortcuts, functions, and VBA.

Without further delay, let's kick things off by looking at keyboard navigation, followed by keyboard shortcuts. If you want to see next-level productivity improvements in your use of Excel, the keyboard is the place to start.

2: Keyboard Navigation

Keyboard navigation is one of the coolest things about Excel (yes, that means not touching the mouse!). Using the keyboard as the primary controls will hugely improve your day-to-day productivity. It's also one of the main ways to set yourself apart as an advanced user of Excel. You'll be the person in the office who can use Excel with just the keyboard. Your legend will be told far and wide.

If your primary tool for using Excel has been the mouse, your world is about to change. First things first, pick up your mouse and put it away – behind the monitor, on top of your desktop tower, wherever it is out of reach. You don't need to touch the mouse in this chapter or the next one. This is Proper Mouse Placement for Advanced Excel.

If you're on a laptop – even better! Those little touchpads are horribly inefficient for Excel. You'll spend half your working life scrolling around and awkwardly clicking.[2] The keyboard is here to help.

At first, ignoring the mouse will be uncomfortable, and it will definitely slow you down. You will be tempted to use the mouse "just a little bit." But trust me: once you master the right shortcuts and key combinations, you'll be cruising through Excel much faster than before!

The Essentials: [Arrows], [CTRL], [SHIFT]

Keyboard navigation all starts with the arrow keys, combined with [CTRL] and [SHIFT]. These are the core tools of keyboard navigation:

- [Arrows] Move one cell at a time
- [CTRL] + [Arrows] Move to end of contiguous range
- [SHIFT] + [Arrows] Select cells while moving
- [CTRL] + [SHIFT] + [Arrows] Select while moving to end of contiguous range

Use the **[UP], [DOWN], [LEFT], [RIGHT] keys to move around**. If you've never used these, it's time to take them for a test drive. In close quarters, arrow keys are faster than picking up the mouse and clicking around. What about navigating the entire sheet? These arrow keys (I'll just refer to them as [Arrows] going forward) move you one cell at a time, which is pretty slow. Things get much better when you add the [CTRL] key.

[2] You might want to turn off your laptop touchpad entirely while you're working in Excel. You don't want to accidentally click with your palms while you're typing! For most computers you can easily turn off the touchpad. I recommend buying an external Bluetooth mouse if you do any meaningful amount of work on a laptop.

[CTRL] + [Arrows] move to the end of a contiguous cell range. Contiguous just means they are together with no empty cells in between; so if you have a big block of cells with data in them, move to the end of them with [CTRL] plus one of the arrows.

The [CTRL] + [Arrows] shortcut is especially useful when you have a large amount of data. Let's say you have a big data set with hundreds of rows, and you're at row 1. How do you know how many rows there are? You have a few options:

a) Start scrolling down with the mouse all the way. That is slow and doesn't look advanced at all.
b) Use the little scroll bar at the right side of the window. Still so slow! And not advanced.
c) Press [CTRL] + [DOWN]. Immediately you're at the bottom of the data and it's easy to see which row you are in. Perfect!

Use the [Arrow] keys, and mix in [CTRL] when needed, to move around your spreadsheet. This will become natural with practice, and in most cases much faster than finding your mouse pointer and clicking where you're going. Faster = more productive!

[SHIFT] + [Arrows] selects cells as you move around. Hold down [SHIFT] and use the arrows to select an area. Once you have some cells selected, you'll be able to do cool things with them (copy, paste, insert rows, delete, move things around, etc.) – all using the keyboard, of course.

[SHIFT] + [CTRL] + [Arrows] works as a combination of [CTRL] and [SHIFT], meaning you select cells and move to the end of a contiguous range.

If you have a fairly large set of contiguous data, you now have a very fast way to select that entire data set (so you can copy it, format it, reference it, or delete it). If you're using the mouse, you would have to click the top-left cell, hold down the mouse, scroll over to the bottom right cell, and let go. Too slow! Try this instead:

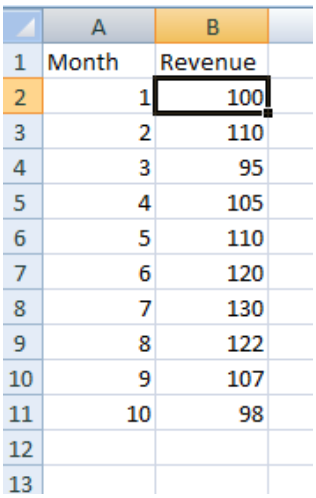
1. Start anywhere inside the data set
2. [CTRL] + [LEFT] and [CTRL] + [UP] to go to the top left cell
3. [CTRL] + [SHIFT] + [RIGHT] and [CTRL] + [SHIFT] + [DOWN] to go to the bottom right cell

Much easier and much faster with the keyboard!

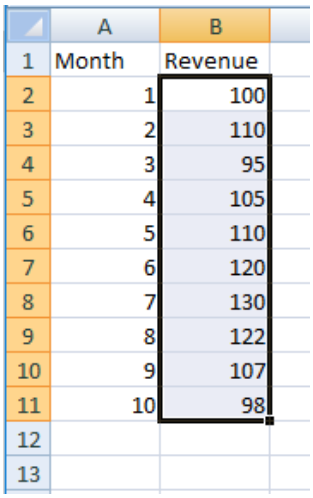
Another common task: auto-sum a column of data. You can do this with just the keyboard:

1. Use [Arrows] and/or [CTRL] + [Arrows] to make your way to the top of the data column.
2. [CTRL] + [SHIFT] + [DOWN] to select to the bottom of the column.
3. [ALT] + [=] to sum this data into the cell below it.

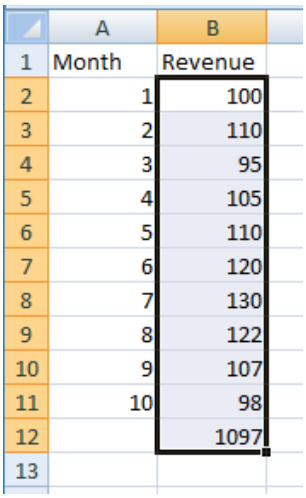
[ALT] + [=] is the shortcut to auto-sum your selection. Below are the screenshots for each step above, all with the keyboard:



	A	B
1	Month	Revenue
2	1	100
3	2	110
4	3	95
5	4	105
6	5	110
7	6	120
8	7	130
9	8	122
10	9	107
11	10	98
12		
13		



	A	B
1	Month	Revenue
2	1	100
3	2	110
4	3	95
5	4	105
6	5	110
7	6	120
8	7	130
9	8	122
10	9	107
11	10	98
12		
13		



	A	B
1	Month	Revenue
2	1	100
3	2	110
4	3	95
5	4	105
6	5	110
7	6	120
8	7	130
9	8	122
10	9	107
11	10	98
12		1097
13		

Get a feel for using [CTRL], [SHIFT], and the [Arrows] for navigation. Take your time with this, and force yourself to use the keyboard as your primary tool in Excel. It will feel natural soon enough. If you're still skeptical, here's a brief preview of a few more things you'll be able to do with the keyboard after the next chapter:

- "Drag down" formulas
  - Select the cell with the formula, use [SHIFT] + [DOWN] to select the target range, then hit [CTRL] + [D]
- Cut and paste and move things around
  - In addition to the arrows, you'll be using [CTRL] + [C], [CTRL] + [X], and [CTRL] + [V], plus a lot of Paste Special.
- Edit a formula in a cell (without double-clicking the cell, or clicking in the formula bar)
  - [F2] to go inside the cell to edit it; [Enter] to enter it; [ESC] to back out without making changes.
- Select an entire column (without right-clicking the letters in the column headings)
  - [CTRL] + [SPACE]
- Insert a row
  - [ALT], then [I], then [R] … if this is weird at first, don't worry, we'll get there.
- Go to a different tab (worksheet)
  - [CTRL]+ [PAGE UP], [CTRL] + [PAGE DOWN]. That's coming up.
- Check if a row is empty
  - [CTRL] + [RIGHT] to go all the way right, and [CTRL] + [LEFT] to go all the way left in the row. If you don't hit any cells along the way, you know the row is empty.

Example: Filling Formulas in an Entire Column

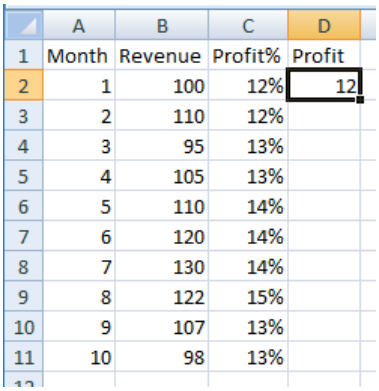
Here's a common situation: you have a column of numbers, and you want to apply some formula to them in the next column to the right. You start by entering the formula in the first row. Now you want to fill this formula down the entire column.

a) Using the mouse, you fill formulas down by grabbing the bottom-right corner of the cell (the little square), and drag it down. It works, but it's slow and not very impressive.
b) A faster way with the mouse is to double-click the same little square; this causes the formula to fill all the way down along the adjacent column.

c) My preferred way is to use a series of keyboard shortcuts. I call it the counter-clockwise method, which we'll cover step by step.

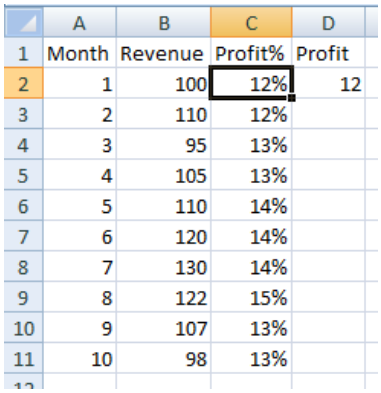
The keyboard method involves more steps. But with some practice, it will be faster than moving your hand to look for the mouse. Below are the keyboard shortcut steps, starting from the cell that contains your new formula, at the top of the new column (that's the first screenshot).

1. [LEFT] Go left to the top of the adjacent column which has data in it
2. [CTRL] + [DOWN] Go to the bottom row of this data column
3. [RIGHT] Go to the bottom of the new column
4. [CTRL] + [SHIFT] + [UP] Select the new column including the top row with your formula
5. [CTRL] + [D] Fill formulas down within the selection



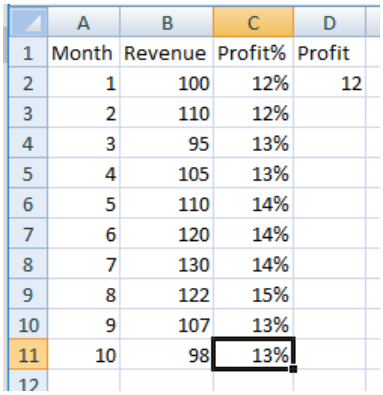
	A	B	C	D
1	Month	Revenue	Profit%	Profit
2	1	100	12%	12
3	2	110	12%	
4	3	95	13%	
5	4	105	13%	
6	5	110	14%	
7	6	120	14%	
8	7	130	14%	
9	8	122	15%	
10	9	107	13%	
11	10	98	13%	

Starting point



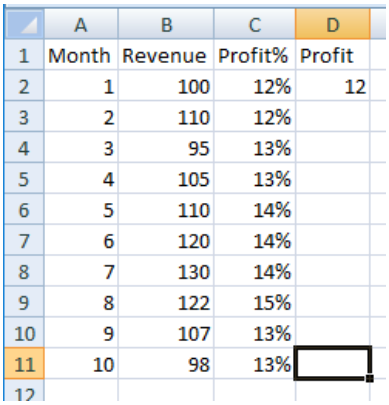
	A	B	C	D
1	Month	Revenue	Profit%	Profit
2	1	100	12%	12
3	2	110	12%	
4	3	95	13%	
5	4	105	13%	
6	5	110	14%	
7	6	120	14%	
8	7	130	14%	
9	8	122	15%	
10	9	107	13%	
11	10	98	13%	

Step 1: Go left



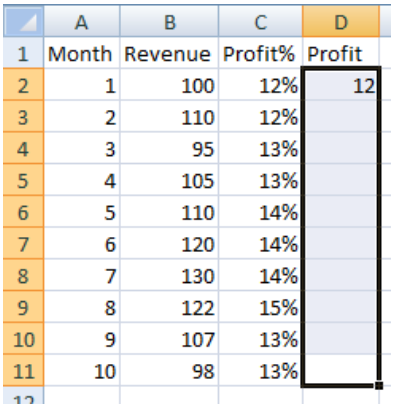
	A	B	C	D
1	Month	Revenue	Profit%	Profit
2	1	100	12%	12
3	2	110	12%	
4	3	95	13%	
5	4	105	13%	
6	5	110	14%	
7	6	120	14%	
8	7	130	14%	
9	8	122	15%	
10	9	107	13%	
11	10	98	13%	

Step 2: Go to bottom of column



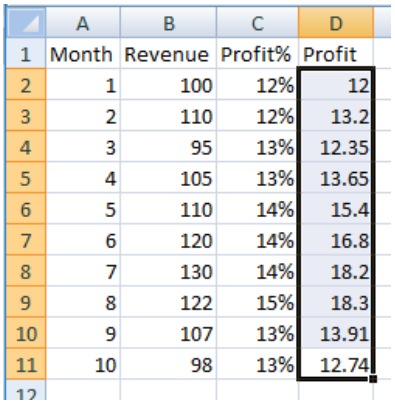
	A	B	C	D
1	Month	Revenue	Profit%	Profit
2	1	100	12%	12
3	2	110	12%	
4	3	95	13%	
5	4	105	13%	
6	5	110	14%	
7	6	120	14%	
8	7	130	14%	
9	8	122	15%	
10	9	107	13%	
11	10	98	13%	

Step 3: Go back right



	A	B	C	D
1	Month	Revenue	Profit%	Profit
2	1	100	12%	12
3	2	110	12%	
4	3	95	13%	
5	4	105	13%	
6	5	110	14%	
7	6	120	14%	
8	7	130	14%	
9	8	122	15%	
10	9	107	13%	
11	10	98	13%	

Step 4: Select up to top of column



	A	B	C	D
1	Month	Revenue	Profit%	Profit
2	1	100	12%	12
3	2	110	12%	13.2
4	3	95	13%	12.35
5	4	105	13%	13.65
6	5	110	14%	15.4
7	6	120	14%	16.8
8	7	130	14%	18.2
9	8	122	15%	18.3
10	9	107	13%	13.91
11	10	98	13%	12.74

Step 5: Fill formulas down

Visualize going in a counter-clockwise circle. Before the final step, you end up selecting exactly the range of cells you want to fill. Then the extremely handy [CTRL] + [D] shortcut fills the selection with your formula.

Use this technique and use it often! Filling formulas down is a very common task in Excel, and anyone watching will be highly impressed with your skills!

As a final tip: the end result has very poorly formatted numbers right now. Press [CTRL] + [SHIFT] + [1] to quickly apply some nice number formatting with two decimals. We'll cover formatting with the keyboard in the next chapter.

Other Keyboard Navigation

You'll need a few more keyboard shortcuts to effectively navigate the entire Excel workbook:

- [CTRL] + [PAGE UP] Go to previous sheet
- [CTRL] + [PAGE DOWN] Go to next sheet
- [CTRL] + [Mouse Scroll] Zoom in/out
- [CTRL] + [TAB] Switch between open Excel files
- [ALT] + [TAB] Cycle through open Windows programs
- [ESC] Close dialog box with Cancel (also: cancel editing a cell)
- [ENTER] Close dialog box with OK (also: finish editing a cell)
- [TAB] Move one cell to the right, works like [ENTER] in data entry
- [PAGE UP] Scroll up one page worth of rows
- [PAGE DOWN] Scroll down one page worth of rows
- [CTRL] + [HOME] Go to cell A1
- [CTRL] + [END] Go to the bottom-right-most cell you have edited
- [CTRL] + [F] Find (search for cell contents)

Switching between sheets in the same file: [CTRL] + [PAGE UP] and [CTRL] + [PAGE DOWN]. These shortcuts take you to the previous sheet or the next sheet. The keyboard method is usually much faster than picking up the mouse and clicking the little tabs at the bottom, especially if you're switching back and forth frequently between two tabs.

Zooming in/out with [CTRL] + [Mouse Scroll]. Yes, this technically involves the mouse, but it's really the easiest way to zoom in and out in Excel. I usually prefer doing [CTRL] + a single scroll down to get to 85% zoom. It'll be big enough that you don't distort any formatting, but small enough to see a larger area than the default 100% zoom.

Switching between Excel workbooks with: [CTRL] + [TAB]. If you have multiple files open within the same instance of Excel, you can easily switch between them with [CTRL] + TAB]. (If you're practicing and don't have another Excel file open, hit [CTRL] + [N] to start a new one).[3]

Switching between Excel and other programs: [ALT] + [TAB]. You probably already know this one since it's a general Windows shortcut. Hold down [ALT] and keep hitting [TAB] to cycle through all open programs on your computer. It always starts with the most recent other program you were in.

The [ESC] key is very versatile for closing dialog boxes, or cancelling things. Dialog boxes are various popups such as the Format Cells box. They have OK and Cancel buttons and various options.

If you just want to get out, use [ESC] as your Cancel button without saving changes. Use [ENTER] as your OK button. You can even make dialog box changes with just the keyboard, but we'll cover those later. For now, if you are stuck in a dialog box and have the urge to pick up the mouse, just hit [ESC] instead.

[TAB] lets you move one cell to the right. Exactly like the right arrow, except it works like [Enter] with data entry. I prefer to always use [ENTER] and then the arrows to get where I want to be, but if you are entering data horizontally, [TAB] will save you time.

[SHIFT] + [TAB] works like [TAB], except moves you to the left. I have never found good use for this one, it's safe to just forget about it.

---

[3] I highly recommend having no more than two Excel files to switch between. It gets tougher to cycle through a lot of files and remember where you were. If you must have more than two Excel files open, it's often better to have the rest of them in a separate instance of Excel. You basically start Excel again and open more files in that new version. The two instances won't interfere with each other when switching with [CTRL] + [TAB].

[PAGE UP] / [PAGE DOWN] lets you scroll up or down one "page" worth of rows… or about a screen height. Useful for navigation when [CTRL] + [UP] or [CTRL] + [DOWN] don't get you where you want to go, such as the middle of a long contiguous range of data.

[CTRL] + [HOME] goes to the first cell, A1.

[CTRL] + [END] goes to the bottom-right-most cell you've ever edited. In other words, this is the right-most column and lowest row that at some point you have made changes to. If all of your sheet's contents are bounded within B3 through G10, [CTRL] + [END] goes to G10. If you enter a value in N5, [CTRL] + [END] now goes to N10. If you now erase the contents of N5, [CTRL] + [END] still goes to N10. Excel doesn't forget that you used column N at one point.

[CTRL] + [END] is useful for two reasons: you might find that there is additional data, or perhaps hidden or invisible information, in rows or columns that are out of view. [CTRL] + [END] lets you know there's something there, or at least, there used to be something there. Secondly, there's a neat trick for reducing the file size of your spreadsheets. We'll go over this in Chapter 9, but I'll add a preview in this footnote.[4]

Regular **[HOME]** and **[END]** (without [CTRL]) are much less useful. Home goes to the left-most cell in the row, so column A of the same row. You could do the same with [CTRL] + [LEFT], so don't worry about this shortcut. [END] is even less useful. It turns on "End Mode", which is the same as if you held down [CTRL] while pressing the arrow keys. You're better off just using [CTRL], but if you're curious about End Mode then go ahead and try it.

[CTRL] + [F] brings up the Find dialog box. Sometimes you know what text or formula you're looking for, but you're not sure where on the sheet it is. Within the find box, start typing for what you need. Press [ENTER] to find. I consider this part of "keyboard navigation" because you can use it to get to specific parts of your sheet very quickly!

A quick note on Find: by default, Find searches within formulas and not the values of the cell. If you have a cell whose output value is 5, but the formula is `=2+3`, searching for "5" will not go to this cell. You can change to searching for values by clicking Options in the Find box, and choosing Look In: Values. Sorry, this part does use the mouse. Read Chapter 9 for more Find/Replace Tips.

Entering and Editing Formulas

Besides navigating with the keyboard, you'll also spend a lot of time entering and editing formulas. Let's see how that works without the mouse.

When entering formulas into a new cell, just start typing the formula with the [=] key. Learn the functions you need (in Chapter 4), and enter them by typing the formula directly. For example, you can sum a range just by typing `=SUM(A1:A5)`.

This brings us to entering cell references in formulas. You can still use the keyboard, in two possible ways:

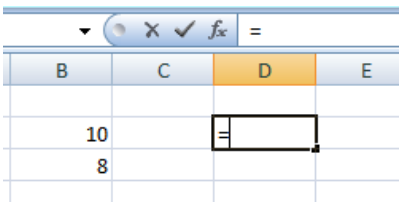
1. Directly type in the cell reference you want, with cell letters and numbers ("A1:A5" above).
2. Use the keyboard to select cells or ranges while you're in the middle of entering a formula.

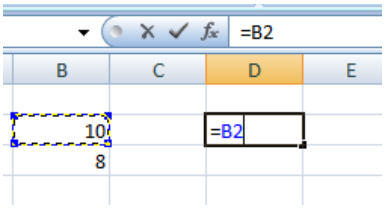
---

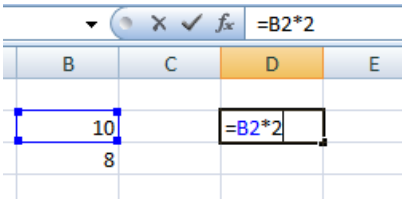
[4] Sometimes you might notice you have a large file (at least 5 MB), but when you open it up there's only visible information in a small part of it. This should take a lot less than 5 MB – and you're right, except sometimes Excel thinks you're using a huge area of the sheet and takes up file space. Use [CTRL] + [END] to see how much of the sheet Excel thinks you're using. To reduce the size of your file: use [CTRL] + [END] to find the last cell. Delete any unnecessary rows and columns working up and left from the last cell. Then you must *save* and *close* the file immediately. When you re-open it, the area within [CTRL] + [END] will be smaller, and your file size will be smaller too.

To reference cells in a formula with the keyboard, use the [Arrow] keys as usual! For example, to enter this formula in cell D2: `=B2*2` use these steps:

1. Press [=]
2. Hit [LEFT] [LEFT] to select B2 and add that reference to the formula
3. Keep typing the formula `*2`

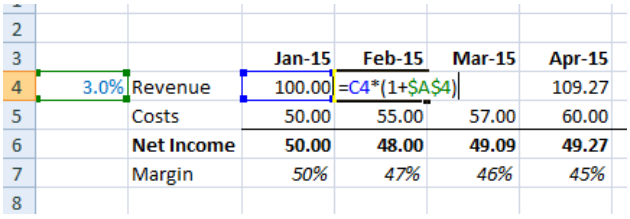


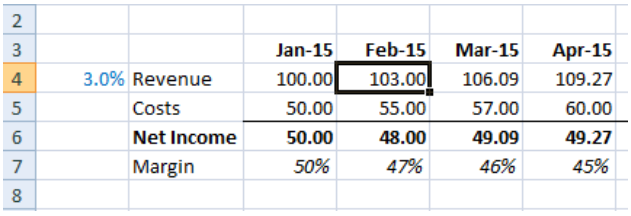




Use [F2] to jump inside the formula within a cell, so you can make changes to it. Now you are editing the formula inside the cell itself, without using the formula bar on top. Most of the time, this is far more efficient than shifting your eyes away from the cell (where you're navigating) to the formula bar (where you're editing). Edit your formulas within the cell and your work will flow much better. Use [F2].

[F2] and [ESC] are your best friends for formula editing. [F2] lets you edit a formula, and [ESC] exits editing without making changes. Alternate [F2] and [ESC] to quickly see what's inside a cell. Excel very helpfully color-codes the cells that drive each formula.





Cruise through cells with alternating [F2] and [ESC], using [Arrow] keys to keep moving to the next cell. It's a great way to see how the sheet's formulas flow logically, and to make sure the references are correct in each column and row. In the screenshots above, I can use [F2], [ESC], [RIGHT], [F2], [ESC], [RIGHT], and so on to quickly look at the formulas for the Revenue row.

> **Advanced Tip**
>
> Use [F2] instead of the mouse to review and edit formulas. It's a huge time-saver.

Unfortunately [F2] and [ESC] have a useless key in between them: [F1]. Try not to push that one accidentally; it pulls up the built-in help menu, which is quite slow and much less helpful than a Google search.[5]

If you do accidentally bring up the Help window, you can close it with the keyboard. [ALT] + [F4] closes the active window, just like the X button in the top-right corner. It's a helpful keyboard shortcut for all Windows apps.

[5] If you work on a desktop with an external (not too expensive) keyboard, I recommend removing the [F1] key. You can just pry it off with your fingers – and if you ever decide you miss it (you won't) you can always put it back on. People sometimes think my keyboard is broken, but I tell them it's more efficient without the [F1]!

Recap of Keyboard vs. Mouse

Getting used to keyboard navigation takes practice! Combine these tools with the keyboard shortcuts you're about to learn in the next chapter. With time, you'll be cruising through spreadsheets, updating formulas, and error-checking all your work without the mouse. That's what I call Advanced, Productive Excel!

By the way, you'd have to be a little crazy to *never* touch the mouse in Excel. The idea is to use the keyboard when it's the most efficient tool for the job, and not over-use it.

I recommend using the keyboard for these kinds of tasks:

- Understanding a large spreadsheet
- Entering formulas and checking them for consistency and typos (using [F2] and [ESC])
- Basic formatting such as bold, underline, resize columns
- Copy-pasting, including paste special; inserting rows; moving cells around
- Especially on a laptop: use the keyboard as much as possible; touchpads are very slow

When to still use the mouse:

- More complicated formatting, conditional formatting, borders
- Building and manipulating Pivot Tables (Chapter 6)
- Naming cells and managing names (Chapter 5)
- Creating and modifying charts
- Just scrolling around on a spreadsheet, without a care in the world

In the next chapter we'll cover the most useful Keyboard Shortcuts that will change the way you see Excel.

3: Keyboard Shortcuts

We're about to cover a large number of keyboard shortcuts to improve your Excel productivity. Remember, this is not a competition to see who knows the most shortcuts! Learn the ones you will actually use. For most people, these are the ones covered in this chapter and listed on the shortcuts sheet.

We'll start with basic shortcuts that should be familiar already. These shortcuts should be muscle memory for a regular user of any Office software. Take them for a test drive, and please promise me that you'll never pick up the mouse to click the little "B" button when you want to bold text. Then we can move on to the advanced stuff!

- [CTRL] + [Z] Undo
- [CTRL] + [Y] Redo
- [CTRL] + [C] Copy
- [CTRL] + [X] Cut
- [CTRL] + [V] Paste
- [CTRL] + [B] Bold
- [CTRL] + [U] Underline
- [CTRL] + [I] Italics
- [CTRL] + [N] New Workbook
- [CTRL] + [S] Save!

The shortcuts covered in this chapter (almost 100 of them!) are available as a full-page reference sheet on this book's website: www.AdvancedExcelBook.com. Print it out, stick it somewhere near your desk, and use it as you learn the rest of this chapter.

Aren't there way too many shortcuts to remember? Focus on the ones that would help you day-to-day, and build from there. Start with just these shortcuts:

1. Keyboard navigation with [ARROWS], [CTRL], and [SHIFT]
2. The basic [CTRL] shortcuts, such as copy-paste, bold, save, undo, and redo
3. [ALT] keys for Paste Special (we'll cover this next)
4. [ALT] keys to insert and delete cells (we'll cover this soon)
5. [F2] to edit cell values and formulas
6. [ALT] shortcuts for trace dependents and trace precedents

A note about saving: **save often**! Save different versions (MyFile v1.xlsx, MyFile v2.xlsx, etc) if you're doing something complex. [CTRL] + [S] should be pretty much a reflex.

A note about redo: do you ever use it? It's more than just "undo of an undo". If you apply some formatting to a cell, and you want to do the same with another cell: use redo: [CTRL] + [Y]. Think of redo next time you find yourself doing the same thing over and over – it's especially useful with cell formatting.

Let's throw in two more shortcuts to complete the list of basics:

- [F12] Save As
- [DEL] Clear Cell Contents

[F12] is a neat little shortcut, and significantly faster than getting to Save As using all the menus (especially in 2013 or newer Excel). Besides saving your work often, I highly recommend **saving multiple versions** of the same file. Just add "v1", "v2", "v4.2", and so on to end the file name. The point is: Excel can crash, and Excel files can become corrupted and broken. Disasters will happen sooner or later. Minimize the damage by saving multiple copies of your work!

Finally, the [DEL] key is used to clear the contents of cells. I've seen people use [SPACE] for clearing out cell contents, which is the wrong method – while the cell appears empty, the single space character messes with any COUNTA or similar formulas referencing it. Don't take a chance and don't be sloppy when clearing cells!

> As a bonus tip – don't do what an employee at a Westpac, a publicly traded bank in Australia, did in 2005 with a financial model sent out to analysts: the employee sent out a spreadsheet that contained confidential, un-released data on the latest profit numbers. The data was accessible "with a minor manipulation of the spreadsheet." What could that "minor manipulation" be? Apparently the employee thought they were hiding the numbers by applying a black background fill to those cells; obviously that doesn't hide the actual cell contents.[6]

The lesson: if you want to properly clear data from some cells, use the [DEL] key!

The [ALT] Key

The true sign of an advanced Excel user, besides confident use of keyboard navigation, is use of [ALT] key shortcuts. You're already in the top 5% of Excel users if you use [ALT] shortcuts in your day-to-day work.

Let me digress for a little history lesson. We're talking way back in the days of Excel 2003 and earlier. Did you ever notice those single underlined letters underneath menu items in Windows? Such as File, Edit, View, and so on. Here's a screenshot from Excel 2003:

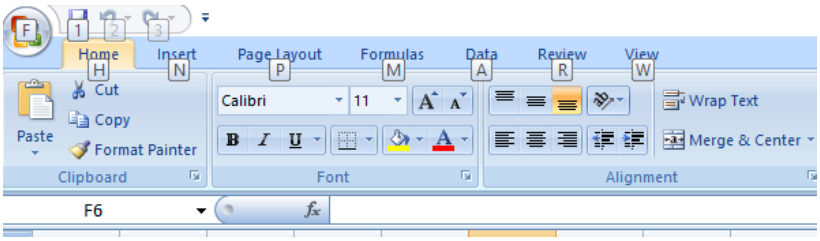
The underlined letters are your [ALT] shortcuts to open the menu. So in Excel 2003, you could push [ALT], then [E], to open the Edit menu dropdown. This reveals more menu options with their own underlined letters. Keep hitting the corresponding letters (no need to push [ALT] again), and you're drilling down into the menu structure to bring up any command. For example, [ALT], then [E], then [S] brings up Edit, Paste Special. You can make Excel do just about anything with a couple of keystrokes, without touching the mouse.

[6] "Eight of the Worst Spreadsheet Blunders". CIO. http://www.cio.com/article/2438188/enterprise-software/eight-of-the-worst-spreadsheet-blunders.html

In newer versions of Excel (2007+), these plain dropdown menus have been replaced with the “ribbon” toolbars, and prettier menus. But the old keyboard shortcuts still work! So while no menu comes up when you push [ALT] [E], the full shortcut [ALT] [E] [S] still brings up paste special. The entire old menu structure lives on in the form of [ALT] commands.

For example: there used to be an Insert menu you could open with [ALT] [I]. From there you could choose Insert Rows with [R]. That same sequence of [ALT] [I] [R] still works for inserting rows![7]

Excel 2007 and newer have their own [ALT] shortcuts. When you push ALT, the letters come up indicating what menus you can open.



For example, try [ALT] [H] to go into the Home part of the ribbon. You can access some very useful items from here; below are just a few of them. Keep in mind that you push these keys one after another, not at the same time:

- [ALT] [H] [0] Increase decimal places
- [ALT] [H] [9] Decrease decimal places
- [ALT] [H] [A][R] Align right (yes, that’s a lot of keystrokes, but useful!)
- [ALT] [H] [A][C] Align center
- [ALT] [H] [A][L] Align left

Within dialog boxes that pop up, the underlined letters for ALT shortcuts are visible even in the newest versions of Windows and Excel.[8] Below is a screenshot of the Paste Special box. Instead of clicking the various radio buttons, just use the keyboard shortcuts for the underlined letters. To paste values, press [V] for values. We’ll cover paste special in much more detail shortly.

To summarize: anywhere you can click with the mouse, you can get to with [ALT] and the keyboard. If you’ve memorized the key combinations for common things like paste special, you can execute these commands in a split second with keyboard shortcuts!

The main distinction between [CTRL] and [ALT] shortcuts: with [CTRL] you would hold down keys together. Hold [CTRL] and [B] at the same time to bold. [ALT] is more of an on-off switch. Push [ALT] to turn on “Alt Mode”, then type some sequence of letters to get to your shortcut. You don’t have to do this quickly, just in sequence.

If you push [ALT] accidentally, just hit [ESC] to cancel. [ESC] is usually the first key to try if you’re in the middle of something and don’t like it (closing dialog boxes, exiting a cell without making changes, and

[7] Where did “insert rows” go in the 2007 and newer interface? I have no idea. I guess they want you to right-click on the side of the screen on the row number, then choose insert rows. Too slow! Use [ALT] [I] [R] instead. Insert. Row.

[8] As of Windows 10 and Office 2016; hopefully they never take these [ALT] shortcuts away in future versions!

stopping an [ALT] key sequence). A confident push of [ESC] is usually all you need when Excel is doing something unexpected.

Paste Special

We all know regular copy-pasting. It's very simple. With regular paste, you're pasting all of the properties of the copied cell: formulas, formatting, and any comments and data validation.

With paste special, you can pick and choose. Maybe you want to paste the value of the cell instead of the formula. Or paste the formula of the cell, but not the formatting. Or quickly apply the same formatting to different areas of your sheet by pasting just the formatting. Paste special is versatile.

Advanced Tip

Memorize the most common Paste-Special shortcuts: [ALT] [E] [S], then either [V], [F], or [T].

Your shortcut is either [ALT] [E] [S] or [CTRL] + [ALT] + [V]. The latter only works in Excel 2007 and newer. Either shortcut brings up the Paste Special dialog box.[9]

Paste Special may seem like a lot of keystrokes to remember, but you'll be using it so often it's worth memorizing. You should also memorize the shortcuts for the most common types of paste (with the [ALT] letters underlined, just like Excel does):

- Values (ignore formatting and formulas, just paste hard-coded values)
- Formulas (paste relative references like regular copy-paste, but without formatting)
- Formatting (paste only the formatting, not the content)

To paste special values, your full shortcut is [ALT] [E] [S] [V] [ENTER] or [CTRL] + [ALT] + [V] then [V] then [ENTER].

The less common paste special options are:

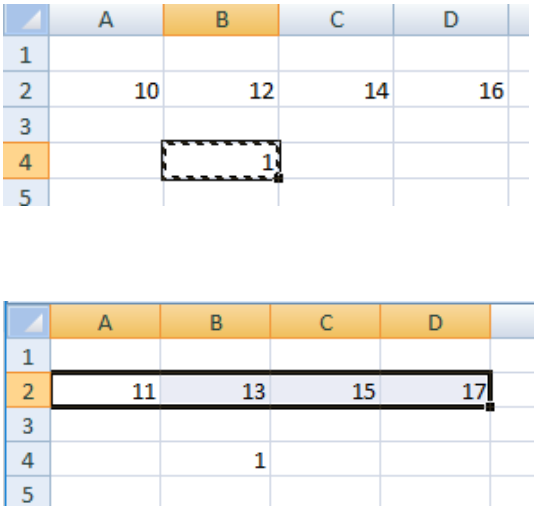
- Comments (just paste the comment attached to the cell. Use [SHIFT] + [F2] to see what I mean by comments)
- Data validation (what types of data are allowed to go into that cell; useful for building user-friendly models)
- Transpose (flip the data 90 degrees. If you copied two columns you'll paste them as two rows. Yes, this works for formulas!)
- Add
- Subtract
- Multiply
- Divide

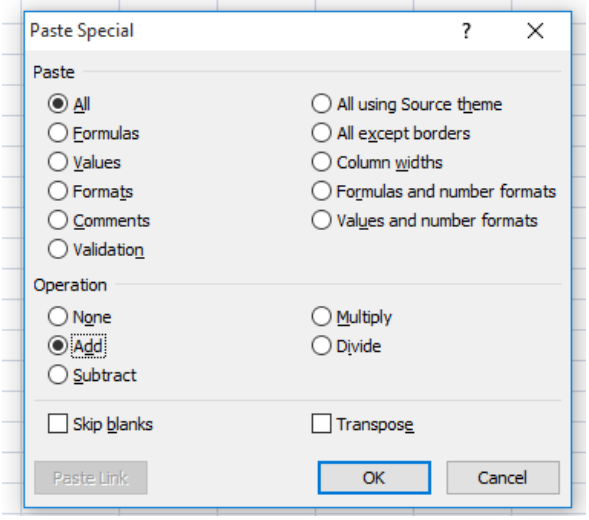
These last four can be useful in a few limited situations. Let's say you have a row of numbers, and you want to add 1 to every number. There are two different ways to do this quickly:

- Insert a row below your numbers. Write the formula to take a cell above and add 1. So B5 would be the formula: `=B4+1`. Drag this formula across to fill the row.[10] Copy the new row, and paste special values ([ALT] [E] [S] [V]) over the original data (to keep just the values, not formulas). Then delete the extra row you inserted.
- Type the number 1 into any cell. Copy this cell, and paste special add over your original row of data ([ALT] [E] [S] [D]). Excel adds 1 to all the values. Now you can clear that extra cell with the 1 in it.

[9] Make sure you've copied or cut something (using [CTRL] + [C] or [CTRL] + [X]) before you try to paste.
[10] Or use [CTRL] + [R] with the keyboard. We'll get to that later in this chapter.

The second way is faster. It's also much more impressive if anyone is watching you do it. You can refer to these screenshots:





You could use paste special divide to divide every number by 100 (converting to percentages, for instance). Just enter 100 into a cell, copy that value, and paste special over the cells you want to divide by 100.

Insert and Delete Cells

Learn the following shortcuts for inserting and deleting cells; you will use them often:

- [ALT] [I] [R] Insert rows
- [ALT] [I] [C] Insert columns
- [ALT] [I] [E] Insert cells (and shift things down or left)
- [CTRL] + [SHIFT] + [=] Insert cells (and shift things down or left)
- [SHIFT] + [SPACE] Select entire row
- [CTRL] + [SPACE] Select entire column
- [CTRL] + [-] Delete selection

[ALT] [I] [R] inserts a row and [ALT] [I] [C] inserts a column. If you want to insert multiple rows, select multiple rows of cells (holding down [SHIFT]), and use [ALT] [I] [R] to insert that many rows. These should be among your favorite shortcuts.

[SHIFT] + [SPACE] selects the entire row(s) you are in. [CTRL] + [SPACE] selects the entire column(s).

Once you have a row or a column selected, use [CTRL] + [-] to delete it.

When deleting a blank row, you'll usually want to make sure it really is blank. Otherwise you might delete something important that is off-screen, and not even know right away. So I recommend using [CTRL] + [RIGHT] and [CTRL] + [LEFT] to navigate along the row. If you bump into any cells along the way, you know they are not empty. Otherwise these keys take you to the very last column (labeled XFD) and back again. Now you're safe to delete the row. The same trick goes for columns with [CTRL] + [DOWN] and [CTRL] + [UP].

You can also insert and delete ranges of cells (versus entire rows and columns). To insert cells:

1. Select a range of cells; for example, 2 x 4 rows and columns
2. Press [ALT] [I] [E] (insert cells)
3. Excel asks if you want to shift cells down or shift cells right (or insert an entire row or column). Use the arrows and the [ENTER] key to make a selection

Inserting cells is a useful way to move things around on a sheet. You might want to make yourself more space for three columns of cells in a few rows, without inserting three full columns. Just select sufficient rows in those three columns, then [ALT] [I] [E], then [ENTER] to choose Shift Cells Right.

[CTRL] + [SHIFT] + [=] does the same thing as [ALT] [I] [E]. You need to use the [=] / [+] key; the [+] on the keypad does not work for this shortcut.

Deleting cells is similar to inserting: use [CTRL] + [-] with the cells selected. Excel asks if you want to shift cells left or up; use [UP]/[DOWN] and [ENTER] to make the selection.

To summarize: use the keyboard to move cells and ranges of cells around on the sheet:

- Use [ARROWS], [SHIFT], and [CTRL] to select things
- Use [CTRL] +[X] and [CTRL] + [V] to cut and paste
- Use the [ALT] shortcuts and [CTRL] + [-] to insert and delete cells

The best way to remember these shortcuts and make them part of your daily work is to **practice whenever you use Excel**. In the next section you'll get a chance to try these out on a fun little exercise.

Practice: Rearranging with the Keyboard

The goal is to rearrange the colored cells to look like the Windows logo (four colored squares with some space in-between) using only the keyboard, no mouse. You'll also need to delete the yellow cells that are in the way. Download the file for this exercise from www.AdvancedExcelBook.com or just set up something similar for yourself to practice on. The next page shows the starting layout; the cells are labeled by their colors and where they belong in the overall diagram:

- Red goes to the top left
- Green goes to the top right
- Blue goes to the bottom left
- Orange goes to the bottom right

To solve this task you need to use keyboard shortcuts to move around the sheet, select cells, copy-paste, delete, and insert rows/columns. All of these tasks should become routine in your daily work, and they require practice!

Working with Formulas

Excel is all about formulas. As an advanced and productive Excel user, you'll want to be comfortable entering and editing formulas as efficiently as possible.

To enter a formula such as `=2+B2+B3` into cell A1, follow these steps:

1. Press [=] to start a formula in cell A1[11]
2. Type `2+`
3. Use the [RIGHT] and [DOWN] arrows to select cell B2
4. Type `+`
5. This time, type `B3` instead of navigating to the cell
6. Press [ENTER]

[11] Some people begin their formulas by typing "+". It's convenient when you use Excel as a calculator, for things like 1+1. There's no = key on the keypad but the + is there, and it happens to work. You could enter +1+1. While this technically works, I consider it a bad habit. When you go back to this cell it contains =+1+1, and the extra plus makes the formula hard to read.

If you follow these steps, you'll end up with the formula you wanted. Notice that you're using two different ways to add a cell reference without the mouse: either navigate to the cell with the keyboard (B2), or just type in the address of the cell (B3).

While navigating to cells for your formulas, you can use all of the keyboard navigation you've learned already!

- Use [ARROWS], with [CTRL] if necessary, to get to individual cells
- Hold down [SHIFT] to select ranges for your formula. Useful for SUM or AVERAGE functions such as `=SUM(B3:B10)`
- Use [CTRL] + [PAGE UP] and [CTRL] + [PAGE DOWN] to go to cells on another sheet
- Use [F3] to add a named reference, or just start typing the name of the reference (we'll talk about these more in Chapter 5)

If this doesn't feel natural yet without the mouse, keep practicing! Try entering the following formulas (starting in A1, then A2, A3, etc) with the keyboard:

- `=C1*2`
- `=B2+B3+B4`
- `=AVERAGE(C2:C5)`
- `=B3+D3-SUM(E3:F5)`
- `=Sheet2!A1`

Note: you need a sheet called Sheet2 in your file to be able to do the last one.

In the case of SUM, Excel has a special auto-sum keyboard shortcut:

1. Use [SHIFT] + [CTRL] + [Arrows] to select the range of cells you want to sum (in a row or a column).
2. Press [ALT] + [=] (holding down [ALT] and the [=] key together).

The next cell (below or to the right) will instantly have the SUM function to add up the selected range! This is a very convenient shortcut for a commonly used task.

Use [F2] to edit formulas with the keyboard. This key jumps into the cell and lets you start typing. It's one of the most important Excel shortcuts, so I'm repeating it from the previous chapter.

Advanced Tip

When editing formulas, the arrow keys move you inside the cell. Press [F2] to let the arrows select other cells instead.

When editing an existing formula, notice that the [LEFT] and [RIGHT] arrows move the cursor within the cell. When you were entering formulas, you were able to reference cells with the arrows. Right now that doesn't work: you are stuck inside the cell. Press [F2] again to switch the arrows back to selecting cells and ranges.

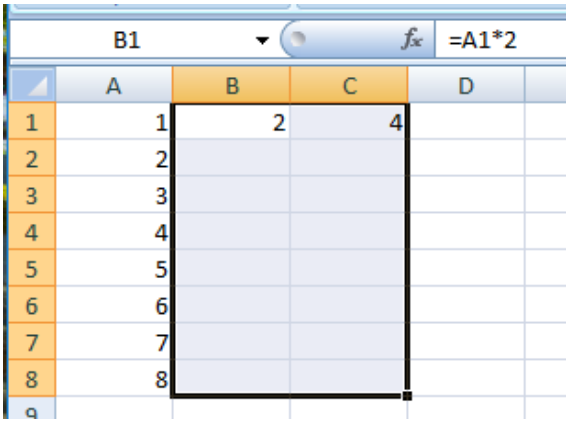
When you're done editing a formula, use [ENTER] to make the change, or [ESC] to cancel and leave the cell unchanged.

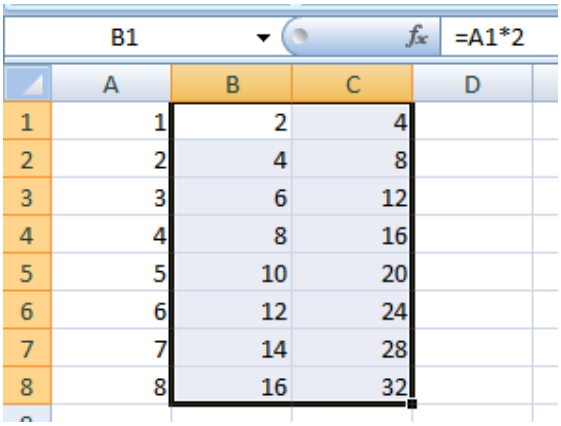
Copying Formulas ("Dragging Down" and "Dragging Across")

In Chapter 2 we looked at dragging down formulas to fill an entire column. With the mouse, you can either drag or double-click the "fill handle" which is the bottom-right corner of the cell.

As an alternative, I mentioned you can use [CTRL] + [D] to fill the formula down. This shortcut works like this: within a selected range, [CTRL] + [D] fills the top row's formula into all of the other rows, in all selected columns.

In the screenshot below, B1 has the formula `=A1*2`. C1 has the formula `=B1*2`. With both of these columns selected, we can use [CTRL] + [D] to fill these formulas for the entire selected range.





The above shortcut has a counterpart: [CTRL] + [R] is the shortcut to fill across to the Right (with R) instead of Down (with D). Use it to fill the left-most formula across one or more rows.

Note that both of these shortcuts copy formulas and formatting. If you want to just copy formulas, you'll need to copy the cell with the formula, select the cell you want to copy it to, then use Paste Special, Formulas: [ALT] [E] [S] [F]

To recap:

- [CTRL] + [D] fills down the formula in the top cell of the selection to the rest of the selection.
- [CTRL] + [R] fills right the formula in the left cell of the selection to the rest of the selection.

Anchoring Cells

When you copy a formula down, and the formula includes a cell reference (for example, `=A1*2`), Excel treats this as a relative reference. The next row is `=A2*2`, the next is `=A3*2`, and so on. Most of the time that's exactly what you want. But what if you want the reference to A1 to stay the same? You can "anchor" that cell in your formulas.

Use [F4] to toggle anchoring while you're editing a formula. Place your cursor next to a cell reference such as A1. Then use [F4] to cycle through anchoring. First you'll see A1 turn into $A$1, meaning both rows and columns are anchored. Keep pressing [F4] to get A$1, then $A1, then back to regular A1. The $ sign precedes the column or row that is anchored.

Experienced Excel users know how to use anchoring to their advantage. If your rows are anchored (A$1) then dragging or copying formulas down keeps the row reference unchanged (row 1). If your columns are anchored ($A1) then dragging formulas across keeps the column reference unchanged (column A).

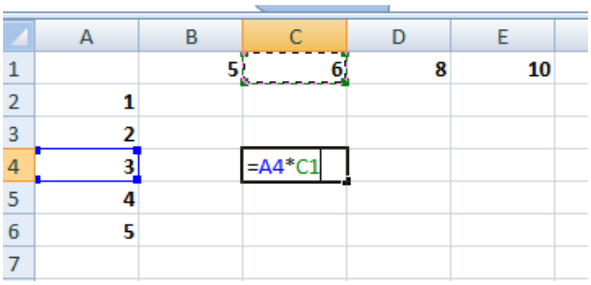
Don't ignore the two "partial" anchors. Sometimes you'll want to deliberately anchor rows but let columns change; or vice versa. Think strategically when you're entering your formulas in terms of anchoring!

Below are two examples showing anchoring in action.

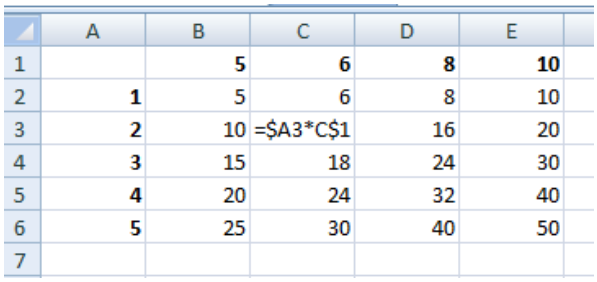
Example 1: multiplication tables. Fill out a two-dimensional table, with constant numbers in the top row and the left column. Each cell of the table is the multiple of the cell's top row value and left column value.

Filling out this table without anchoring would be a lot of work! Every formula is slightly different, but here's the trick: anchor just the columns of the first multiplication term, and anchor just the rows of the second term. So the formula in any particular cell looks like `=$A4*C$1`.

Once you have the formula correctly set up, just copy-paste it to the entire table to get the full table results.

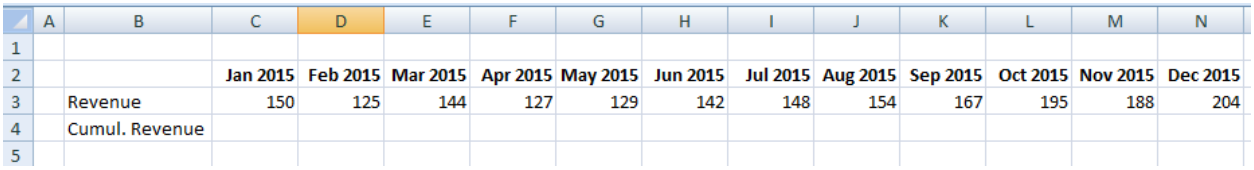


	A	B	C	D	E
1		5	6	8	10
2	1				
3	2				
4	3		=A4*C1		
5	4				
6	5				
7					



	A	B	C	D	E
1		5	6	8	10
2	1	5	6	8	10
3	2	10	=$A3*C$1	16	20
4	3	15	18	24	30
5	4	20	24	32	40
6	5	25	30	40	50
7					

Example 2: cumulative sums. Use anchoring to sum up a range of cells, with the first row of that range staying constant. For example, we have monthly revenue numbers laid out in a row; we want to calculate a running total of these revenues.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2			Jan 2015	Feb 2015	Mar 2015	Apr 2015	May 2015	Jun 2015	Jul 2015	Aug 2015	Sep 2015	Oct 2015	Nov 2015	Dec 2015
3		Revenue	150	125	144	127	129	142	148	154	167	195	188	204
4		Cumul. Revenue												
5														

To do this, enter `=SUM(C3:C3)` into cell C4 which is the first cumulative revenue. Notice that Excel automatically enters the last C3 after you type the colon (:) which is a nice time saver. Now, go back into the formula with [F2] and anchor the first C3 to make sure it doesn't move. You'll end up with the formula `=SUM($C$3:C3)` which is exactly what you need!

Now use [CTRL] + [R] or copy-paste to fill out the rest of the row. Notice that the sum always starts in cell C3, but automatically moves across the columns. Perfect for cumulative sums!

Trace Dependents, Trace Precedents

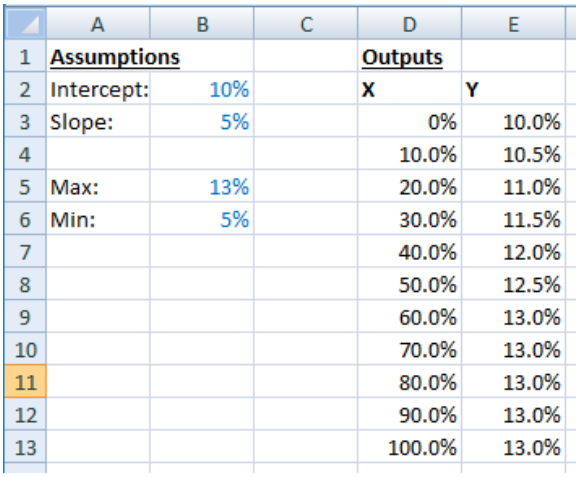
Trace Dependents and Trace Precedents are both incredibly useful, and should become a regular part of your Excel toolkit. If you select a cell on your sheet, **Trace Dependents tells you what other cells are dependent on your cell**. If you've ever tried to figure out how a model works, or what someone else's Excel sheet is doing, this is the way to find out!

> **Advanced Tip**
>
> Use Trace Dependents and Trace Precedents to better understand how your sheet's formulas work.

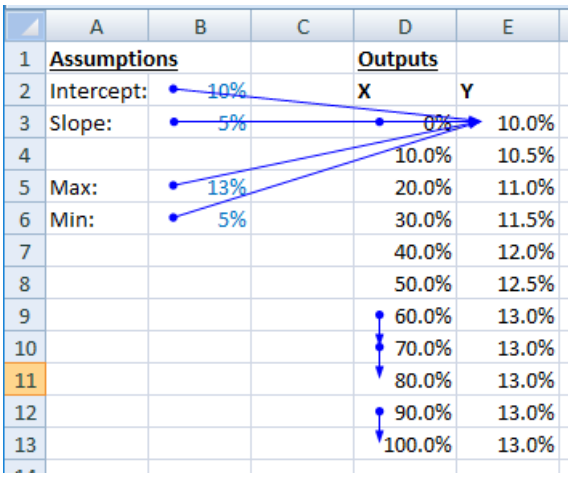
To try it out, select a cell that has a dependent (meaning it's used in another formula somewhere). Next, click Trace Dependents; in Excel 2007 and newer you'll find this under the Formulas ribbon. In Excel 2003, it's under Tools, Auditing. Some blue arrows appear, pointing from your cell to other cells where it is used!

Trace Precedents works the opposite way: if you have a cell with a formula in it, Trace Precedents places arrows pointing from those cells to your selected cell.

Here's a very simple spreadsheet with some assumption cells driving outputs in a column. Use trace precedents on cell E3 to show all five input cells driving this result (including D3). The screenshot also shows trace precedents of D10, D11, and D13 (each precedent is the cell above it).



	A	B	C	D	E
1	**Assumptions**			**Outputs**	
2	Intercept:	10%		X	Y
3	Slope:	5%		0%	10.0%
4				10.0%	10.5%
5	Max:	13%		20.0%	11.0%
6	Min:	5%		30.0%	11.5%
7				40.0%	12.0%
8				50.0%	12.5%
9				60.0%	13.0%
10				70.0%	13.0%
11				80.0%	13.0%
12				90.0%	13.0%
13				100.0%	13.0%

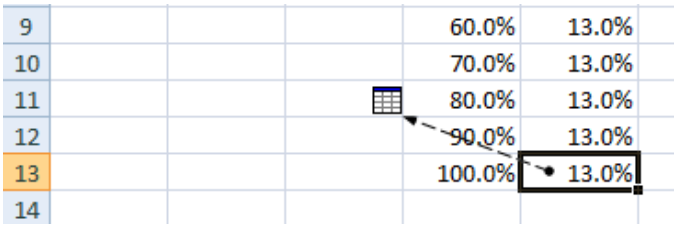


	A	B	C	D	E
1	**Assumptions**			**Outputs**	
2	Intercept:	10%		X	Y
3	Slope:	5%		0%	10.0%
4				10.0%	10.5%
5	Max:	13%		20.0%	11.0%
6	Min:	5%		30.0%	11.5%
7				40.0%	12.0%
8				50.0%	12.5%
9				60.0%	13.0%
10				70.0%	13.0%
11				80.0%	13.0%
12				90.0%	13.0%
13				100.0%	13.0%

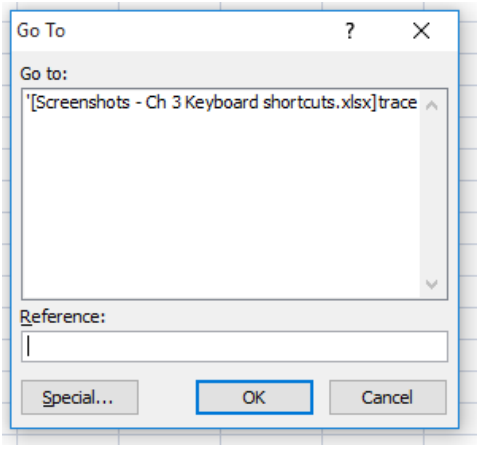
If there are no precedents or no dependents to the cell you're trying to trace, Excel pops up a message: "The Trace Dependents command found no formulas that refer to the active cell." Hit [ENTER] to close this message and move on.

If you end up with too many arrows, get rid of them with Remove Arrows in the Formulas ribbon.

Note that sometimes your dependents or precedents are on another sheet. In that case instead of a blue arrow, you'll see a dotted black arrow pointing to a small box that looks like an Excel sheet. It's not obvious, but you need to **double-click the dotted arrow to see the list of precedent or dependent cells on other sheets**. It brings up the Go To dialog box with shortcuts to the dependent or precedent cells.



9			60.0%	13.0%
10			70.0%	13.0%
11			80.0%	13.0%
12			90.0%	13.0%
13			100.0%	13.0%
14				



It's time to learn the keyboard shortcuts to make these great tools easier to access:

- [ALT] [T] [U] [T] Trace Precedents
- [ALT] [T] [U] [D] Trace Dependents
- [ALT] [T] [U] [A] Remove Trace Arrows

Besides tracing dependents and precedents, you can also jump to the actual cells that are precedents and dependents. These two shortcuts accomplish that:

- [CTRL] + [Go to Precedents
- [CTRL] +] Go to Dependents

Go to precedents is simple: it takes you to one of the precedent cells for the formula. In case there are multiple such cells on the same sheet, the rest of these cells become highlighted and you can cycle through them with [TAB]. You are now cycling through all the cells that would have blue precedent arrows shown with Trace Precedents.

If you find precedents on another sheet, [CTRL] + [takes you to that cell. However, [TAB] no longer works for cycling through precedents across sheets; that's only if everything's on the same sheet.

"Go to dependents" works similarly to "go to precedents", as long as everything is on the same sheet. You jump to one of the dependent cells, and you can cycle through the other dependents on the same sheet with [TAB].

One very important caveat: **Go to Dependents does not work across sheets!** I don't know why; this is a potentially significant Excel flaw. If your cell has dependents on other sheets, Trace Dependents shows them with the dotted arrow. But Go to Dependents says "No cells were found" – which is not true! So be careful, and remember to use Trace Dependents/Precedents if you want to be sure of all your dependents.

Basic Formatting with Keyboard

Formatting cells is generally easier with the mouse, especially tasks like picking colors, applying borders, changing fonts, and so on. But there are some routine formatting tasks that are easier (and more impressive!) with keyboard shortcuts. These are the ones definitely worth learning:

- [CTRL]+[SHIFT]+[1] Number format with commas and two decimals
- [CTRL]+[SHIFT]+[4] Currency format with commas and two decimals
- [CTRL]+[SHIFT]+[5] Percentage format
- [ALT] [H] [0] Increase decimal places
- [ALT] [H] [9] Decrease decimal places
- [CTRL]+[SHIFT]+[7] Apply outside border to selection
- [CTRL]+[SHIFT]+[-] Delete borders from selection
- [ALT] [O] [C] [A] Auto-fit column width
- [CTRL]+[1] Open cell formatting box

[CTRL]+[SHIFT]+[1], [CTRL]+[SHIFT]+[4], and [CTRL]+[SHIFT]+[5] apply various number formatting. 1 is for numeric with comma separators and two decimals, 4 for currency with two decimals, and 5 for percentage.[12] These are easy to remember because of the $ sign on the keyboard above the 4 and the % sign above the 5. Use these shortcuts to quickly get your numbers formatted reasonably nicely!

Now that you have a quick way to turn a number like 54567 into 54,567.00 using [CTRL]+[SHIFT]+[1], you'll want to easily be able to increase or decrease decimal places.

In Excel 2007 and newer, these are found on the Home ribbon, which you get to with [ALT] [H]. Then use [0] or [9] to increase or decrease decimal places.[13]

You can use the keyboard for cleaning up cell borders. [CTRL]+[SHIFT]+[7] **applies an outside border to your selection**, around one or more cells. This shortcut is not as flexible as selecting different types of

[12] There's also [CTRL]+[SHIFT]+[6] for scientific notation, which is rarely used but there you go.
[13] It can be tricky to remember which is which, so I think of it this way: use [0] to add another 0 to the end of your number.

borders by hand, but it can be quite useful. Since it's so easy to apply outside borders with the keyboard, you'll probably find yourself using it quite often!

[CTRL]+[SHIFT]+[-] **removes all borders** from the selection, including inner and outer borders. It's very useful for cleaning up some stray borders you don't want anymore. Especially after copy-pasting some cells with borders, you might end up with a mess like the one below; it's easily fixed by removing all borders and adding back a clean outside border.

	A	B	C	D	E
1					
2		Heading	Heading	Heading	Heading
3		-4	0	1	
4		-3	1	2	
5		-2	2	3	
6		-1	0	4	
7		0	0	5	
8		1	0	6	
9		2	0	7	
10					

	A	B	C	D	E
1					
2		Heading	Heading	Heading	Heading
3		-4	0	1	
4		-3	1	2	
5		-2	2	3	
6		-1	0	4	
7		0	0	5	
8		1	0	6	
9		2	0	7	
10					

[CTRL] + [1] opens the **Format cells** dialog box. This box lets you change number formatting, alignment, font, and borders. You'll want to use the mouse to choose your Format Cells options, but this easy keyboard shortcut will save you from having to right-click a cell and find the "Format Cells" button.

Advanced Tip

Use [ALT] [O] [C] [A] to auto-fit column width for the selected cell(s). This is a frequently needed shortcut!

Changing column widths is extremely convenient with the keyboard. This will be one of your favorites. Select a cell that doesn't quite fit into the current column width. Press [ALT] [O] [C] [A] to resize the column to exactly fit the selected cell.

Want to resize to fit the width of the entire column? Select the column with [CTRL] + [SPACE] and use [ALT] [O] [C] [A] to auto-fit the selection width (the shortcut itself stands for Format, Columns, Autofit Width).

Grouping and Hiding Columns and Rows

Sometimes you want to hide rows and columns to get some interim calculations off the screen. These are the keyboard shortcuts:

- [CTRL]+[9] Hide selected row(s)
- [CTRL]+[SHIFT]+[9] Un-hide rows between the selected rows
- [CTRL]+[0] Un-hide selected column(s)
- [CTRL]+[SHIFT]+[0] Un-hide columns between the selected columns

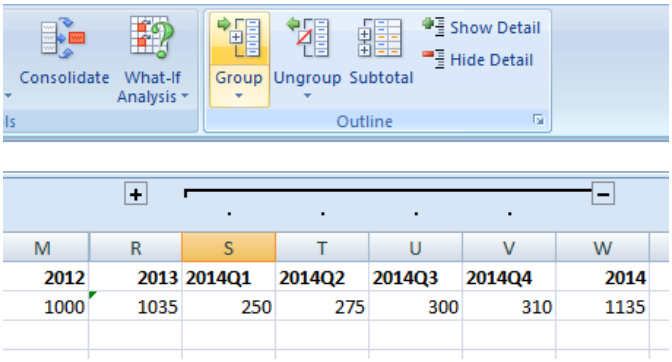
Note: you would expect this last one (un-hide columns) to work the same way as un-hide rows. However, in Excel 2007 and newer, this shortcut doesn't work by default! Don't worry, it can be turned on. Meet me in the footnotes for the step-by-step solution.[14]

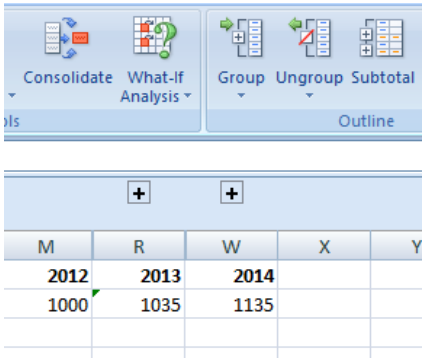
[14] Windows has a special usage for [CTRL]+[SHIFT]+[0] which conflicts with most versions of Excel. Here's how to fix that: go to Control Panel and find Change Keyboards. Go to the Advanced Key Settings tab, then choose Between Input Languages and click Change Key Sequence. Finally, choose Not Assigned for Switch Keyboard Layout.

So that covers hiding and un-hiding. Now I recommend you **completely forget about hiding and un-hiding**. The risk of accidentally making a huge mistake because of hidden rows/columns outweighs any benefits.

> The example of a law associate working on the bankruptcy sale of Lehman Brothers to Barclays in 2008 should serve as a cautionary tale about hiding rows:
>
> As reported by AboveTheLaw.com[15] (and subsequently publicized on dozens of major news sites), a junior associate at a law firm was working late into the night on an Excel sheet. The file listed contracts that Barclays was to purchase out of bankruptcy – we're talking about expensive financial obligations. His fateful mistake was this: he re-sized row heights across the entire sheet to improve the formatting for printing. In the process he un-hid a large number of rows, representing contracts that the bank did not intend to purchase.
>
> In the associate's own words: "Some of the rows of the original Excel spreadsheet were spaced too close together or too far apart, making it difficult to read when printed or converted to PDF format. I therefore globally re-sized all the rows in the document to make it easier to read when printed or converted to PDF format."[16] The whole situation ended up in bankruptcy court and all over the internet.

In this case, the original Excel sheet creator deserves a fair amount of blame. Instead of hiding those rows, he or she should have used the proper user-friendly solution: grouping. **Grouping rows or columns** puts a little + / - toggle box around grouped columns or rows. You can still hide rows and columns from view, but the prominent box at the margins makes it clear that something is there.





To group rows or columns:

1. Select the rows/columns
2. Click Data ribbon, then Group

Excel connects grouped columns using a line at the margins. The little - and + boxes let you hide or unhide these grouped columns with one click! You also can ungroup columns under the Data menu.

The following are your keyboard shortcuts for grouping and un-grouping:

---

[15] "The Case For Sleep: What Happens In Excel After Dark". Above The Law. http://abovethelaw.com/2008/10/the-case-for-sleep-what-happens-in-excel-after-dark/

[16] "Hidden spreadsheet rows hit Barclays with toxic Lehman contracts". AccountingWeb. http://www.accountingweb.co.uk/topic/excel/hidden-spreadsheet-rows-hit-barclays-toxic-lehman-contracts

- [ALT]+[SHIFT]+[LEFT] Group rows or columns (Excel 2007 or newer)[17]
- [ALT] [D] [G] [G] Group rows or columns (all versions)
- [ALT]+[SHIFT]+[RIGHT] Ungroup rows or columns (Excel 2007 or newer)
- [ALT] [D] [G] [U] Ungroup rows or columns (all versions)
- [ALT] [D] [G] [H] Hide grouped columns (shortcut for the - box at the margin)
- [ALT] [D] [G] [H] Show grouped columns (shortcut for the + box at the margin)

Other Shortcuts

Below are a few more shortcuts you may find useful once in a while; don't worry about these until you become a regular user of the other shortcuts in this chapter:

- [F7] Spell check
- [SHIFT] + [F2] Add or edit comment
- [CTRL] + [K] Insert hyperlink
- [CTRL] + [T] Create table from selection
- [CTRL] + [~] View formulas in all cells (try it!); toggle it (press again) to return to normal
- [ALT] [M] [M] [D] Define named range (more on names in Chapter 5)
- [ALT] + [DOWN] Open dropdown cell (if you have data validation dropdowns)
- [ALT] + [;] Select only visible cells (in case you have hidden/filtered cells)

[17] Be careful with [ALT] + [SHIFT] + [RIGHT] or [LEFT]; if you accidentally use [CTRL] instead of [SHIFT], it's a Windows shortcut for flipping your screen 90 degrees left or right. Use [ALT] + [CTRL] + [UP] to get back to normal.

4: Functions

Excel has over 350 functions; as an Advanced Excel user, you'll become familiar with the 50 or so most commonly used ones. As with keyboard shortcuts, the goal is not to memorize the most formulas! Focus instead on expanding your Excel vocabulary with the ones most helpful to your work.

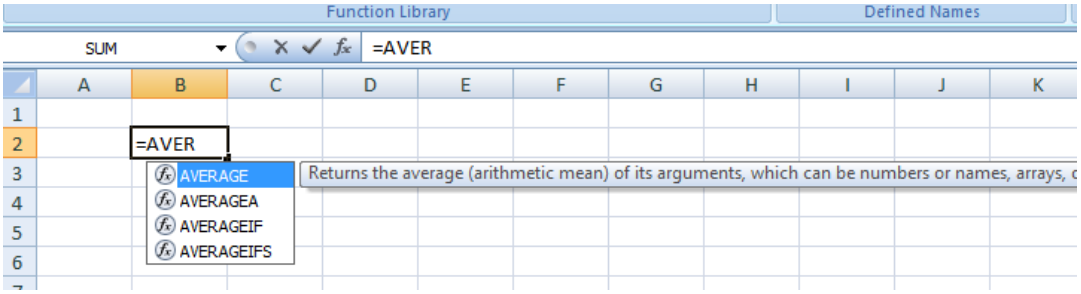
The functions we're about to cover are available in a convenient one-page reference sheet on the book's website at www.AdvancedExcelBook.com, similar to the keyboard shortcuts sheet. Keep one of these near your desk for easy reference. We have a long list of functions to cover, and you'll want to cut straight to the most important ones. If you're familiar with the basics such as SUM, AVERAGE, IF, AND, OR, I recommend you start with the following two sections:

- SUMPRODUCT (used for weighted averages)
- VLOOKUP (a very handy tool with many applications)

If most of these are new to you, no problem! Start from the beginning, and practice your new skills.

There's a good chance you'll one day need a function not covered in this chapter. The first thing I do when I don't know the function I need, is Google it! Search for things like "Excel function find and replace text" or "Excel function probability distributions," and you should find a satisfactory answer. It's rare you come across a question that no one has asked before!

Advanced use of Functions starts with entering them efficiently: meaning, you type = and then the name of the function. Excel shows you an overlay of possible functions, and even describes what they do in a little overlay. Avoid the Insert Function button in the Ribbon and the little fx button next to the formula bar. If anyone is watching, this is a dead giveaway that you don't know what you're doing![18]



[18] If you don't remember the function you need, use my Advanced Excel functions reference sheet, or just Google it.

You can speed up function entry by typing just the first few letters followed by [TAB]. Excel will finish the function name and open the parenthesis. In the screenshot above, type `=AVER` then [TAB] to get `=AVERAGE(`

Advanced Tip

To quickly enter a function, press [=] and start typing the function's name; then press [TAB] to auto-complete. The automatic overlays tell you what the function does and what inputs it needs.

Another great built-in feature: once you type the open parenthesis to open a function, Excel gives you another overlay explaining the arguments that go inside the parenthesis.

For the AVERAGE function, the overlay says “number1, [number2], …” This means it’s looking for the first number to include in your average; the second argument is in brackets because it’s an optional input.

Logic functions: IF, AND, OR, NOT, MAX, MIN

We’ll start with the basic and very versatile IF function. As with any function, start typing `=IF(` in a new cell and Excel brings up a little overlay. Here’s a quick overview:

IF: checks whether “logical test” expression is TRUE or FALSE and returns the corresponding result

logical test	a formula whose result is TRUE or FALSE
value if true	what to return if **logical test** is TRUE
[value if false]	what to return if **logical test** is FALSE; optional but highly recommended

```
=IF( A12 > B12, A2, 0)
```

The Value If False is optional, hence the brackets.[19] A few more notes about the IF function:

- The logical test is usually an expression such as A1 > 0. It can also just be a cell input such as A3, in which case TRUE or FALSE depends on the value of A3. The only cell values that return FALSE are: 0, FALSE, and a blank cell. All other values of A3 (like TRUE, 1, -98.5, or “HELLO”) return TRUE.
- `=IF(-1,1)` is a perfectly valid IF function. It returns 1. That’s because the value -1 counts as TRUE in the language of Excel.
- If you want to output some text as the value if true or false, put it in quotes. Like this: `=IF(A1=1,"one","other")`
- Sometimes you just want value if true to be TRUE, and value if false to be FALSE. Then you could write this formula: =`IF(A1=A2, TRUE, FALSE)`. You can also write it like this: `=A1=A2`

You should use IF functions as much as possible to avoid manual calculations. Don’t even think about manually counting or adding up cell values in Excel! Use the IF function to narrow things down to what you need, and use SUM, COUNT, or similar aggregation functions on the filtered results.

For example: if you have some numbers in column A, and you only want to sum up the positive ones, you can use `=IF(A1>0, A1, "")` in column B. The last argument consists of two quotation marks, which results in a blank cell. Then use SUM on the values of column B. Later we’ll learn an even better way to do this using SUMIF.

[19] Excel actually has a mistake here because the hover bar shows the Value If True in brackets as well, suggesting it is optional. The IF function gives an error if you are missing the second argument; it’s not optional.

The IF function by itself is very simple. It gets more interesting once you add multiple conditions with AND, OR, NOT, as well as nesting multiple IF functions.

Here is the overview of AND, OR, and NOT, which are all very useful when combined with IF:

AND: returns TRUE if *all* of its inputs are true

```
=AND(A2, B3=B4, C6>min_value)
```

OR: returns TRUE if *at least one* of its inputs is true

```
=OR(A2, B3=B4, C6>min_value)
```

NOT: takes a single input and returns the opposite. Returns TRUE if input is FALSE, and returns FALSE if the input is TRUE

```
=NOT(A1>0)
```

As an example of AND with an IF statement, we can check if both cell B1 and B2 have the value 1. Return the text “both” or “not both” depending on the result. Use this formula:

```
=IF(AND(B1=1,B2=1),"both", "not both")
```

Notice that the following is not valid in Excel: `=IF(B1=1 and B2=1,"yes","no")`. Most programming languages allow this use of “and” and “or” between expressions, but Excel strictly requires you to use AND() and OR() as functions, with arguments inside the parentheses, separated by commas.

To check if either cell C1 or C2 have value 2, use this: `=IF(OR(C1=2,C2=2),"either","neither")`

Here is an example of NOT inside of IF: `=IF(NOT(2>1),"apples","oranges")`. This always returns “oranges”. The 2>1 is TRUE, which flips to FALSE and gets passed to the IF logical test.

There’s a special case of OR (and AND) that makes your life easier (and your formulas more readable). Let’s test if cell A1 has any of the following values: “CA”, “AZ”, “NV”, “NM” (state codes). You could do the following:

```
=IF( OR(A1="CA", A1="AZ", A1="NV", A1="NM"), "yes", "no")
```

This is kind of a long formula to type (and a big pain to change from A1 to another cell!); we prefer to keep our code short and easy to read. So, here’s the alternative (notice where the parentheses versus curly brackets go):

```
=IF(OR(A1={"CA","AZ","NV","NM"}), "yes", "no")
```

Some special cases of IF are better handled with MAX and MIN functions instead. For example, let’s say you are looking at A1-B1-C1, and want to return A1-B1-C1 if it is positive or zero, but return zero if A1-B1-C1 is negative.

You could do this with IF: `=IF(A1-B1-C1<0, 0, A1-B1-C1)`

However, this way is easier to understand and faster to type: `=MAX(0, A1-B1-C1)`

Golden Rule #4 of Advanced Excel Modeling (which we’ll cover in Chapter 7) is to simplify and avoid repeating the same code. The MAX formula above satisfies this rule very nicely.

MAX: returns the largest of all input values

```
=MAX(0, A1-B1)
```

MIN: returns the smallest of all input values

```
=MIN(100%, B2*5, B3)
```

A quick note about MAX and MIN: these formulas can take any inputs, including cell values, constants, calculations, and even logical expressions. But be careful about using any logical expressions as inputs into MAX/MIN! For instance, what do you think is the result of this? `=MAX(A1=10,0.5)`

It depends on the value of A1:

- If A1 is actually equal to 10, the first expression is TRUE which in Excel's world has a numeric value of 1. The maximum of 1 or 0.5 is 1, so the result is 1.
- If A1 is any other value than 10, including TRUE or a blank cell, `A1=10` is FALSE. In Excel world FALSE has a numeric value of 0, so the result is 0.5.

While that above behavior may be exactly what you want, it's not at all predictable! Avoid these kinds of constructs as much as possible because they will be confusing to others.

Math Functions

Excel is a handy calculator for doing calculations such as adding, dividing, etc. It's even handier when performing calculations on data.

Sums, Averages, Counting

Let's start with the most basic calculation: SUM. Obviously, this sums up every value referenced inside the function. You can list individual values with commas such as `=SUM(A1,A2,A5)`. More often, you'll reference ranges such as `=SUM(A1:A5)`. You can also put mixed references such as `=SUM(1,input_value,A1:A5)`. It'll just sum up everything, including the contents of the named reference called input_value.

These are the basic aggregation functions:

SUM: sums up all values referenced

AVERAGE: average of all values referenced

MEDIAN: median (middle value) of all values referenced

COUNT: counts the number of cells referenced that contain a number

COUNTA: counts the number of cells that contain any value (not blanks)

COUNTBLANK: counts the number of cells referenced that are blank cells

It's important to remember the difference between COUNT and COUNTA. I think the "A" stands for "All"; COUNTA counts up all the cells in the range that have some value, including text or TRUE/FALSE. Regular COUNT only counts up the numeric values.

Quick detour: there is a function called AVERAGEA, but don't even think about using it! Here's why: regular AVERAGE takes the average of just the numeric values that you pass to it. But AVERAGEA also includes text and TRUE/FALSE inputs in the average calculation! Specifically: any text and FALSE count as value 0

for the average; TRUE counts as a 1; blanks are ignored. That's pretty strange behavior and not worth using.

When working with data, you'll often want to sum up not an entire column of values, but just ones that meet certain conditions. For example: sum up values in a given category, or sum up only positive values in a range. Both cases use the SUMIF function.

SUMIF: sums up only values that meet a particular condition. The condition can be on a different range of values than the values to be summed.

range	the range of value to test against the **criteria** if you don't specify the **sum range**, the **range** values will be summed
criteria	a logical test on values in the **range**
[sum range]	optional: sum the values in the **sum range** instead of the **range**

```
=SUMIF(B2:B10, ">0")
=SUMIF(A2:A10, "Blue", B2:B10)
```

To illustrate SUMIF we'll use the screenshot below. We have a simple dataset in cells A1:C10 and some calculations in B12:B15, B17, and B19.

	A	B	C	D	E	F	G	H
1	Category	Values	Condition					
2	red	-1	TRUE		>0			
3	red	0	TRUE					
4	red	2	TRUE					
5	blue	3	TRUE					
6	blue	-2	TRUE					
7	blue	-1	TRUE					
8	green	0	TRUE					
9	red	1	TRUE					
10	red	2	FALSE					
11								
12		4	=SUM(B2:B10)					
13		8	=SUMIF(B2:B10,">0")					
14		8	=SUMIF(B2:B10, E2)					
15		4	=SUMIF(A2:A10,"red",B2:B10)					
16								
17		3	=SUMIFS(B2:B10, A2:A10, "red", B2:B10, ">0", C2:C10, TRUE)					
18								
19		0.8	=AVERAGEIF(A2:A10,"red",B2:B10)					
20								

The value in B12:B19 is the result of running the corresponding formula that is shown in C12:C19.

Note: to make the actual formulas come up in column C (not the results of the formulas), either set the number format of these cells to Text instead of General in the Home ribbon; or type a single apostrophe character (') before the equals sign.

Advanced Tip

Know when to use quotes and when not to use them with conditional functions such as SUMIF. When in doubt, test it both ways and see!

The first formula is just a simple SUM of all the values, with the result of 4.

The second formula sums up only those values in B2:B10 that meet the condition: “>0”. Those are just the positive values, 2+3+1+2 = 8. The quotation marks are required here! Think of the criteria input as a string value. If you type it into the formula, it needs to be inside quotes.

The third formula is an alternative to the second one. You can put your criteria inside another cell, and refer to it. Just enter >0 in cell E2, without quotes in this case. Then B13 can be:

```
=SUMIF(B2:B10, E2)
```

The fourth formula shows how to sum up values in the red category. We need to do a logical test on the range A2:A10, but sum up values in range B2:B10. In this case A2:A10 is the Range and B2:B10 is the Sum Range. The formula is:

```
=SUMIF(A2:A10,"red",B2:B10)
```

The fifth formula in the screenshot uses the SUMIFS function (that’s with an extra S). SUMIFS lets you run an unlimited number of criteria on various ranges. Then it adds up whatever Sum Range meets all criteria.

SUMIFS: sums up values that meet multiple conditions on multiple ranges. Allows an unlimited number of criteria and criteria ranges.

sum range	the range of values to sum, if all sets of criteria are met
criteria range 1	apply **criteria 1** to this range
criteria 1	logical test on the values in **criteria range 1**
[criteria range 2]	apply **criteria 2** to this range; optional
[criteria 2]	logical test on the values in **criteria range 2**; optional

```
=SUMIFS(B2:B10, B2:B10, "<0", C2:C10, TRUE )
```

The example in the screenshot involves three different criteria: category (column A) is red, values in column B are greater than zero, where column C is TRUE. The values to sum are in column B. The formula is:

```
=SUMIFS(B2:B10, A2:A10, "red", B2:B10, ">0", C2:C10, TRUE)
```

Note: the inputs for these functions may be hard to remember, but Excel is there to help. As long as you remember the name of the function (SUMIF, SUMIFS), just start entering it and open parentheses. Use Excel’s overlay hint to see what arguments are needed in what order.

Similar to SUMIF and SUMIFS, you can conditionally take averages and count cells with numeric values, using AVERAGEIF and COUNTIF, AVERAGEIFS and COUNTIFS. The last formula in the example above shows AVERAGEIF, which is very similar to SUMIF.

AVERAGEIF: take average of values that meet a particular condition

AVERAGEIFS: take average of values that meet multiple conditions on multiple ranges

COUNTIF: count cells with numeric values that meet a particular condition

COUNTIFS: count cells with numeric values that meet multiple conditions on multiple ranges

AVERAGEIF in particular has a very practical use: taking the average of values that are non-zero. I've run into this problem personally many times: my data has zero values that I want to ignore (treat as blanks), and I need the average of the nonzero values. Use this formula:

```
=AVERAGEIF(A2:A100, "<>0")
```

A quick note about COUNTIFS: it's slightly different from SUMIFS and AVERAGEIFS, so don't be surprised. There is no Sum Range or Count Range; you go straight to the criteria and the conditions. For example, here's the formula to count the number of values between 0 and 100, inclusive:

```
=COUNTIFS(A2:A100, ">=0", A2:A100, "<=100")
```

Example: Histograms Using COUNTIF and COUNTIFS

Let's try COUNTIF with a practical example: building histograms. A histogram lets you express how many items fall into given ranges, based on their values. This example works with any set of numeric data and it's a great way to get a sense of magnitudes in a data set: how many are over 100, how many over 10,000, and so on.

In this example we'll build a histogram of insurance claims to get an idea of how often there are large or small claims. The first step is to set up the histogram cutoffs. It doesn't matter what cutoffs you pick right now; you can easily change them later. Since my data is exponential in nature (claims can get very large, but are often small values) my cutoffs are at exponentially larger intervals: 0; 100; 1,000; 5,000; 20,000; 100,000; 1,000,000. Put these cutoffs along a column somewhere to the side.

In the screenshot below, my claim amounts are in column B, and I've added the cutoffs (in ascending order!) in a new table in column E. Ignore anything in column A, it's just extra data that is irrelevant to the histogram.

	A	B	C	D	E	F	G
1	ClaimNum	ClaimAmt					
2	3	1305				Cumul Count	Count
3	7	400			0	132	
4	10	31644			100		
5	11	890			1000		
6	14	10931			5000		
7	20	400			20000		
8	21	700			100000		
9	23	2295			1000000		

I also started filling out column F with a COUNTIF:

```
=COUNTIF(B:B,">="&E3)
```

The first argument should be straightforward: it's the entire B column. With the second argument, we want to generate the string ">=0" but referencing E3 as the 0. Then if you drag that formula down, you'll get >=0,

>=100, >=1000, and so on. In the COUNTIF formula we just replace 0, 100, 1000 with references to E3. (We'll look at string concatenation with the "&" symbol later in this chapter.)

Drag down this cumulative count to fill F3:F9. Now column F shows the cumulative count of items that are greater than or equal to each cutoff. The histogram needs to show the number of items within a particular range, so we need another calculation in G3: `=F3-F4`

This way we just get the incremental count. If you have 132 items >=0 and 114 items >= 100, then you must have 18 items from 0 to 114. The end result looks something like this, with column G containing the histogram outputs we're looking for:

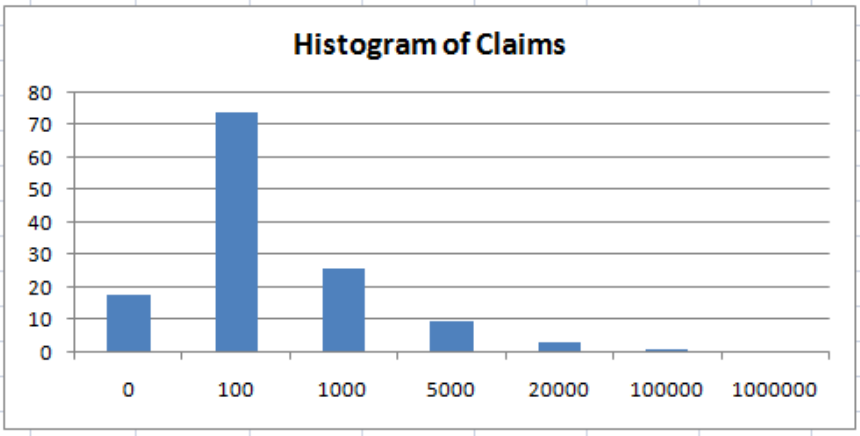
	A	B	C	D	E	F	G
1	**ClaimNum**	**ClaimAmt**					
2	3	1305				**Cumul Count**	**Count**
3	7	400			0	132	18
4	10	31644			100	114	74
5	11	890			1000	40	26
6	14	10931			5000	14	10
7	20	400			20000	4	3
8	21	700			100000	1	1
9	23	2295			1000000	0	0
10	25	400					

There's another way to get the histogram counts directly, without the interim calculations in column F. The solution is to use COUNTIFS in G3 with two conditions: greater than or equal to E3, but less than E4. The formula looks like this:

```
=COUNTIFS(B:B, ">="&E3, B:B, "<"&E4)
```

Note: be careful with the use of equals signs! We want to include values greater than or equal to the lower bounds, but only less than the upper bounds. This way we don't double-count the values that are exactly 100, 1000, etc. So 100 is part of the 100 to 1000 range, not the 0 to 100 range.

Once you have the correct counts, using either the COUNTIF or COUNTIFS method, your final output can be this column chart (the chart is not formatted perfectly, but it gets the job done):



Weighted Average with SUMPRODUCT

Weighted averages are a bit more complex. We'll start with some simple data: three countries, with population and average GDP per capita (per person) for each country. We want the average GDP per capita

for all three countries. It's not the average of column C, it's the average of per capita GDPs (column C) weighted by population (column B).

	A	B	C
1	**Country**	**Population (Mil)**	**GPD Per Capita**
2	United States	319	53,000
3	Canada	35	43,000
4	Mexico	120	16,000
5			

Mathematically, here's what we need to do:

1. Get the total GDP for each country by multiplying population by GDP per capita
2. Sum up all the country GDPs
3. Sum up all the country populations
4. Divide the total GDP by the total population. That's the weighted average GDP per capita.

There's nothing wrong with going through all of these steps manually, creating a new column for total GDP. Your alternative is to use the SUMPRODUCT function!

SUMPRODUCT means you multiply together each set of numbers, then sum up all the results. So it's the sum of the products. In the example above we would use:

```
=SUMPRODUCT(B2:B4,C2:C4)
```

which is the same as writing: `=B2*C2+B3*C3+B4*C4`

SUMPRODUCT: multiply values across arrays (rows or columns), then sum up all the results. Each array must be the same size.

array 1	array of values; each value will be multiplied by the corresponding value in **array 2** (and **array 3**, **array 4**, etc.)
array 2	array of values to multiply with **array 1**
[array 3]	additional array to multiply with **array 1** and **array 2**; optional

```
=SUMPRODUCT(A2:A10, B2:B10)
```

Continuing the example, `=SUMPRODUCT(B2:B4,C2:C4)` gets each population multiplied by each GDP per capita, then sums up the result for each country. This is the total GDP for all three countries combined. Now we need to divide by the total population to get the weighted average GDP per capita.

The final formula is: `=SUMPRODUCT(B2:B4,C2:C4)/SUM(B2:B4)`

The result is 42,894.51.

To recap, weighted average can be calculated by getting the SUMPRODUCT of two arrays: the values and their weights. Then we divide by the sum of the weights to get the weighted average.[20] In almost all cases, the SUM you're dividing by is also one of the columns in the SUMPRODUCT.

[20] In special cases, the weights might be percentages of a total, so they add up to 100%. For example, weights of 50%, 25%, and 25%. In this case you could skip dividing by the SUM of the weights (which is 1), but I still recommend using the full formula every time. It will be correct even if you change the weights to no longer sum to 1.

Special Case: SUMPRODUCT with Conditions

Let's combine the concept of SUMPRODUCT with the SUMIF function that we looked at earlier. We'll do a conditional weighted average: take a weighted average of values, but only using values that belong in a particular category. In this example we'll take the weighted average value of the red category items, using weights in column D.

	A	B	C	D
1	Category	Values	Condition	Weight
2	red	-1	TRUE	1
3	red	0	TRUE	4
4	red	2	TRUE	2
5	blue	3	TRUE	2
6	blue	-2	TRUE	1
7	blue	-1	TRUE	1
8	green	0	TRUE	5
9	red	1	TRUE	1
10	red	2	FALSE	2
11				

We'll use SUMPRODUCT with a little trick. I'll jump straight to the solution:

```
=SUMPRODUCT((A2:A10 = "red")+0,D2:D10,B2:B10)/SUMPRODUCT((A2:A10="red")+0,D2:D10)
```

Advanced Tip

SUMPRODUCT is very powerful! With some trickery, it can even replicate SUMIF or SUMIFS.

This formula looks weird at first, so let's break it down. We're still just getting a weighted average by taking the total of the values and dividing by the total of the weights.

The first condition is new: A2:A10 = "red" tests each item in A2:A10 if it's red. Equal returns TRUE, not equal returns FALSE. But SUMPRODUCT needs numbers instead of TRUE/FALSE, so here's a neat trick: add 0. The value TRUE + 0 returns 1, and the value FALSE + 0 returns 0.

The rest of the first term is the normal weighted average setup: multiply and sum the values and the weights. The only difference is that the rows without the red category get zeroed out.

The second term is almost an exact repeat of the first, with one less thing inside the SUMPRODUCT. It calculates which rows we want to add up for the weights. As with the regular weighted average example, the second term is like the first but excluding the values column.

If you copied my example, you should get 0.8 as the final result. The values in red multiplied by their weights are: -1; 0; 4; 1; 4. These sum to 8. The weights sum to 10. So the weighted average is 0.8.

The above formula could also be written like this:

```
=SUMPRODUCT((A2:A10="red")+0,D2:D10,B2:B10)/SUMIF(A2:A10,"red",D2:D10)
```

The only thing that's changed is how we come up with the weights used. In the first version, we did a SUMPRODUCT with 0's or 1's, and the percentages. Only the red category would get a 1 and keep its weight. In the second version, we used a SUMIF function to sum up weights for the "red" category. These are two ways to accomplish the same thing; do whichever makes more sense to you. I personally like the first version because the two SUMPRODUCT pieces are almost exactly the same.

If this section doesn't make much sense right now, don't worry about it. Come back to this if you ever need to do a weighted average on just a subset of your data. You'll need to remember the trick of adding +0 as a filter.

Special Case: MMULT for Transposed Data

In order to use SUMPRODUCT, the arrays you are multiplying must have the same size and be oriented the same way (both horizontal or both vertical). What if you want to SUMPRODUCT where your values are horizontal, but your weights are vertical? You'll get a #VALUE! error if you try to do that (we'll review formula errors in the next chapter).

Instead, use the MMULT function. This function performs "matrix multiplication" on two arrays. Here's the catch: the first array must be horizontal, and the second array must be vertical. If you flip them, you get the wrong result; you just get the multiple of the first single value of each array.[21]

MMULT: performs matrix multiplication on two arrays. Allows a SUMPRODUCT calculation on a vertical and a horizontal input array.

array 1	array of values; each value is multiplied by the corresponding value in **array 2**; first array must be horizontal
array 2	array of values to multiply with **array 1**; second array must be vertical

```
=MMULT(C2:G2, B3:B7)
```

The best way to really appreciate the power of MMULT is with an example. Make a small array of numbers that is horizontal, four columns wide. Make another that is vertical, four rows tall. Now use MMULT with the two arrays as inputs. Check your work manually to confirm the result is correct; if it isn't, you probably have the arrays in the wrong order.

Rounding

Excel has several functions for rounding.

ROUND: round to X number of digits after decimal to closest value; 0.5 is rounded away from 0

ROUNDUP: round to X number of digits after decimal; always rounding away from 0

ROUNDDOWN: round to X number of digits after decimal; always rounding towards 0

TRUNC: same as ROUNDDOWN

ROUND is your familiar rounding formula. It rounds to the closest value with your specified number of decimals. The most common usage is with 0 decimals, rounding to the nearest integer. But you can round to the nearest tenths or the nearest multiples of 10, 100, 1000, etc. Just put positive or negative values in the second term. Some examples:

- `=ROUND(5238.76, 0)` returns 5239
- `=ROUND(5238.76, 1)` returns 5238.8
- `=ROUND(5238.76, -1)` returns 5240

[21] If you're comfortable with the mathematics of matrix multiplication, you can use inputs that are larger than 1 row/column. For instance, you can do matrix multiplication on a 5x2 by 2x5 set of inputs.

- `=ROUND(5238.76, -3)` returns 5000
- `=ROUND(5238.76, -4)` returns 10000

What if, instead of the nearest integer, or nearest multiple of 10, you want to round to the nearest 5? Or the nearest 3%? Or the nearest B1, whatever the value of cell B1 is?

Here's how each of these would look:

- `=ROUND(A1/5,0) * 5`
- `=ROUND(A1/3%,0) * 3%`
- `=ROUND(A1 / B1,0) * B1`

In each example we're dividing by X and then rounding to an integer, which gets the nearest multiple of X. Then re-multiply by X to get the rounded value of your initial input.

ROUNDUP and ROUNDDOWN are similar but always round away from zero (larger absolute values) for ROUNDUP, and towards zero (smaller absolute values) for ROUNDDOWN:

- `=ROUNDUP(4.7, 0)` returns 5
- `=ROUNDDOWN(4.7, 0)` returns 4

Note: Excel rounds negative numbers as if the negative sign was not there (i.e., based on their absolute value). So `=ROUNDUP(-4.7, 0)` *returns -5. You might argue that -5 is a smaller number than -4.76, so why is this rounding up? That's just the convention for Excel. It is what it is.*

Finally there is TRUNC, which means truncate. It's exactly the same as ROUNDDOWN.

Here's a table that you might find useful for understanding the different forms of rounding. The top numbers are inputs, and the labels on the left are the formulas. The table values are the results for these formulas for each input:

Input Value:	**0**	**0.5**	**0.9**	**1.499**	**-0.5**	**-0.9**	**-1.499**
ROUND to 0 digits:	0	1	1	1	-1	-1	-1
ROUNDUP to 0 digits:	0	1	1	2	-1	-1	-2
ROUNDDOWN to 0 digits:	0	0	0	1	0	0	-1
TRUNC to 0 digits:	0	0	0	1	0	0	-1

In practice, you'll be able to test rounding as you use these formulas in Excel. Just be aware of how negative numbers behave, and how to round to multiples of any number.

There are two other types of rounding: CEILING and FLOOR. They round to total significant digits instead of decimals, which is unlikely to be what you want in real life. Also, in Excel 2007, they both return #NUM! errors. Given these limitations, you'll want to use ROUNDUP and ROUNDDOWN instead.

Additional Math Functions

These functions deal with absolute values and positive or negative signs:

SIGN: returns 1, -1, or 0 based on the sign of the input value (positive, negative, or zero)
ABS: returns absolute value of the input value

These functions should be self-explanatory. You could use IF functions to replace them, but the SIGN or ABS functions make your formulas more compact.

One great trick using the ABS function is to check if two numbers are approximately equal, within a tolerance. Sometimes with complex calculations or SUMs, a cell's value can be 10.0000001 or similar. You want to check if this number (in cell A1) is equal to 10, which it technically isn't. But you can build in a small tolerance of say 0.0001 either up or down from 10. Instead of `=A1=10`, the formula is:

```
=ABS(A1-10) <= 0.0001
```

This way, small differences still return TRUE. The ABS function allows the difference to be positive or negative.

Excel can generate **random numbers** with various distributions. Every time you re-calculate your sheet (press [F9] or change any cell to cause a re-calculation), the values returned by these random functions will change.

An important caveat: some academic papers have shown that Excel does not generate truly random values. The functions below are adequate for casual use and are approximately correct, but I wouldn't risk a billion-dollar project on their effectiveness. Use a different tool if you want perfect random distributions.[22]

RANDBETWEEN: returns a random *integer* between the lower and upper bounds you specify

RAND: returns a random *decimal value* between 0 and 1

Both RANDBETWEEN and RAND return values equally distributed within their bounds. For example `RANDBETWEEN(1,4)` returns only the values 1, 2, 3, or 4, each with 25% probability. RAND returns any decimal between 0 and 1 (not including 1), such as 0.6354225, out to a very large number of digits.

You can manipulate RAND to work similar to RANDBETWEEN; for instance, these two formulas give the same results with the same probabilities:

```
=RANDBETWEEN(1,4)
=ROUNDDOWN(RAND()*5, 0)
```

If you don't believe me, test it out! Make a column of values (around 4000 gets you a good distribution) with the first formula, and another column with the second formula. Then make a table to count up the number of 1, 2, 3, 4 values in each column (use the COUNTIF function we looked at earlier). Each column will have approximately 1000 observations of each value.

Regular RAND and RANDBETWEEN generate uniform distributions; every value has the same probability of coming up. To generate something like a normal distribution, you'll use the NORMINV function with RAND as the input!

NORMINV: returns values within a normal distribution using input value between 0 and 1 for the position along the normal curve; also needs mean and standard deviation of the normal distribution

NORMSINV: same as NORMINV, but for the standard normal distribution with mean of 0 and standard deviation of 1

To use NORMINV and generate actual random numbers with a normal distribution, combine it with RAND like this:

```
=NORMINV(RAND(), 3, 1)
```

[22] "Microsoft Excel's 'Not The Wichmann–Hill' random number generators" B.D. McCullough. Computational Statistics and Data Analysis. http://www.pages.drexel.edu/~bdm25/excel-rng.pdf

This formula generates random numbers with a normal distribution, centered around 3, with a standard deviation of 1.

Nested Functions

A quick detour for nested functions: these are functions inside of other functions. You're most likely to see this with nested IFs, which let you run through a large number of "if this… then this" scenarios all in one cell. Here's an example:

```
=IF(A1=1,"one", IF(A1=2,"two", IF(A1=3,"three", IF(A1=4,"four", ….
```

and so on.

Advanced Tip

Good Excel users test their nested functions to ensure they work correctly; advanced Excel users also break nested formulas apart so they are easy to understand.

In this formula you're stepping through logical conditions. When you hit a true one you get the corresponding result; otherwise you keep going deeper and deeper into layers of IFs. This gets the job done, but entering a long formula like this is a pain. Understanding it and modifying it is an even bigger pain. We'll cover an easy solution to this problem in the next section with VLOOKUPs.

The more complex your formula is, the more important it is to **test with various inputs** to make sure it works correctly. In the example above, you should test out various values of A1 (including strange ones like negative numbers and decimals) to make sure your formula works as intended.

My recommendations when you are building long nested functions:

- Keep track of your parentheses. Fortunately, Excel color-codes your open and close parentheses so you can see which belong together. But it can still be difficult to know which part of which formula you're currently on.
- Use spacing for readability. You can space out your IF function by adding extra spaces before and after parentheses and commas. Like this: `=IF( A1=1, "one", IF(A1=2, "two", IF(`
- Use named references like input_value instead of cells like A1. It's easier to read your formula with named variables instead of letters and numbers. We'll talk about named references in Chapter 5.
- When your formula gets too long and complex, break it apart over multiple cells! Maybe have one half of an IF in one cell, another in a second cell, and use a third cell to choose between the two. It's easier to troubleshoot your formula this way, since you can see the interim outputs in each cell. Take advantage of Excel's key strength: it always shows you the value of each cell calculation.

Excel's help section says you can't nest more than 64 levels of functions in a cell. Sixty-four levels of functions! Please don't even think about using that many; if your particular problem requires more than five levels of functions you're probably doing it wrong. Start by breaking the formula apart into multiple cells, or use a different function entirely.

VLOOKUP

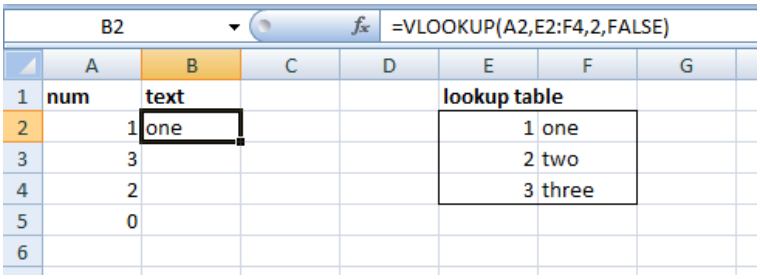
I've seen dozens of people's resumes specifically mentioning VLOOKUP to indicate their advanced Excel knowledge. It also comes up in interviews ("I know Excel very well, for example vlookups"). It's time to master this very versatile function and add something new to your resume.

We'll start with an easy example of nested IF statements from the previous section. You have a cell (A1) that can take the numeric values 1, 2, or 3. In another cell (A2), you want to output the corresponding word

"one", "two", or "three". We'll output "other" for any other input. Here's the solution with a long nested IF statement:

```
=IF(A1=1,"one",IF(A1=2,"two",IF(A2=3,"three","other")))
```

The above is an acceptable, but not a particularly advanced solution. Let's use VLOOKUP instead, which requires a lookup table. Here's the setup:



We have values A2:A5 as given inputs. We're trying to fill out B2:B5 using a lookup formula. We also have a lookup table in range E2:F4. The VLOOKUP function goes in B2, and it's this:

```
=VLOOKUP(A2, E2:F4, 2, FALSE)
```

How to read this? Look up the value in A2, in the first column of table E2:F4, and once you find the row return whatever is in column 2 of the table (F is the 2nd column of the lookup table E2:F4). The last FALSE in the VLOOKUP formula means we're looking for an exact match of the value in A1 (rather than a ranged VLOOKUP, which we'll cover later).

VLOOKUP: looks for a match in the first column of a table, and returns the corresponding value in any other column

lookup value	a single value to find a match for in the **table array**
table array	lookup table range; look for the lookup value in the first column of this table
column number	which column's value to return from the **table array** in the matching row
[range lookup]	FALSE means exact match; TRUE means ranged lookup; defaults to TRUE if omitted

```
=VLOOKUP(A2, Sheet2!A1:D100, 3, FALSE)
```

The last argument, Range Lookup, is technically optional. But I highly recommend including it – TRUE or FALSE makes a big difference! The distinction is this:

- FALSE means an exact lookup. We're looking for a value equal to the lookup value in the first column of the lookup table. If there's no match, the VLOOKUP returns an #N/A error.
- TRUE means a ranged lookup. We're looking for a value in the table that less than or equal to the lookup value. In other words, the lookup table values are lower bounds to your input.

Let's go back to the example above. The formula in B2 works for row 2, but there are a few problems with it. First problem: if you drag it down, the lookup reference moves as well! You'll need to anchor the lookup table reference (press [F4] while editing that part of the formula). This is better:

```
=VLOOKUP(A2, $E$2:$F$4, 2, FALSE)
```

Second problem: drag it down until you get to a mismatching input, the 0. There is no 0 in the lookup table, so the VLOOKUP returns #N/A. Errors are annoying! Fortunately, there are two great little functions to handle errors and output a more reasonable value: IFERROR and IFNA.

> **IFERROR**: if the first input is an error, return the second input; otherwise return the first input
>
> **IFNA:** same as IFERROR, but only for #N/A errors (Excel 2013 and newer only)

IFERROR and IFNA are perfectly designed to complement VLOOKUP – in fact you'll almost always want to use VLOOKUP with one of these functions around it!

Advanced Tip

Anytime you use VLOOKUP or HLOOKUP, you'll want to use IFERROR to handle non-match errors.

To fix the #N/A error with our VLOOKUP formula, use IFERROR around the whole thing:

```
=IFERROR(VLOOKUP(A2,$E$2:$F$4,2,FALSE),"other")
```

The first argument in IFERROR is the function we really want. If this function is not an error, we just get the normal lookup result. If it's an error we get the second IFERROR argument, which just outputs "other".

Try this out on a simple example like the one above. VLOOKUP will feel natural with some practice, as will IFERROR.

Note: IFNA would be the better version of the two if your version of Excel supports it – VLOOKUP and MATCH return #N/A errors, and you might want to handle other types of errors differently. IFERROR just treats all errors the same. But be careful: IFNA does not exist in Excel 2010 and earlier. I prefer IFERROR so my spreadsheets are compatible on all computers.

Example: VLOOKUP to Combine Data Sets

A very common use of VLOOKUP is to combine data from two different data sets. For this example, we have a table of accounts with account balances. We want to map the name of each account holder from a separate table. How do you pull in the correct names?

	A	B	C	D	E	F	G	H	I	J
1		Account balances						Names lookup		
2		**Account**	**Balance**	**First Name**	**Last Name**			**Account**	**First Name**	**Last Name**
3		12445	1250					12445	Steve	Smith
4		12447	479					12446	Adam	McDonald
5		12449	5475					12447	April	Alexander
6		12454	7488					12448	Joanna	Bryan
7		12451	1000					12449	Lewis	Clark
8		12457	127					12450	Corey	Smith
9		12456	100					12451	Sam	Rivers
10		12450	5579					12452	Monique	Ortiz

Advanced Tip

When merging two data sets, don't assume they match row for row. Use VLOOKUP to line things up correctly and avoid major errors!

Refer to the screenshot as we solve this simple example. The power of VLOOKUP will be even more apparent with thousand-row data sets!

The first table of values is in columns B:E and the other in columns H:J (it could also be on another worksheet). We're trying to fill out columns D and E with the correct values by matching account numbers.

Notice a few things right away: the two tables don't line up row for

row, and the first table is not even in ascending order. In other words, don't just set D3 = I3 (or copy-paste the tables next to each other) and assume the data sets match up correctly. In practice, even if the rows seem to line up at first glance, it's much safer to assume the elements don't match up line by line. It's easy to make a huge mistake otherwise.

The correct solution involves VLOOKUP. The formula for cell D3, including properly anchored references, is:

```
=VLOOKUP($B3, $H3:$J10, 2, FALSE)
```

This means we look for the account number from B3, inside the lookup table's first column (the match is always in the first column). When we find a match, take the value in column number 2. We want exact matches so the last term is FALSE.

The lookup table might go many more rows than J10. You could reference the entire columns H:J for the lookup, assuming there's nothing else below the lookup table (such as calculations or subtotals, or unrelated data). The formula with full columns ensures the lookup still works, even if you add more rows to the table:

```
=VLOOKUP($B3, $H:$J, 2, FALSE)
```

You can drag this formula down to fill out first names (column D), and drag it right to fill out last names (column E). Except: for last names you want the third column instead of the second column, so just change the 2 to a 3 in the formula.

Watch out for the possibility of multiple matches. If VLOOKUP finds multiple matching rows, it only returns the first one and neglects to tell you there are others. To see if any row has multiple matches, use a COUNTIF formula in new column:

```
=COUNTIF($H$3:$H$10, A3)
```

If this formula returns a number greater than one for any of the rows, you have multiple matches. To solve this you might have to remove duplicates from the lookup table, or find another way to tell Excel the correct matching row.

HLOOKUP

A quick word about VLOOKUP's brother, HLOOKUP. V means vertical and H means horizontal. So an HLOOKUP function looks in the first row of a table, horizontally, until it finds a match. Then it returns the value from the specified row number in that column. HLOOKUP is exactly the same as VLOOKUP, but flipped on its side.

You'll probably end up using it about 1% as often as VLOOKUP, so I'll continue to refer to just VLOOKUP in this book. They are interchangeable.

VLOOKUP #N/A Errors

What if the VLOOKUP/HLOOKUP produces #N/A error values? There are two possible reasons:

- Excel couldn't find an exact match anywhere in the lookup table; or
- the lookup table doesn't have as many columns as the column number you entered (for example, you're asking for the fifth column in a four-column table).

The second problem is easy to fix: if you want the third column from your lookup table, make sure that table is at least three columns wide. Just expand the lookup reference. The first problem is more interesting.

We saw earlier that VLOOKUP returns an #N/A error value when there is no match in the lookup table. Error values are never good to have in your Excel sheets, but sometimes a non-match is a perfectly legitimate

result. It would be better to output something else though – if your sheets have errors people will assume you made a mistake!

We already looked at IFERROR for this task. Use a formula like this to replace any VLOOKUP errors with a blank cell (using two quotes to output blank):

```
=IFERROR(VLOOKUP(A2,Sheet2!A1:B100,2,FALSE), "")
```

Example: VLOOKUP with Error Handling

Sometimes you just want to know whether a data item exists in another set at all. You can use VLOOKUP to make a formula that, for each item in the first column, returns a 1 if there's a match and a 0 if no match in the second column.

You could accomplish the same thing with VLOOKUP and ISERROR:[23]

```
=IF( ISERROR( VLOOKUP(A2,Sheet2!A1:B100,2,FALSE) ), 0, 1)
```

This is a long function but it's not that bad. Let's work from the inside out. Start with the usual VLOOKUP function, which returns either a matching value or an #N/A error. Then we run ISERROR on that, which is a new function. It returns TRUE if the formula inside of it is an error; FALSE otherwise. If there is no match with the VLOOKUP, ISERROR returns TRUE. Now we just need to test this result with the IF function; for TRUE we return 0 (meaning no match), for FALSE we return 1 (meaning match). That's exactly what we wanted!

ISERROR: takes one argument and returns TRUE if the input is an error, FALSE if not an error

Use ISERROR to determine whether there's an error, yes or no. This is a common usage:

```
=IF(ISERROR(<some calculation>), "Error", "OK")
```

What you don't want to do is this:

```
=IF(ISERROR(<some calculation>), "Error", <same calculation again>)
```

The above formula repeats code, which is very inefficient. It's the perfect situation to use IFERROR instead:

```
=IFERROR(<some calculation>, "Error" )
```

VLOOKUP with Range

An alternative use for VLOOKUP is "ranged lookup", which means setting the last function input to TRUE instead of FALSE. To illustrate how cool this feature is (and how it could save you huge amounts of time), we'll start with another example involving nested IF.

We want to return "0-9" if the input number is between 0 and 9, "10-19" if it's greater than 10 and up to 19, "20+" if it's greater than 20. We'll return "negative" if it's less than 0. You could do this with a series of nested IF functions:

```
=IF(A1>=20, "20+", IF(A1>=10, "10-19", IF(A2>=0, "0-9", "negative")))
```

This formula does what you want it to do,[24] but there are problems with it:

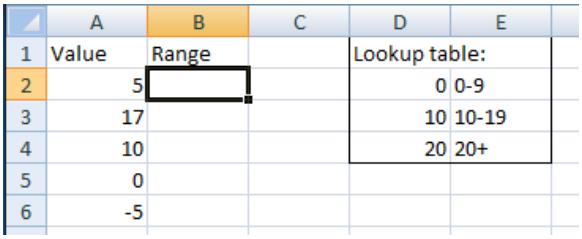
---

[23] Or use a COUNTIF formula: `=MIN(1, COUNTIF(A2, Sheet2!A1:A100))`

[24] Notice that we did not have to specify the IF conditions as between 10 and 20. For example, this function would be overkill for the job: `=IF(AND(A1>=0,A1<10),"0-9", IF(AND(A1>=10,A1<20),"10-19", IF(A1>=20,"20+",`

- There are a lot of steps to type out, and it's easy to make a mistake.
- It's difficult to understand. You have to open up a long formula and really study it.
- If you decide to modify the 10 cutoff to a 12, you need to change the formula in three different places.
- It's not visual. It would be nicer to see a table of the cutoffs and values.

There's an alternative way which fixes all of these issues: a VLOOKUP formula and (as usual) a lookup table. Take a look at this screenshot:



	A	B	C	D	E
1	Value	Range		Lookup table:	
2	5			0	0-9
3	17			10	10-19
4	10			20	20+
5	0				
6	-5				

Note: as you're entering "10-19" Excel will convert it to a date. To make it stop doing that, type an apostrophe (') character first.

This lookup table has just two columns: the input cutoffs, and the output values. There is no need to specify greater than, less than, greater than or equal to, etc. Excel's VLOOKUP always uses greater than or equal to your cutoff. So 0 corresponds to ">=0", 10 corresponds to ">=10", and so on.

Note: with ranged lookup your lookup table cutoffs must be in ascending order. Otherwise, you will get unpredictable and incorrect results.

The formula for B2 is the following, with proper anchoring:

```
=VLOOKUP(A2,$D$2:$E$4,2,TRUE)
```

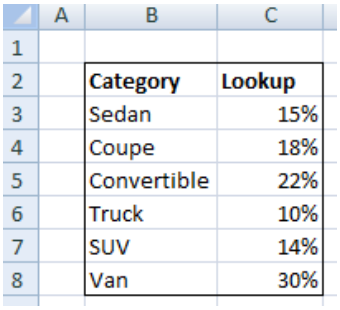
Notice the last "TRUE", which lets Excel know to use ranged VLOOKUP. You can drag this formula down to all the input values, and the results should all be as expected. Except: we get an #N/A error when the input value is negative.

With ranged VLOOKUP, an input value less than your lowest cutoff returns an error. In this case, the lowest cutoff was 0 so any negative value returns #N/A. The solution: use IFERROR to specify what you'll return for negative values. The modified ranged lookup formula looks like this:

```
=IFERROR( VLOOKUP(A2,$D$2:$E$4,2,TRUE), "negative")
```

To recap, VLOOKUP with range set to TRUE allows ranged matches; FALSE requires exact matches. TRUE returns #N/A for inputs less than the first cutoff value; FALSE returns #N/A for inputs that don't match any values. If you mix them up, you'll get the wrong result! For example, here's an exact match lookup table:

`"negative")))` . The formula we used above is easier to type, easier to read, and less likely to have a mistake; but the preferred solution is VLOOKUP.



	A	B	C
1			
2		**Category**	**Lookup**
3		Sedan	15%
4		Coupe	18%
5		Convertible	22%
6		Truck	10%
7		SUV	14%
8		Van	30%

Now let's do a lookup when our input value is a Sedan. The correct formula is this:

```
=VLOOKUP("Sedan", $B$3:$C$8, 2, FALSE)
```

It returns the value of 15%, as it should. But what if we forget the last VLOOKUP input, or set it as TRUE?

```
=VLOOKUP("Sedan", $B$3:$C$8, 2)     or     =VLOOKUP("Sedan", $B$3:$C$8, 2, TRUE)
```

Both of these formulas return 22%, which is obviously wrong! Unfortunately the ranged lookup is not designed for a categorical table like this. To avoid this problem, explicitly use TRUE or FALSE every time with VLOOKUP. And sanity-check what your lookup is doing; if the results don't make sense, then you might have the wrong lookup type.

One final point about VLOOKUP with range: Excel says this returns an "approximate match." This does not mean you can use it for approximate text matching. For instance, it won't help you match "123 Main Street" with "123 N Main St". You do have some options for true approximate matching (also called fuzzy matching) which we will cover in Chapter 9.

INDEX-MATCH

Now that we've covered VLOOKUP, let's look at a pair of functions that accomplishes essentially the same thing: INDEX and MATCH. Here's how you translate a VLOOKUP to an INDEX-MATCH:

```
=VLOOKUP(A2,Sheet2!A1:B100,2,FALSE)
=INDEX(Sheet2!B1:B100, MATCH(A2, Sheet2!A1:A100, 0))
```

Let's break this down from the inside out. The MATCH function finds a match for A2 within a single column, A1:A100 on Sheet2. The function returns the row number of the first matching result. Then the INDEX function looks in B1:B100 for the row number returned by MATCH. If MATCH finds a match in row 5 of column A, INDEX returns the value of row 5 in column B.

MATCH: looks for a match to a lookup value within a row or column array, and returns the corresponding position. It can do exact match, less than, or greater than.

lookup value a single value to find a match in the **lookup array**
lookup array a range that has one column or one row where Excel looks for the **lookup value**
[match type] 0 means exact match; 1 is ascending ranged lookup; -1 is descending ranged lookup

```
=MATCH(A2, Sheet2!A1:A100, 0)
```

MATCH does half of what VLOOKUP does, which is to find the matching row number. Set the match type to 0 if you want an exact match, same as FALSE in the last argument of VLOOKUP. Use match type 1 to replicate a ranged VLOOKUP, with a lookup table of ascending numbers.

Don't use match type -1! Technically it's the reverse of 1: you'll use descending numbers in the lookup table, with less than or equal as the matching rule. But honestly, this feature is more confusing than it is useful. Stick to 0 or 1, which work the same as in VLOOKUP, and build your lookup tables in ascending order.

INDEX: returns the value in a particular position in an array

array	array that the resulting value will be picked from
row num	the row number from the **array** to return; usually the output of a MATCH function
[column num]	the column number to choose within the **array**; skip this if the array is a single column, which is most common

```
=INDEX(Sheet2!B1:B100, 10)
```

INDEX is the second half of the lookup, returning the value from the correct row. The general format for a combined INDEX-MATCH looks like this:

```
=INDEX(<column of source data>, MATCH(<lookup value>,<column of lookup table>, 0))
```

Although you'll usually use INDEX and MATCH together (MATCH inside of INDEX), it might help to see the two separately. Take a look at this screenshot, a repeat of the example from before. This time we have a new column C to store the "match row", which contains the row number where we find a match. The formula in C3 is:

```
=MATCH(A3,H:H,0)
```

	A	B	C	D	E	F	G	H	I	J
1	Account balances							Names lookup		
2	**Account**	**Balance**	**Match Row**	**First Name**	**Last Name**			**Account**	**First Name**	**Last Name**
3	12445	1250	=match(A3,H:H,0)					12445	Steve	Smith
4	12447	479						12446	Adam	McDonald
5	12449	5475						12447	April	Alexander
6	12454	7488						12448	Joanna	Bryan
7	12451	1000						12449	Lewis	Clark
8	12457	127						12450	Corey	Smith

Then the formula for D3 uses INDEX to extract the correct row from Column I, using the MATCH result in C3:

```
=INDEX(I:I,C3)
```

The result is "Steve" which is the correct first name.

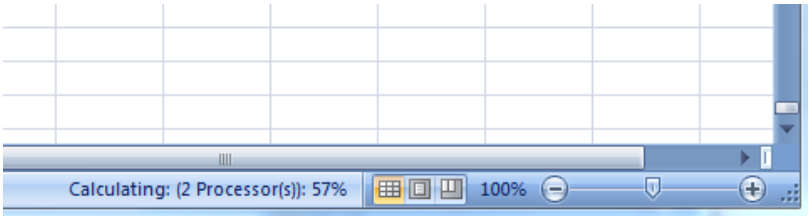
Now here's something pretty cool. You've already got the correct match row in column C. So to get the last name in E3, use another INDEX formula referencing C3:

```
=INDEX(H:H,C3)
```

It becomes very easy to match up multiple columns this way! Just remember to anchor the column $C3 if you drag the formula across.

Why bother learning INDEX-MATCH when VLOOKUP does the same thing? Because **VLOOKUP has some major limitations that INDEX-MATCH fixes**:

- With VLOOKUP you must match the first column of the lookup table. If you want to match based on column C and return column B, you cannot use VLOOKUP. You could perhaps duplicate the data in column B into a new column E, then use VLOOKUP from column C to E (in this case the VLOOKUP uses column 3, to return the third column out of C, D, E). But it's an awkward limitation.
- The VLOOKUP formula requires hard-coding the column number you want, which can lead to errors! Normally Excel's formulas adjust properly if you move columns around or add new columns. The column number of VLOOKUP does not adjust. So you'll need to avoid making too many changes to the lookup table – which can be inconvenient.
- VLOOKUP is inefficient in terms of computing power. When you're dealing with a large amount of data (a few thousand rows or more) with multiple VLOOKUPs from the same table (as we did with the first and last name lookup above), your calculations will really start to slow down. You'll start seeing the "Calculating" message come up more often:



The first and second issues are resolved automatically: The INDEX part needs an array input, and the MATCH part needs another. It doesn't matter what order they are in. It also doesn't matter if you move the columns around: your formulas continue to reference the correct data. It's much easier than changing the VLOOKUP column number (and having to count off columns by hand).

Advanced Tip

VLOOKUP and MATCH are slow. If you are pulling multiple columns from a lookup, use a single MATCH and multiple INDEX formulas to speed things up.

INDEX-MATCH also speeds up your computations. VLOOKUP and MATCH are equally computationally intensive: Excel has to look through all those rows of data one at a time to find a match. But with MATCH you are able to perform a single lookup, and save the row you need. Store the match result in a column called "match row", and reference that value in multiple INDEX functions. The INDEX functions are fast. If you used VLOOKUP you would be using several slow functions, bogging down your entire spreadsheet.

As for non-matching items and the resulting #N/A errors, they work the same way with MATCH as they do with VLOOKUP. Use IFERROR to handle these errors, just like we covered earlier.

INDEX-MATCH also works as an **alternative for ranged lookup**. You'll use 1 as the final argument in the MATCH formula (the match type) instead of 0. Here's how a VLOOKUP compares to INDEX-MATCH if you're using ranged lookup:

```
=VLOOKUP(A2,$D$2:$E$4,2,TRUE)
=INDEX($E$2:$E$4, MATCH(A2, $D$2:$D$4, 1))
```

That last 1 tells Excel to match in the lookup range, with greater-than-or-equal-to logic, exactly like VLOOKUP. The formula works the same way as before.

You can use MATCH to **determine whether a range contains a particular value**. To see if the value of A1 exists within the range B1:B100, use this formula:

```
=IF(ISERROR( MATCH(A1,B1:B100,0)), 0, 1)
```

Reading this from the inside out: the MATCH part returns a row number (1 to 100) if there's a match, and returns an error value if no match. ISERROR returns TRUE for error values and FALSE for everything else. The IF function returns 0 in the error case (no match) and 1 if there's no error (yes match).

OK, that was a long section! Here's a quick summary of VLOOKUP and INDEX-MATCH:

- Use lookups to replace several complicated nested IFs
- Use them for matching up data across multiple tables
- Use them for converting ranges of values to some other output with a lookup table
- Use ISERROR to determine whether data in one table has a matching record in another table
- Use IFERROR for proper error handling any time you have a VLOOKUP or MATCH function
- INDEX-MATCH is more flexible than VLOOKUP

A few final words on troubleshooting your lookup functions (this applies to both VLOOKUP and INDEX-MATCH). Sometimes you may get a bunch of #N/A errors, even though you can "clearly see" there should be a match in the data. Here are the main reasons this could happen:

Check if Excel thinks the values match, using =. Like this:

```
=A2=Sheet2!A7
```

You'll get TRUE if they're actually equal, and in this case VLOOKUP should pick up a match. What if you have seemingly equal values (a 2 and a 2 for instance) that aren't picking up as equal?

- **Numbers stored as text.** If you notice a number being left-aligned within the cell (and you didn't specifically choose left alignment), then Excel thinks your number is text. A number stored as text is not equal to the same number stored as a number. Change it back to a number: sometimes you can open it with [F2] then press [ENTER]; sometimes you need to change the number format from Text to General. And sometimes Excel gives a little overlay to click which converts cells to proper numbers.
- **Extra spaces.** If one of the cells you're trying to match has a trailing space, or two spaces instead of one, their values are not technically equal. Use the TRIM function to get rid of excess spaces, or perhaps SUBSTITUTE. We'll cover these later in this chapter.
- **Anchoring.** Check your formula and make sure the reference to the lookup table is anchored. Otherwise, as you drag down your formula the reference table moves down as well! The reference A2:B5 will move down to become A3:B6, A4:B7, and so on. Anchor it! A good alternative is to reference the entire column, like B:B. Dragging down the formula won't affect a column reference.

Text Functions

We'll look at functions for cleaning up data and extracting values from text. These are best illustrated with a series of examples.

Example 1: Clean up excess spaces from text

Get rid of the extra spaces in this text: " Here's some text with too many spaces. "

Solution: Use the TRIM function to get rid of leading and trailing spaces, as well as any superfluous spaces between words. If your original text is in A1, you want `=TRIM(A1)`. The result is: “Here’s some text with too many spaces.”

The SUBSTITUTE function is also good for modifying text. You might use `=SUBSTITUTE(A1," ","")` to substitute all spaces (“ “) with nothing (““); in other words, to remove all spaces. In this example it would return: “Here’ssometextwithtoomanyspaces.” You could also use SUBSTITUTE to replace any specific text with some other text; for example, use `=SUBSTITUTE(A1," ","-")` to replace all spaces with hyphens. This would return: “Here’s-some-text-with-too-many-spaces.” You could use this to format phone numbers, for example.

> **TRIM**: removes all excess spaces from text, including double spaces between words.
>
> **SUBSTITUTE:** takes three arguments: text, old text, new text. The function starts with the **input text** and replaces any instances of **old text** with **new text**.

Example 2: Extract state and zip code from an address

The key with this exercise is to make sure the formula works on all inputs, as long as those addresses are in a specific format: street address, city, state zip. Comma placement matters. The inputs are standard U.S. formatted address such as “123 Main St, Los Angeles, CA 90001”.

This screenshot contains the examples. It also shows additional columns I’ve set up for interim results:

	A	B	C	D	E	F	G
1	**Full Address**	**Zip**	**State 2-Char**	**Full excl. Zip**	**1st comma**	**2nd comma**	**State**
2	123 Main St, Los Angeles, CA 90001						
3	1600 Amphitheatre Parkway, Mountain View, CA 94043						
4	1 American Rd, Dearborn, MI 48126						
5	100 North Tryon Street, Charlotte, North Carolina 28255						

Solution: We’ll use multiple functions and multiple interim calculations. The key is to stay organized and go step by step.

The zip code is easy, so we’ll start there. Use the RIGHT function to get just the rightmost 5 characters. Put this in B2 where we’ll save the zip codes:

```
=RIGHT(A2, 5)
```

Drag down this formula for each row. You’ll see it works for all rows!

Let’s get the state next. It’s easy if your states are always two characters long. This formula goes in C2:

```
=MID(A2, LEN(A2)-7, 2)
```

The MID function grabs a certain number of characters from the middle of the text. The first argument is the text (the full address). The second argument is the starting position. How do we know which one that is? We need whichever one is 8 characters from the end of the address (because all of them end with 5 characters for the zip, 1 for a space, and 2 for the state). So use `LEN(A2)` to get the total length of the address. Back up 7 characters from there (one less than the 8). The last argument says we need 2 characters in the output.

Figuring out the correct inputs to MID can be quite tricky, especially getting the starting position correct. When working back from the end, you’ll always use LEN and subtract some number. I recommend using trial and error until you get the right result. If it works once it works every time, as long as the inputs are formatted consistently.

Are we done? Yes, if the state is always two characters!

The above solution breaks down if the state name is spelled out, like “North Carolina”. We’ll need some other way to identify when the state starts. Let’s do this: look for the second comma and pick up everything else to the right of it (besides the zip code). This will take a few steps, but it’s doable – you’ll like the trick you’re about to learn.

We already know where the state name ends: just before the zip code. So start by extracting everything before the zip, and store this in column D. The formula to cut off the last 6 characters (the zip code and the preceding space) is LEFT:

```
=LEFT(A2, LEN(A2)-6)
```

And the result looks something like: “100 North Tryon Street, Charlotte, North Carolina”

Now we need to know where “North Carolina” begins. It’s always after the second comma; use the SEARCH function to find it. First let’s get the character number of the first comma, using the SEARCH formula in E2:

```
=SEARCH(",", D2 )
```

It means we are looking for a comma inside text D2, searching from left to right. The result is a number which represents the position of the first comma.

There’s a third argument for SEARCH which is optional: the starting character. You can tell SEARCH to skip ahead before looking for the comma. To find the second comma, we need to SEARCH again for a comma in D2; this time starting right after the first comma. Use this formula in F2:

```
=SEARCH(",", D2, E2+1)
```

With the North Carolina example, you should have 23 in E5 (the first comma is the 23rd character) and 34 in F5 (the second comma is the 34th character). Finally we’ll grab everything from the right, except these first X characters up through the second comma. The final formula to get the state in column G is:

```
=RIGHT(D2,LEN(D2)-F2-1)
```

This will be familiar now: go from the right and grab all characters that are the length of D2, except the first F2 amount of characters, minus 1. That last minus 1 gets rid of the space after the comma. Here’s how the final result looks:

	A	B	C	D	E	F	G
1	**Full Address**	**Zip**	**State 2-Char**	**Full excl. Zip**	**1st comma**	**2nd comma**	**State**
2	123 Main St, Los Angeles, CA 90001	90001	CA	123 Main St, Los A	12	25	CA
3	1600 Amphitheatre Parkway, Mountain View, CA 94043	94043	CA	1600 Amphitheat	26	41	CA
4	1 American Rd, Dearborn, MI 48126	48126	MI	1 American Rd, D	14	24	MI
5	100 North Tryon Street, Charlotte, North Carolina 28255	28255	na	100 North Tryon S	23	34	North Carolina

LEFT: takes a certain number of characters from the beginning (left) of a text

RIGHT: takes a certain number of characters from the end (right) of a text

MID: takes characters from the middle of a text, starting at a specified character number

LEN: gets the length (number of characters, including spaces) of a text

SEARCH: looks for specific text inside another text; returns error if not found. Not case sensitive

FIND: same as SEARCH, but case sensitive

A different use of SEARCH or FIND is to see if any cell in a range contains another cell's value (not equal, just containing). For example, is the text in A1 anywhere inside the cells within B1:B10? Use this formula in C1:

```
=IF(ISERROR(SEARCH($A$1, B1)), 0, 1)
```

C1 has the value 1 if A1 is contained within B1. Drag this formula down through C2:C10 to check for A1's value inside B2:B10. Then, if any cells in Column C are a 1, you have a match in the corresponding row of Column B.

Example 3: Joining strings together

There are several ways to join strings of text together. One way is the CONCATENATE function, which joins together the value of all of its arguments.

CONCATENATE: joins together multiple strings into one string

If we have cells A1, A2, A3 with values "red", "white", "blue", then `=CONCATENATE(A1, A2, A3)` outputs "redwhiteblue" without any spaces. You can add the spaces as well:

```
=CONCATENATE(A1, " ", A2, " ", A3)
```

There's a simple alternative to CONCATENATE: use the "&" character between strings. The previous function could be replaced with:

```
=A1&" "&A2&" "&A3
```

Unfortunately the following does not work: `=CONCATENATE(A1:A10)`. You can't concatenate ranges with Excel functions, but you will learn in Chapter 8 to build a custom VBA function that does concatenate a range.

CONCATENATE and VLOOKUP

A great practical use of concatenation involves VLOOKUP for matching two data sets. The earlier examples all involved matching based on a single column's values. What if multiple items need to match? For example, take some data that includes first names, last names, and states. We want all three fields to be equal across the two datasets to consider the row a match. VLOOKUP and MATCH can only use one field, so how do we match on three?

The solution: define a new column that is the combination of first name, last name, and state. Create this new column in both your tables (the target table and the lookup table). Ideally you should add in separating characters such as an underscore character (_) to make it easier to read. Your new column in each table should have formulas like:

```
=A2&"_"&B2&"_"&C2
```

These columns then have values like: JOHN_SMITH_CA. Now you can use VLOOKUP or INDEX-MATCH on the new columns, just like before. Any match you do find will have first name, last name, and state all in common.

Example 4: Clean up text to make a valid file name

In cell A1 we have some text that contains special characters such as period, question mark, quotation, number sign, etc. Like this: Joe's "#1" Motors!![25]

[25] It's the name of a fictional car dealership. They're terrific.

I want to save an Excel file with this text as the filename, but that's impossible! Windows does not allow number signs and exclamations in file names. We need to clean this up a bit first. Then we could create a macro that saves the current file, named as the contents of this cell.

The solution: use the SUBSTITUTE function multiple times to replace text with different text (in this case, to replace certain prohibited characters with nothing). First let's get rid of the exclamation (!) characters:

```
=SUBSTITUTE(A1, "!", "")
```

We looked at the SUBSTITUTE function earlier; it first takes an input text, then what text to replace, and finally what text to put as replacement. The formula above means we take cell A1 and replace all exclamations with blank (double quotes).

Use SUBSTITUTE again to get rid of the quotation marks:

```
=SUBSTITUTE( SUBSTITUTE(A1, "!", ""), """", "")
```

You might be wondering about the four quotation marks all in sequence. It's the correct way to refer to the quotation mark character, which we have to put inside two other quotes. Excel complains if you only put three quotes, so type four of them and you're all set.

The above nested SUBSTITUTE gets rid of question marks and quotes. If you want, you could keep going with more layers of SUBSTITUTE, removing characters such as #.

Example 4: Converting to upper, lower, and title case

Sometimes you want to convert text to all uppercase (capital letters), all lowercase, or title case (proper case, capitalizes first letter of each word).

The functions are: UPPER, LOWER, PROPER.

UPPER: converts text to all uppercase

LOWER: converts text to all lowercase

PROPER: converts all text to proper case, meaning it capitalizes the first letter of all words

For an example, let's have cell A1 containing the text "This is a POORly formatted SENTence."

```
=LOWER(A1) would return: "this is a poorly formatted sentence."
=PROPER(A1) would return: "This Is A Poorly Formatted Sentence."
```

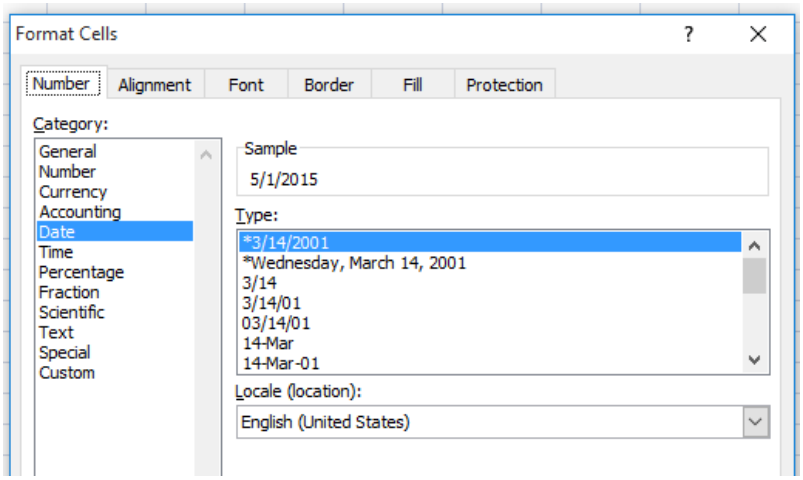
Now, if you know something about proper English grammar, you'll recognize that the above "proper case" is not really correct "title case". You don't want to capitalize every single word; "A" and "Is" should be lower case. Excel doesn't have a built-in function to handle these exceptions.[26]

Date Functions

When you work with date values in Excel, you might notice some strange behavior. You've probably seen a bunch of 1/0/1900 dates on your sheet – that's not even a real date! Or you might see a bunch of values like 41905 where dates should be. The reason: Excel stores dates as numbers. The number "0" corresponds to the date just before the first day of 1900 (really December 31, 1899, but displayed as January 0, 1900). Then you add 1 to move forward one day, so 1/1/1900 has the value 1. You're reading this book over 42,500 days from the year 1900 (42,500 corresponds to 5/10/2016).

[26] And building a custom function for it is actually quite difficult. I think we're stuck without a good solution to output "title case", unless you want to Google for some pre-made code.

Whether these values are displayed as dates (4/9/1900) or numbers (100) depends on the data type for that cell. For working with dates, you'll need to switch between General and Date format. In Excel 2007 and newer, use the dropdown under the Home ribbon to pick Short Date. In all Excel versions you can bring up format cells ([CTRL] + [1]) to get a lot more date formatting choices.



Notice that changing the formatting of the cell from numbers to dates doesn't change the value of the cell. The value is still a number, which happens to be expressed as a date. If the value of A3 is 100, and B3 is `=A3*2`, then B3 has value 200. If you convert A3 to a date, B3 will still have value 200.

You can do simple calculations on dates; for example, counting the number of days until a certain date. Since every date is a number, subtracting one date from another date tells you the number of days between them. If your column A contains a bunch of dates, here's how many days from today those dates are:

```
=A2-TODAY()
```

You can drag down this formula to get date differences as of today. You definitely don't want to do this date calculation by hand!

These are some easy date functions:

TODAY: outputs today's date; takes no inputs but requires "()" at the end

NOW: outputs current date and time; takes no inputs but requires "()" at the end

The following functions let you extract the year, month, or day value from a date.

YEAR: returns the year (as a number) from an input date

MONTH: returns the month (as a number 1-12) from an input date

DAY: returns the day (as a number 1-31) from an input date

For example: let's say A1 shows the date 9/30/2014. The formula `=MONTH(A1)` returns the value 9. And by the way, `=YEAR(42000)` returns 2014, because the number 42000 corresponds to the date 12/27/2014 (remember that every date has a corresponding number value).

Be careful when working with dates inside functions. For example:

```
=MONTH(9/30/2014)
```

This returns the value 1. Why? Excel evaluates the slashes as division: first take 9, divide by 30, then divide by 2014. That's a value between 0 and 1, which in date form would be on 1/0/1900. The month portion of this date is 1, so that's the result. Instead, use quotes:

```
=MONTH("9/30/2014")
```

This correctly returns 9. Because of the quotes, Excel reads the date as text rather than a number, and evaluates it to be a date in September. The result is 9. Here's another way:

```
=MONTH(41912)
```

This also returns 9, because 41912 is the Excel numeric value for the date 9/30/2014. The MONTH function takes the month part of this date, which is 9. In real life, you won't know these numeric values and you would never type out `MONTH(41912)`. But it could be the value of another cell, which you pass to the MONTH function. Finally, another option:

```
=MONTH(DATE(2014, 9, 30))
```

This also returns 9. The DATE function is used to build a date from individual inputs for year, month, and date. Then the MONTH function just takes out the 9.

DATE: builds a date output from separate year, month, day inputs

year	input year as a number (2016)
month	input month as a number (5)
day	input day as a number (12)

```
=DATE(2015, A1, B1)
```

The DATE function lets you manipulate dates; for instance, you can move backward or forward a certain number of months. Below are a few examples. In each case we reference cell D1, which contains the date 10/15/2014.

Find the first day of the current month:

- `=DATE( YEAR(D1), MONTH(D1), 1)`
- This works by building a date with the DATE function, from the year portion of D1 (2014), the month portion of D1 (10), and forcing 1 as the day.

Find the last day of the previous month:

- `=DATE( YEAR(D1), MONTH(D1), 0)`
- Exactly the same as before, but using 0 as the day value forces Excel to wrap around to the last day of the previous month.
- Keep in mind that since months can end on day number 30, 31, 28, or even 29, we can't do this: `=DATE( YEAR(D1), MONTH(D1) -1, 30)`. Yes, this function works when D1 is October, but we need our functions to work for all inputs. If the input is in November, we would get October 30 as an output, which is not the last of the month. If the input is in March, we would actually get March 2 as an output (because there is no 30th of February, the date wraps around into March).

Find the last day of the current month:

- `=DATE( YEAR(D1), MONTH(D1)+1 , 0)`
- Go forward to the next month, and then back to "day 0", which is the last day of the current month.

- If you want to type less and remember one more function, use EOMONTH. Like this: `=EOMONTH(D1)`. This function returns the last day of the current month.

Find the same day from one year ago:

- `=DATE( YEAR(D1), MONTH(D1)-12 , DAY(D1))`
- Or: `=DATE( YEAR(D1)-1, MONTH(D1), DAY(D1))`
- In either case we are backing up a full year from the current date.
- Don't use this: `=DATE( YEAR(D1), MONTH(D1), DAY(D1)-365)` because in a leap year you get the wrong result.
- Another function to accomplish the same thing: EDATE, which lets you move forward or backward a specific number of months. In this case `=EDATE(D1, -12)` goes back exactly 12 months.
- EDATE is smart enough to know that if the input is the last of the month, the result is also the last day of the month – even if those months have a different number of days! For instance, `EDATE("12/31/2015", -1)` and `EDATE("12/30/2015", -1)` both return 11/30/2015. That's some smart Excel!

EOMONTH: returns the last day of the month for the input date

EDATE: moves a specified number of months forward or backward from the input date

NETWORKDAYS: calculates number of workdays between two input dates

NETWORKDAYS counts the number of workdays between dates, as opposed to total days. This function can be useful for calculating business days in a month. For example `=NETWORKDAYS("2016-01-01","2016-01-31")` returns 21, meaning 21 weekdays between these dates. But it does not account for holidays such as January 1! To specifically exclude holidays from this function, you'll need to list them as the third function argument. To best way to do this is to set a range of cells where you list holidays; for example 1/1/2016, 1/18/2016, 2/15/2016 in range A1:A3. Then your function is:

```
=NETWORKDAYS("2016-01-01","2016-01-31", A1:A3)
```

The above function outputs 19, which is the original 21 days minus the two holidays in January.

Data Types

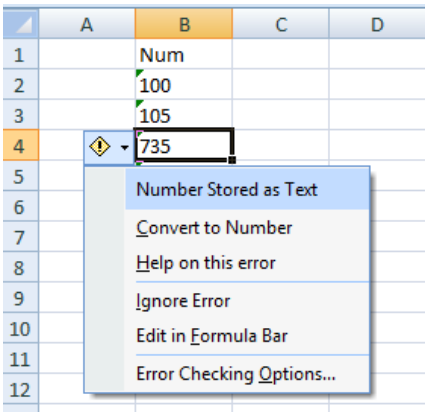
Excel data types matter. A cell's given value can be displayed as a number, a date, or text. We saw that VLOOKUP will not find a match between two numbers if one is formatted as a number and the other formatted as text. And we saw in the previous section that dates are really just numbers formatted in a particular way.

This section reviews two functions designed to navigate between number and text formatting: VALUE and TEXT.

VALUE: converts text formatted as number to a number; returns error if input is not a number

TEXT: converts an input value into a custom number format

The easiest way to identify a number formatted as text is by seeing it left-aligned in the cell. By default Excel left-aligns text and right-aligns numbers (you can override alignment manually of course, but left-aligned numbers should be very rare). You'll sometimes also get a green triangle in the cell's corner. You can click the yellow exclamation box to see some options in the hover overlay. And yes, you can choose to Convert to Number, but only one cell at a time!



If you have a lot of numbers stored as text, you'll want an easy way to turn them all back into numbers. The trick is to use **Text to Columns** under the Data ribbon. This command is designed to convert a long string of text into multiple columns, by looking for separators such as tabs or commas. We can also use it to convert a single column of numbers stored as text into proper numbers:

1. Select the entire column of numbers stored as text
2. Click Text to Columns which brings up a wizard popup
3. Keep the default choice of Delimited
4. You have a choice of delimiters, you want Tab
5. The last step asks for the column data format; General is perfect

That's it! As long as your original cells didn't have any Tabs in them (they shouldn't, since these were all numbers), they are now converted to numeric values.

As an alternative, use the VALUE function in a new column (assuming your data is in columns) to replicate the values as numbers. Here are the steps:

1. In a new column, put `=IFERROR(VALUE(B2), B2)`, where Column B has your numbers stored as text. Drag this formula to fill as many rows as you need. The new column now has the same values as the original Column B, formatted as numbers.
2. Convert the original data in column B to General format (anything but text).
3. Copy and Paste-Special Values from the new column into the original column B.
4. Delete the formulas from the new column.

Notice that I combined IFERROR and VALUE. Remember this little piece of code:

```
=IFERROR(VALUE(A1), A1)
```

This converts numbers stored as text into proper numbers, but leaves other text alone. Without the IFERROR, any non-numeric text would return error values.

To do the opposite, you can use the **TEXT function to convert a number into text**. More importantly, you can use this function to specify how you want your number formatted (using decimals, commas, dollar signs, etc.).

Use this formula to output "50% complete" in cell, referencing cell F5:

```
=TEXT(F5, "#%") & " complete"
```

The TEXT function takes the value of F5 (which is 0.5 or 50%, depending on the cell formatting) and outputs it specifically in a percentage format. The second argument "#%" specifies that we want a number (#)

followed by the percentage sign. The last part of the formula just adds on the word “complete” with a space in front.

How do you know about “#%” format inside the TEXT function? These custom TEXT formats are also used in cell formatting; open up Format Cells ([CTRL] + [1]), the Number tab, and the Custom category. You’ll see several pre-defined formats, and you can also make your own. We’ll cover what the various symbols mean in Chapter 9. For now I’ll show you five common number formats. In each case below, the input value is 42345.678, the middle column is the number formatting string, and the last column is the output display.

Round to whole number	#	42346
Round to whole number with thousands commas	#,###	42,346
Display exactly 2 decimals	#.00	42345.68
As date: 01/10/2015	MM/DD/YYYY	12/07/2015
As date: year, month, day	MMM D, YYYY	December 7, 2015

One last point about data types: Excel likes to convert things that kind of look like dates into dates. For example if you enter any of the following values into cells, they all convert to October 15 (displayed as “15-Oct” which is a pretty unusual date format):

- 10/15
- 10 / 15
- 10-15
- 10 - 15

So how do you display “10-15” (ten to fifteen)? You have two options:

- Type an apostrophe before the text, which forces it to display exactly what you entered (‘10-15) without showing the apostrophe itself.
- Set the cell’s data type to Text (on the Home ribbon, instead of General) prior to entering anything. Then enter 10-15 and it will stay that way.

Financial Functions

Excel is a great tool for financial calculations. If you’re familiar with financial concepts such as amortizing loan calculations, NPV and IRR, and CAGR, you’ll definitely want to know how they work in Excel. Everyone should have some practice with the PMT function – it lets you calculate a monthly mortgage or car loan payment based on the loan amount and the interest rate!

For amortizing loan calculations (such as a mortgage) the relevant variables are rate, payment amount, number of payments, present value, and sometimes future value (the ending balance, which is $0 in most cases).

These five functions are all inter-related; if you know four of them you can calculate the missing item:

- PMT
- NPER
- RATE
- PV
- FV

FV is an optional argument; if you skip it, Excel assumes 0.

Below is the full function for PMT. The others look very similar:

PMT: calculates the periodic payment amount for an amortizing loan or investment

rate	periodic interest rate (not necessarily annual)
nper	number of periodic payments
pv	present value
[fv]	future value; defaults to 0 if omitted
[type]	0 to pay at end of period (default); 1 to pay at beginning of period

```
=PMT( 6%/12, 30*12, -300000)
```

The example in the box above would calculate the monthly payment for a mortgage of $300,000, assuming a 6% annual rate (divide 6% by 12 to get the monthly rate), over 30 years (multiply 30 by 12 to get the number of months).

The Type argument lets you choose beginning of period or end of period payments. If you're not sure what to pick, use the default end of period payments. That is most common in real life.

You can calculate number of periods with NPER:

```
=NPER( 6%/12, 1799, -300000)
```

This formula gives us number of months, where the annual interest rate is divided by 12 to make it a periodic (monthly) rate, and the payment amount $1799 is a monthly payment. Divide this result by 12 to get number of years instead of number of months.

To get the annual rate, use RATE:

```
=RATE( 30*12, 1799, -300000)*12
```

This formula calculates the periodic (still monthly) interest rate given $1799 payment and 360 periods (30 years). Don't forget to multiply by 12 at the end to get the annual interest rate of 6%.

To get the present value, use the PV function:

```
=PV( 6%/12, 30*12, 1799)
```

This formula answers the question: how large a loan can you afford by paying 6% interest per year, for 360 monthly payments of $1799? The answer is $300,000.

Finally, you could calculate future value of an investment with FV:

```
=FV( 6%/12, 30*12, 1798.65, -300000)
```

This particular formula is close to a $0 result, because we already calculated the inputs such that the loan pays down to zero after 360 months. If you used a smaller payment or a larger interest rate, there would be a positive future value at the end of the term.

Remember that these functions use **directional cash flows**. This means that the positive or negative sign of the PMT, PV, and FV values matters. From the perspective of a lender, PMT is a positive cash flow (money in) but PV is negative (money out). Use opposite signs for PMT and PV or you'll get the wrong result.

In the next section you'll practice the PMT function using the example of a car loan. You'll also calculate how much money the bank makes on that loan.

Calculating NPV or IRR of a series of cash flows involves two variations of the IRR and NPV functions. With annual cash flows, use these functions:

NPV: calculates NPV of a stream of *annual* cash flows at a specified annual discount rate

IRR: calculates internal rate of return of stream of *annual* cash flows

Each of these takes a range of cash flows; NPV also requires a discount rate input. IRR has an optional Guess argument, which is helpful in case you get a result that doesn't make sense (such as a negative value or a really large one). For some cash flows, mathematically the IRR function can have multiple solutions. Put in a reasonable value such as 12% for the guess, and Excel will do a better job returning the result that makes sense.

IRR sometimes returns a #NUM! error, meaning there's no solution (it's not possible to solve for a rate of return that yields zero NPV). If you don't think that's right, common things to check are: your first cash flow should (usually) be a negative value, and the sum of all cash flows should (usually) be greater than zero.

Advanced Tip

Use XIRR and XNPV instead of IRR and

I recommend the XIRR and XNPV functions instead of IRR and NPV. These take any periodic cash flows and let you provide your own dates. If you have monthly cash flows in a column, create another column with dates one month apart. Regular IRR and NPV can only use annual cash flows, but very few real-life financial products or projects have cash flows just once a year. (What if you don't care about the actual dates? Then just set the first cash flow at an arbitrary date like 1/1/2015. Use the EDATE function to advance by one month in the rest of the column.)

Once you have a series of cash flows and a matching series of dates, you can use XNPV and XIRR:

XNPV: calculates NPV of cash flows on specified dates, with a chosen *annual* discount rate

rate	annual discount rate
values	range containing a series of cash flows
dates	range containing a series of dates (corresponding to the cash flows)

```
=XNPV(10%, A2:A10, B2:B10)
```

Note: For XNPV the discount rate is still an annual rate, even if your cash flows are not annual. This is just the convention in finance.

XIRR: calculates IRR of cash flows occurring on specified dates

values	range containing a series of cash flows
dates	range containing a series of dates (corresponding to the cash flows)
[guess]	a reasonable guess value to help estimate the IRR result

```
=XIRR(A2:A10, B2:B10, 15%)
```

XIRR also has a guess argument; feel free to put a reasonable number such as 15%. Note that the result of the IRR formula is an annual rate. And remember that XIRR requires that the first cash flow is negative (but XNPV also works with positive first period cash flows).

One last calculation: compound annual growth rate (CAGR). Unfortunately if you Google the term "Excel CAGR" you are likely to find an article that tells you to use the XIRR function.[27] This article is incorrect! XIRR would get you the wrong result: it calculates an annual rate of return on an investment with a negative first period cash flow. CAGR, by contrast, calculates the annual growth rate of a quantity such as the annual sales for a company, and it always starts with a positive value.

There is no dedicated function for CAGR, but you can use the following formula to calculate it:

```
=(ending/beginning)^(1/periods)
```

The variables "ending", "beginning", and "periods" could be cell references or hard-coded values as usual. In Chapter 8 you will learn how to build a custom CAGR function.

Practice: Amortizing Cash Flows

We'll take a short detour to test out your use of advanced functions, anchored references, and of course keyboard shortcuts. Think of it as a summary of Chapters 2-4 so far.

Goal: build an amortizing cash flow table showing interest and principal payments on a car loan with fixed monthly payments. Calculate NPV and IRR of the cash flows as outputs. This task involves:

- Basic calculations
- Financial functions
- Anchoring cells
- Keyboard shortcuts to copy formulas
- Keyboard shortcuts for formatting
- Using EDATE to work with dates
- XIRR and XNPV

Go to www.AdvancedExcelBook.com for a template (and the solution) for this exercise. You'll need to fill out all the cells highlighted in gray.

	A	B	C	D	E	F	G	H	I
1		**Inputs**				**Outputs**			
2		Loan Amount	$25,000.00			*Returns for Lender:*			
3		Loan Term	60	*months*		NPV		3.0%	*discount rate*
4		Annual Rate	4.90%			IRR			
5		Monthly Payment							
6									
7		**Month**		**Begin**	**Total**	**Interest**	**Principal**	**Ending**	**Lender**
8		**Number**	**Date**	**Balance**	**Payment**	**Paid**	**Paid**	**Balance**	**Cash Flow**
9		0	1/1/2016		$0.00			$25,000.00	
10		1							
11		2							
12		3							

[27] "Calculate a compound annual growth rate (CAGR)", http://office.microsoft.com/en-us/excel-help/calculate-a-compound-annual-growth-rate-cagr-HP001122506.aspx

To set up the exercise without downloading the template, use this screenshot. All the numbers are hard-coded values (constants), with the exception that the Ending Balance for Month 0 links to the original loan amount. Fill out the rest of the model with formulas. The values for Month 0 are already set up in the screenshot. Here are some hints for how to calculate Month 1:

- Date = 1 month after Month 0's date
- Beginning balance = the ending balance of Month 0
- Total payment = link to the payment amount (calculated)
- Interest paid = beginning balance x monthly interest rate (link to the annual interest, divide by 12)
- Principal paid = total payment minus interest paid
- Ending balance = beginning balance minus principal payment

You'll know you did the amortization correctly if the ending balance after the last period is approximately $0.00. It may be off by a fraction of a cent due to rounding. To get NPV and IRR values, you'll need the lender's cash flows (both positive and negative cash flows) all in one column; you'll also need the dates.

Try this exercise using only keyboard shortcuts (no mouse). It can all be done with the core keyboard shortcuts. You'll also want to format your cells with proper number formatting, using two decimal places and dollar signs.

When you're done, let's move on to a few more functions.

ISBLANK, ISNUMBER, IS-Whatever

These functions are best used with IF functions. They all take a single input, evaluate that input, and return either TRUE or FALSE. Usually you'll put cell references within these functions, but you could put more complex expressions. Here's a simple example:

```
=IF(ISNUMBER(B1), "B1 is a number", "B1 is not a number")
```

The following are the common IS-Whatever type functions you might find useful.

ISNA: returns TRUE if the input to the function is an #N/A error; returns FALSE otherwise

ISERROR: returns TRUE if the input is any kind of error

ISNUMBER: returns TRUE if the input is a number; blank cells will return FALSE

ISBLANK: returns TRUE if the input is a reference to a blank cell

ISEVEN / ISODD: returns TRUE if the input is an even number / odd number

OFFSET and INDIRECT

OFFSET and INDIRECT are both used to dynamically reference cells and ranges. OFFSET in particular is one of my favorite functions; it adds great flexibility to your work!

OFFSET: changes the reference range; moves it up/down, left/right, and resizes it

reference	start at this reference range (one or more cells)
rows	shift this number of rows down (negative value to shift up)
cols	shift this number of columns right (negative to shift left)
[height]	change height to number of rows (minimum 1); optional
[width]	change width to number of columns (minimum 1); optional

```
=OFFSET(A1, 0, 1, 10, 1)
```

OFFSET may seem a little weird. A few examples should help; first without altering height or width:

- `=OFFSET(A1, 0, 1)` is a reference to cell B1 (move 0 rows down, 1 column to the right)
- `=OFFSET(B2, -1, -1)` is a reference to cell A1 (more 1 row up and 1 row left)
- `=SUM(OFFSET(A1:A3, 0, 1)` sums up the values in B1:B3

And if you use the last two parameters, you can resize references:

- `=OFFSET(A1, 0, 0, 10, 1)` is a reference to A1:A10
- `=OFFSET(A1, 1, 0, 10, 2)` is a reference to A2:B11
- `=OFFSET(A1, 0, 0, 10, 0)` returns a #REF! error because you can't have 0 width

The best use of OFFSET is for scenario analysis in modeling: you'll be able to easily plug different sets of assumptions into your models. Here's what I mean: let's say your Excel model takes five key assumptions. Line these up in a column and label them. Column B will have the values that actually drive the model (see screenshot below).

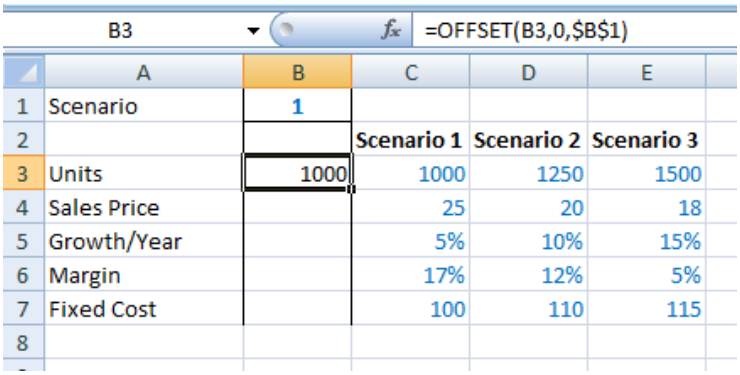
Advanced Tip

Use OFFSET to easily switch between different sets of inputs (assumptions) into your models. Good Excel models have various scenarios, such as a conservative and an optimistic one.

Next, start creating scenarios. Scenario 1's assumptions go in column C; Scenario 2 goes next to it in column D; and Scenario 3 goes in column E (see screenshot on next page, columns C, D, E). You now have three columns with hard-coded inputs that serve as your three sets of assumptions. How do you easily switch the model between these three scenarios? Use an OFFSET function for each input in column B. So B3 would have the formula:

```
=OFFSET(B3, 0, $B$1)
```

This gives you a circular reference warning – don't worry about it yet! We haven't set any value in B1, so by default B3 is now referencing itself (offset by 0 rows and 0 columns). Let's fix that now: type 1 in cell B1, and let's also label it in A1 as "Scenario". Here's what this looks like when everything is set up so far, with a bit of formatting (blue cells are hard-coded constants; black cells are formulas or labels):



B3 =OFFSET(B3,0,$B$1)

	A	B	C	D	E
1	Scenario	1			
2			Scenario 1	Scenario 2	Scenario 3
3	Units	1000	1000	1250	1500
4	Sales Price		25	20	18
5	Growth/Year		5%	10%	15%
6	Margin		17%	12%	5%
7	Fixed Cost		100	110	115
8					

Don't forget to anchor the B1 reference in cell B3. Drag down this formula through B3:B7. Now comes the fun part: switch B1's value to 2 or 3, and all the values in B3:B7 change! If these values are driving the rest of your spreadsheet (don't link anything to C3:E7 directly), you'll quickly see how a different set of assumptions impacts your calculations. **This is scenario analysis; it's incredibly useful in Excel modeling**.

The OFFSET function's last two arguments (height and width) work well with SUM or AVERAGE functions that take ranges as inputs. Your offset can turn a cell reference such as A1 into a range reference such as A1:A10. You can even make the size of this range dynamic based on cell inputs:

```
=SUM(OFFSET(A1, 0, 0, B1, 1))
```

This function takes the sum of cells A1 down through B1 number of rows in height. Just make sure B1 is an integer greater than 0.

The INDIRECT function takes a single input, a "reference text string". Here's what that means: if cell A1 has the value 1, then `=INDIRECT("A1")` is the same as writing `=A1`, and returns the value 1.

INDIRECT: references a cell based on input text (e.g. "Sheet1!A1")

INDIRECT becomes useful when you pass cell references to it, instead of hard-coding text such as "A1". For example, you might have cell C1 with the value "Sheet2", C2 with the value "B", C3 with the value "2". What does this function reference?

```
=INDIRECT(C1&"!"&C2&C3)
```

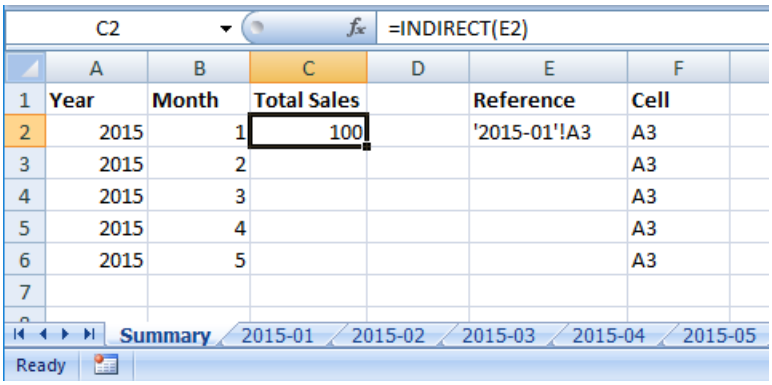
The text inside INDIRECT is "Sheet2!B2", meaning the value of the function is whatever is in B2 on Sheet2 (don't forget the exclamation after the sheet name). The benefit of this function should be obvious: changing the value of C1, C2, C3 lets you reference a different cell with the INDIRECT function.

Sometimes your sheet names have spaces or dashes in them; these characters confuse the INDIRECT function. For instance, this doesn't work: `=INDIRECT("My Sheet!A1")`. You need to enclose the sheet names with single quotes ('), like this: `=INDIRECT("'My Sheet'!A1")`. The formula inside the parentheses starts with a double quote and then a single quote. It's tricky but it works!

Here's a practical example of INDIRECT: let's say your company maintains a big file with tabs representing monthly sales numbers. Your tabs are labeled "2015-01", "2015-02", "2015-03" and so on; and from each tab you just want the value from cell A3 (you'll want to put the same outputs of each tab in the same cell!). To reference January 2015's data, pass the following text into the indirect function: `'2015-01'!A3`

Remember to include the apostrophe before and after the sheet name if it has spaces or dashes. The exclamation separates the sheet name from the cell reference. A3 is the cell we want.

Once you have the actual reference formula, use it in a simple INDIRECT function to get the result. Refer to the screenshot below:



The tricky thing is not Column C (that's just simple INDIRECT), but rather how to make Column E dynamic based on columns A, B, F. The formula for E2 is:

```
="'" & A2 & "-" & IF(B2<10,"0","") & B2 & "'!" & F2
```

This looks like a lot, but don't just skim over it! Let's take it step by step. There are various components here, concatenated by "&" signs:

- Start with single apostrophe inside double quotes (')
- Reference A2 for the year ('2015)
- Add a dash inside double quotes ('2015-)
- If the month is single-digit, add a leading zero ('2015-0)
- Reference B2 for the month number ('2015-01)
- Another apostrophe to end the sheet name, and an exclamation ('2015-01'!)
- Reference F2 for the cell address ('2015-01'!A3)

With E2 correctly set up, you can drag down E and C and get the results you want from each sheet. For this kind of problem, you definitely don't want to manually reference cells on 10+ sheets! Use INDIRECT – it should only take a minute or two.

A final note about OFFSET and INDIRECT: neither of these references is picked up with Trace Dependents and Go to Dependents. We covered these two shortcuts in Chapter 3 as great tools for following the logic of your sheet. If cell B5 contains a function referencing cell A3, using Trace Dependents from A3 points to B5. However if B5 uses OFFSET or INDIRECT to reference A3, such as `=OFFSET(A2,1,0)`, then B5 is no longer a dependent of A3 (it's a dependent of A2). Be aware of this inconvenience.

5: Advanced Formulas

Named References

A named reference is a cell or range of cells that you gave a specific name to. Simply put, named references make your formulas easier to understand. If you have a formula in cell D2 that references a discount rate, instead of using its address (such as F1 or $F$1) you can reference it as DiscountRate:

```
=(1+F1)^2
```

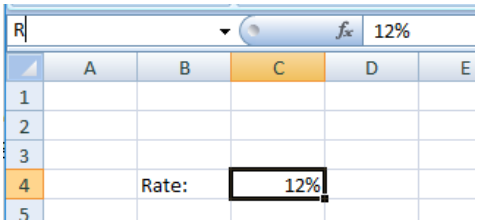
can be replaced with:

```
=(1+DiscountRate)^2
```

The second version is a much more readable formula. It's also easier to enter, without having to remember which cell had your discount rate! (And if you reference cells within VBA macros, always use named references instead of addresses like F1; you'll see this in Chapter 8.)

Naming Cells and Ranges

To name a cell (or range of cells): click inside the little white box just above the A and B column headings, and start typing. Press [ENTER] to register the new name for your selection. Here's a little screenshot taken when I started typing Rate:



Alternatively, for keyboard fans: [CTRL] + [F3] to bring up the Name Manager, then [N] to add a new name. Type in the name and press [ENTER].

Valid cell names must meet all of these requirements:

- Start with a letter or underscore (_N, MyName, New_Rate, etc)
- No spaces or special characters except underscore or period
- Can't be an actual cell's address like YZ123

Named references are not case sensitive, so "Rate" and "rate" are the same reference.

Note also that periods are allowed, but I would seriously discourage using them. Any other programming language (including VBA) would complain about periods in variable names. Underscore is a safe special character for separating words (such as: New_Variable, Sales_2016).

Inserting Named References into Formulas

Once you've named some cells, let's use them in formulas. The easiest way: start typing the name! Excel shows an auto-complete popup. Once you've typed "di" Excel gives a pop-dropdown with two choices: DISC the function, and DiscountRate the named reference. Use [DOWN] to select DiscountRate and [TAB] to auto-complete (you can type out the full name but [TAB] speeds things up).

Alternatively, while entering a formula, press [F3] to bring up a little window with named references. Use arrows and [ENTER] to make your selection. I've never used this shortcut in real life (it's pretty easy to just type the name of the reference), but you might find it useful.

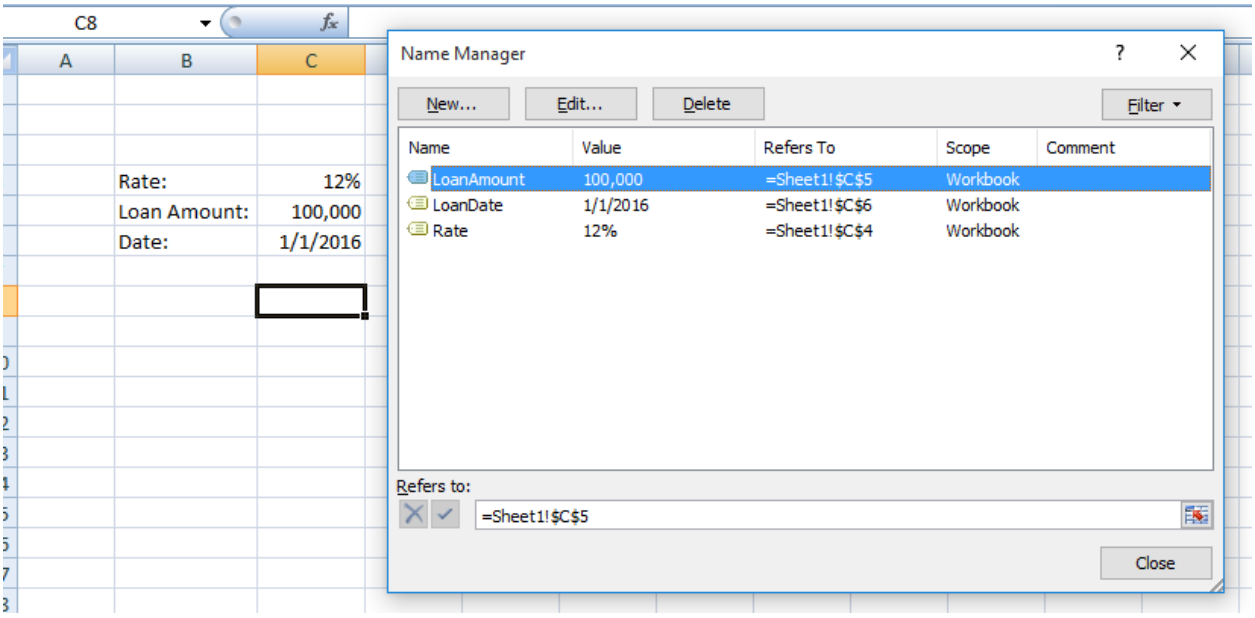
Finding Existing Named References

OK, so you've got some cells named. Maybe you used them in some formulas too, but how do you know which cells the names reference? Where did they go? You have a few options:

- Remember that little box above the A and B column headings? Click the down arrow at the right of it, and you'll see a full list of named references. Click one to go to it.
- Hit [F5] to bring up the Go To window with named references. Click one of them to go there.
- Enter the named reference in a new cell formula, and use trace precedents ([ALT] [T] [U] [T]) to show an arrow pointing from the named cell. It's nice and visual! (We covered Trace Precedents in Chapter 3.)

Name Manager

Use the Name Manager ([CTRL] + [F3]) to review all of your existing named references. All of your references are there in alphabetical order, with the following details: the name of the reference, the value of the cell (or values if it's referencing multiple cells), the address of the cell or range under Refers To, and the Scope. This is all useful information.

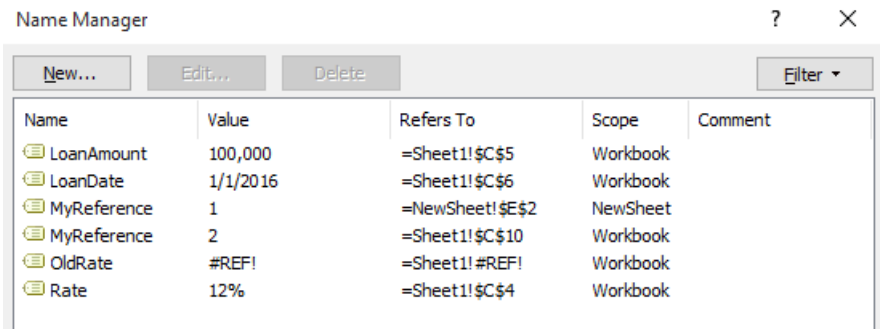


You can remove a named reference by selecting it and clicking the Delete button (or press [DEL]). Doing this does not delete or change the value of that cell; you just won't be able to reference it by its name anymore.

To change a named reference, click Edit. A new window comes up that lets you make changes to the name itself, or to the cell(s) it references.

A quick note about editing the Refers To box: the formula in this box can quickly get messy while editing. If you use arrows inside this box, Excel thinks you're trying to select cells on the sheet. If your formula gets messed up, cancel editing (back out with [ESC] a few times) or just clear the whole field and start over. We'll talk about this formula editing box in Chapter 6.

Finally, you have the "scope" of a named reference. By default the scope is "Workbook", meaning you can reference this cell from any sheet in the same Excel file. This is how it should be! Avoid references with the scope of a particular sheet (such as "NewSheet"): you might end up with two different cells with the same name, one with a Workbook scope and one with a worksheet scope on NewSheet, as in the screenshot below:



Notice there are two entries for MyReference in the Name Manager. Now, the formula `=MyReference` refers to a different cell depending on where you put it! It has a value of 2 if you're using it on NewSheet, and a value of 1 if you're using it anywhere else in the file. It's a bad idea to have this confusing situation in your Excel files.

Advanced Tip

Keep your Name Manager clean. Fix any #REF! values (delete the named reference) and scope issues. Use consistent cell naming.

How does a scope issue happen in the first place? Usually, by copying a sheet to another Excel file, which brings over the source file's references as well. If the two files both had a named reference (MyReference), you now have a conflict! The new name from the new sheet gets transferred over with a sheet-level scope, alongside the old name with the workbook-level scope.

How to find and remove scope issues? Open the Name Manager and sort your names by Scope (click the Scope heading to sort by it). Find anything that doesn't have a workbook level scope, and delete the reference (you may want to first check that there's a workbook level scope with the same name, just in case). Everything should be back to normal after that. If something goes wrong and you get a #NAME error in your sheet (meaning you have a formula referencing a name which you deleted from the Name Manager), you can undo the deletion, or re-create the missing names.

Sometimes you'll see #REF errors in the Value and Refers To fields of the name manager, such as the reference to OldRate in the screenshot above. This happens if you delete the cell it referred to. You should delete all the names with #REF! (nobody likes a messy Name Manager); sort by Value to find all the #REF! ones and delete them all at once.

One last tip about named references and the Name Manager: choose a consistent naming convention. Since you can sort your names alphabetically, name your ranges so that similar things go together. For example you might have cells for a company's sales in states CA, AZ, and NV. You could name these CA_Sales, AZ_Sales, and NV_Sales, but then they would be pretty far apart in the Name Manager. Consider naming them Sales_CA, Sales_AZ, and Sales_NV. Now they'll be listed together alphabetically. A bit of planning will make a long list of named references easier to work with!

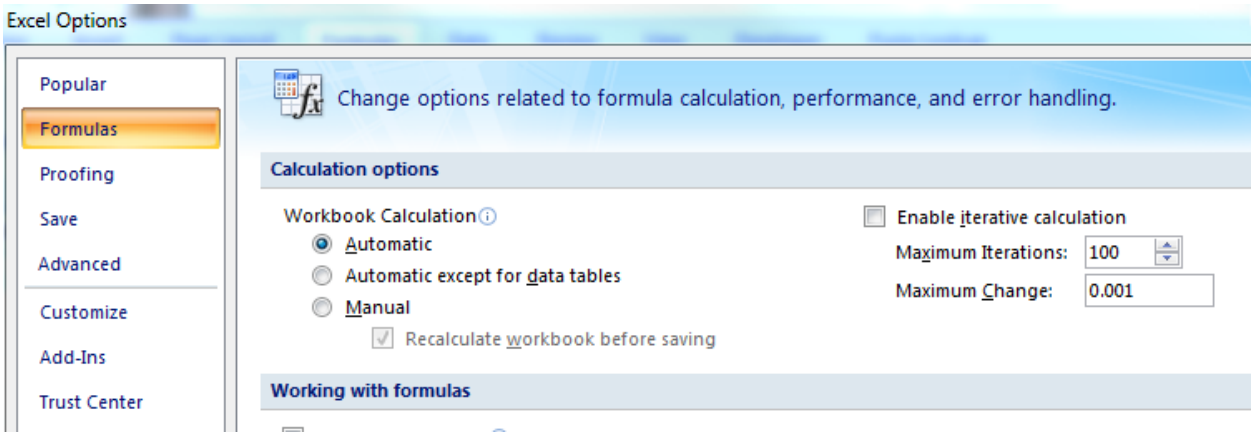
Calculation Options

You'll find yourself changing the Calculation option in Excel more frequently than any other setting. By default, Excel uses Automatic calculation mode, which means all formulas are automatically re-calculated every time you make a change in the file. That keeps all of your cells showing the correct information, and it works great most of the time.

But if you have a large file (with a lot of data and a lot of formulas, especially VLOOKUP formulas) you will notice your entire Excel getting very slow. Every time you change a cell, it recalculates the whole workbook and could take several seconds to do so.[28] Unacceptable. To make your file more manageable, try switching to Manual calculation mode. That way, nothing gets recalculated unless you press [F9].

The Workbook Calculation setting is under Excel options, Formulas:

- In Excel 2013: go to File, Options, then the Formulas tab, Calculation Options section.
- In Excel 2007-2010: click the circular button on the top left and go to Excel Options, then the Formulas tab, Calculation Options section.
- In Excel 2003: click on Tools, Options, then the Calculation tab.



Manual calculation mode takes some time to get used to. You'll notice formula results that make no sense. You'll copy a formula and see the value of the original cell. You'll change things and nothing reacts properly.

[28] Technically, it recalculates only dependents and "volatile functions", not literally every calculation. But for complex workbooks, it's still a very large number of calculations even if you only change one isolated cell.

Don't be alarmed – just **press [F9] to recalculate**! If you don't like it, you can always switch back to Automatic calculation.

The middle setting, Automatic Except for Data Tables, is a good compromise. Most formulas will recalculate automatically, but big data tables will require [F9] to refresh. Of course, this setting only makes a difference if you have data tables (see Chapter 6).

Enable Iterative calculation is another useful setting, which we'll cover in the next section on circular references.

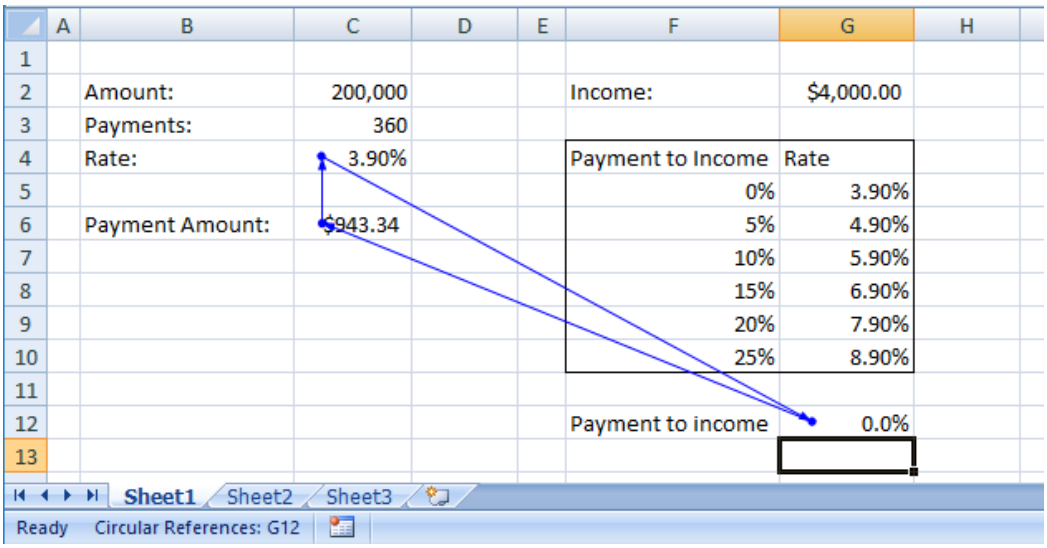
Circular References

When working with complex sheets such as financial models, you will eventually encounter the dreaded circular reference. This error comes up if you have one cell driving the value of a second cell, but that second cell also determines the value of the first cell. Excel doesn't know how to handle such a situation.

The simplest example is this: A1 has the formula `=B1+1`, and B1 has `=A1+1`

Most circular references are not this obvious, and there can be several other cells between these two. You might have A1 determining C10, which determines another cell on Sheet2, which determines a function back on Sheet1, which influences the original A1 through an IF function. The point is that you've come full circle, and you'll need to fix it.

Click OK when you get Excel's Circular Reference Warning, and you'll get arrows showing the circularity (it also opens the help menu, but you don't need that). The arrows look like trace dependents arrows, and they show the order of the circular logic.



The status bar at the bottom of the window now says "Circular References: G12" (or one of the other cells). The problem involves G12 (the payment amount calculation), but it flows through multiple cells: C4, C6, and G12.

If you lose the trace arrows for circularity, go to the Formulas ribbon, click the dropdown arrow for Error Checking, then Circular References. (In Excel 2003 you need the circular references toolbar under Menu, View, Toolbars.)

You should address circular references right away; otherwise, a large number of calculations on your sheet will no longer make sense. Excel will stop recalculating anything linked to the circularity.

Most commonly, you'll choose to eliminate the circular reference. As a first step, think through the situation to see if you can substitute an approximation, or slightly different logic. In the example above, the issue is that payment determines payment-to-income, which determines interest rate, which determines the payment – it's circular! You could substitute some constant interest rate to calculate an approximate payment for payment-to-income, which would remove the circularity.

In some cases you want to preserve the circularity, and try to converge on a single solution by iterating the circle many times. This situation comes up in financial models. For example: the amount of debt a company borrows determines the amount of their interest payments, which determines the amount of cash flow they have toward paying down debt, which determines the amount of debt they can borrow. It's circular, but each calculation converges on a single solution!

To turn on iterative calculations in Excel, open Excel Options, Formulas, and Calculation options. Check the box for Enable Iterative Calculation. The default setting of 100 iterations is usually fine; it will cause Excel to run the calculation 100 times (the whole thing should only take a split second).

Iterations will not work for the simple example I gave you at the beginning, where A1 is B1+1 and B1 is A1+1. In that case Excel will increase both values during each iteration, and you end up with 199 and 200. There isn't a single solution to converge to, so in this case the solution is to change the formulas and eliminate the circularity.

Formula Errors

Excel can generate some scary-looking errors, as I'm sure you've seen before. You might delete a seemingly empty row, and your entire sheet fills up with #REF! errors (quick, undo!). In this section we'll look at the types of Excel errors, what they mean, and how to avoid them.

Keep in mind that while errors all look ugly, Excel is trying to tell you what's wrong. Don't ignore the kind of error you get; the different codes mean different things. They are:

- #DIV/0! (divide by zero)
- #NAME! (unrecognized named reference or function)
- #VALUE! (mathematical operation on a non-numeric field)
- #NUM! (invalid number used in a function)
- #N/A! (common result of VLOOKUP and MATCH functions)
- #REF! (invalid reference)

Notice that errors propagate forward in Excel. If I have a divide by zero error (#DIV/0!) in cell C2, any cell that references C2 will have the same error! Then cells that reference those cells will also be #DIV/0!, and so on. To find the original cell that caused the error, use trace precedents and follow the red arrows.

Advanced Tip

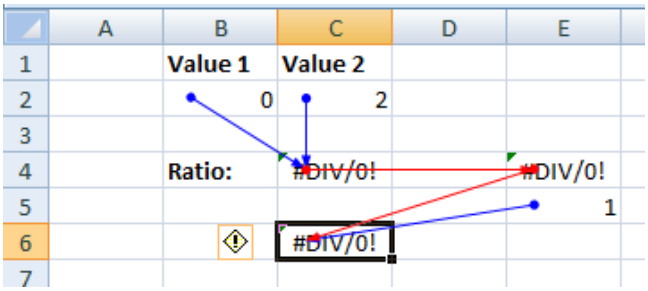
Use [ALT] [T] [U] [T] and follow the red trace dependents arrows to find the original source of the error.

To use trace precedents in error checking, do this: click on a cell with the error, and click Trace Precedents in the Formula ribbon under Formula Auditing.[29] Or use the keyboard shortcut: [ALT] [T] [U] [T].

You'll see one or more arrows pointing to your cell. The red arrows indicate an error is being passed through. Follow these red arrows back to the original cause of the error!

In this screenshot, we are tracing the error back from cell C6. Keep clicking Trace Precedents and follow the red arrows to the source; in this case, C4 was dividing by the value of B2, which is 0:

[29] In Excel 2003 it's under Tools, Formula Auditing.



#DIV/0! Error

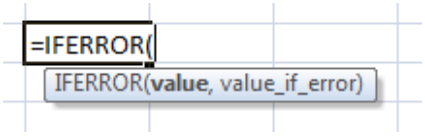
This is one of the easy ones. Does your formula involve dividing by zero? For example: `=A1/A2`, where A2 is either blank or zero. If this is the case, you'll get this error.

The solution: use some kind of error handling to return the result you want in cases where the cell you're dividing by has a zero value. For example, to return 0 when dividing by 0:

```
=IF(A2<>0, A1/A2, 0)
```

#NAME! Error

Name errors usually result from typos, deleted named references, or text without quotes in your formulas. When you type some text into a formula (like this: `=IFEROR`) Excel checks if this is a valid function or an existing named reference. In this case, IFEROR is a typo for IFERROR; you'll just have to fix the spelling. It helps that Excel brings up a little hover menu when you are typing a valid function, like this:



Another cause of the name error is a deleted named reference. Maybe you have a cell with the formula: `=myinput`, referencing a named cell called myinput (we talked about named references earlier in this chapter). If myinput does not exist, you'll get a name error. Hopefully you didn't delete the name in Name Manager! The solution is to once again define a cell with that name, or to change your formula.

The last potential cause of a name error is that you are trying to output a string (meaning numbers and letters) in something like an IF function, but you forgot the quotation marks. This would be an error:

```
=IF(A1=open, yes, 0)
```

Replace it with this:

```
=IF(A1="open", "yes", 0)
```

#VALUE! Error

Value errors are typically caused by doing a math calculation with text input.

For instance, you might have the formula `=C4+1`, but C4 contains a text value, so you get a #VALUE! error. The solution: don't use math formulas on text inputs.

But what if C4 looks like it contains a number? The error is a giveaway that you actually have a number stored as text. You can tell when this happens because the number is left-aligned by default, indicating that Excel thinks it's text. Excel right-aligns numbers but left-aligns text.

The solution: convert your number stored as text to an actual number. If Excel gives you a popup, click Convert to Number. Go back to Chapter 4 if you need a refresher on data types, including how to convert an entire column of numbers stored as text into their actual values.

#NUM! Error

This error is typically returned by financial functions such as IRR, or by certain other functions if you pass an invalid value as an input. The solution is to check the inputs into the function; often the problem is that you are passing a negative or zero value where it is not allowed. For example, `=LN(0)` attempts to calculate the natural log of zero, which does not exist. The result is a num error.

If you don't want to change your formula, use an IFERROR around the original function to output a custom result in place of the error.

#N/A! Error

This error is typically returned by VLOOKUP and MATCH functions, which were covered in the previous chapter. For those functions, #N/A means there was no match found. The solution is either to modify your formula (for example, expand the lookup range), or to use IFERROR.

#REF! Error

This error happens when you delete a cell which was referenced in a formula. Since you still needed that cell, the formula is now broken. The result is #REF!

The ref error is the worst to deal with. All of the other errors above have a simple fix: find the original formula with an error, and fix the formula or the cells that feed into it. You don't have to guess what caused the error. Ref is particularly nasty because the word "#REF!" gets substituted in place of the old cell references, so there is no obvious easy fix! Ref errors have no place in your formulas, and you should fix them immediately.

Here's a quick example of how the error comes up:

1. Start with a formula in A1: `=B1+C1`
2. Now delete column C
3. A1 now reads: `=B1+#REF!`

C1 no longer exists, so Excel had to do something to your formula. It put #REF! in place of the old C1.

The worst part is, there's not an obvious solution for replacing the #REF! part. I certainly can't give you a magic solution. Hopefully you were paying attention and noticed this formula used to reference C1, which should never have been deleted. Undo the deletion, and move C1 to D1 before you remove column C.

When you first notice a #REF! error, use the trace precedents technique to trace back to the original cell(s) causing the error. Alternatively, use [CTRL] + [F] to find the text "#REF" in the sheet, which takes you to cells with errors in the formula. You'll need to fix these with new correct references.

Since fixing #REF! errors is such a pain (unless you can quickly undo), the best solution is prevention. Best practices for preventing #REF! errors:

- Notice the error immediately. If you just deleted some cells, undo! You probably deleted something important.
- Before deleting a blank row (or column), make sure it's really empty! Use [CTRL] + [LEFT] and [CTRL] + [RIGHT] on that row. If the row is truly all blank, these shortcuts take you between the first and last cells. Otherwise, you land on some non-empty cells in-between. Use trace dependents to make sure these cells aren't used in any formulas before deleting the row.

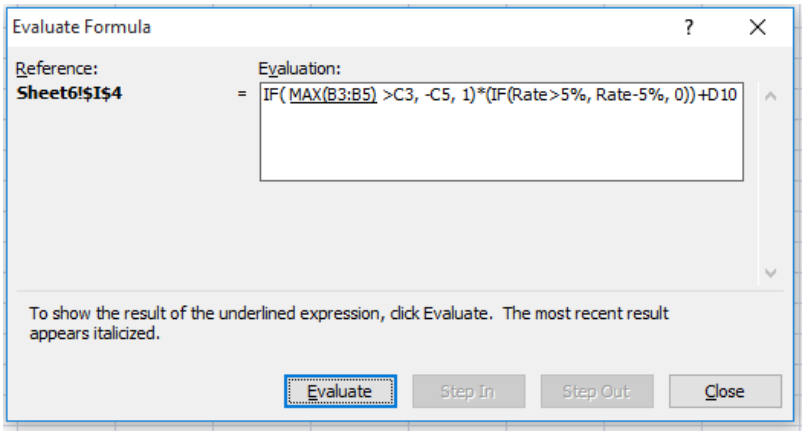
- Create a new cell to serve as your “REF error checking cell”. In this cell, enter a formula like this: `=IF(ISREF(SUM(A:Z)),"REF!","OK")`. This formula sums up everything on your sheet (up through column Z), and displays the message “REF!” if there are any REF errors and displays “OK” otherwise. If you have persistent problems with REF, it may be worth making this little helper cell. Keep an eye on this cell, and if it no longer says “OK”, then undo whatever you just deleted.

Debugging Formulas

If you have a long and complicated formula, it’s easy to lose track of exactly how it works. Take a complicated formula like this:

```
=IF( MAX(B3:B5) >C3, -C5, 1)*(IF(Rate>5%, Rate-5%, 0))+D10
```

How do you test whether it’s working properly? One good tool is Evaluate Formula, which you’ll find under the Formulas ribbon near the Trace Dependents and Trace Precedents buttons. It brings up this popup:



Next, you can repeatedly click Evaluate to see how Excel calculates each step of the formula. It’s a good way to see if something is not behaving as intended.

Another way to evaluate and debug your formulas: use the [F9] key. **Highlight a section of your formula and press [F9] to make Excel calculate just that part of the formula** (not the entire cell). Keep in mind you need to highlight entire sections of code that make sense on their own!

Take a formula such as the above:

```
=IF( MAX(B3:B5) >C3, -C5, 1)*(IF(Rate>5%, Rate-5%, 0))+D10
```

Highlight the section `MAX(B3:B5)` and press [F9] to get the result of just that piece of code. If you select just `MAX(B3:` and press [F9], Excel gives you an error because that snippet is not a valid formula.

But the best way to debug a long and complicated formula is to convert it into multiple short and simple formulas! Break it apart into multiple cells, and you can see all the results of all the interim calculations at once. Excel is a great visual calculator, so use it! We’ll talk more about simplifying formulas in Chapter 7.

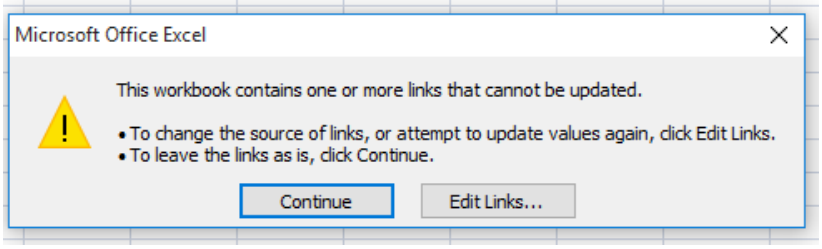
External References

External Reference means you are referencing a cell in another file. There may be legitimate uses for these, but be careful to avoid unintentional external references in your files. The danger is that your Excel sheet now has a dependency on another file – and who knows what that file is doing, or whether it will always be accessible later!

It's very important to be aware of all external references in spreadsheets – they can cause unexpected behavior, and can even cover up malicious practices.

> Consider the example of a rogue trader at Allied Irish Bank in 2002: this trader apparently "circumvented controls by manipulating spreadsheet models used by the bank's internal control staff". Specifically, he set up this risk control Excel sheet to pull data from a file on his personal hard drive, rather than from a Reuters data feed[30] as required. Could something like this happen within your own organization? If you consistently seek out external references in critical spreadsheets, these risks should be mitigated.

When you open an Excel file with external links (especially if that file was emailed to you), you'll often see this error: "This workbook contains one or more links that cannot be updated." The two choices you're given are not that helpful: Edit Links pops up a window telling you that somewhere in this file you reference another file, but it doesn't tell you which cells contain external references.



The solution: click Continue and ignore the Edit Links window. You'll instead use Find to **remove external references**. Fortunately, all external links have a recognizable format, such as:

```
=[Otherfile.xlsx]Sheet2!B1
```

Press [CTRL] + [F] to open the Find popup. Use the Find Options to choose Within Workbook instead of Within Sheet. Now you're searching the entire Excel file. **Search for ".xls" to find any references to external file names**.

You can also search for "[" (just an open bracket) as a backup. But open bracket also finds any formulas that reference data tables, so it's not a perfect solution.

If you think you've found and eliminated all external references, close and re-open the file. If the external links message no longer comes up, you've succeeded. If Excel says you still have external links, you definitely have them (even if you can't find them!). Try looking in two more places:

- Open the **Name Manager** and look for any names pointing to outside files. The Find feature does not look at named references.
- Edit the **data source of all your charts and pivot tables**, and see if those point to external data. We'll cover pivot tables in the next chapter.

[30] "The Role of Spreadsheets in the Allied Irish Bank / Allfirst Currency Trading Fraud". Raymond J. Butler. http://arxiv.org/ftp/arxiv/papers/0910/0910.2048.pdf

6: Data, Pivot Tables, and Charts

Excel is very useful for analyzing data, especially when that data is well organized. These are my golden rules for working with data most effectively:

Data Rule #1: Organize your data. For best results, the first row should contain column headings. The second row should contain the first line of data. In other words: don't enter headings into multiple rows, and don't skip rows above or below the heading. Delete any unused blank columns or blank rows. Finally, convert the data set into a data table.

A pair of screenshots should illustrate my point. This is well-organized data:

	A	B	C	D	E	F	G
1	Record ID	Account Num	Balance	Limit	Last Payment Date	Last Payment Amount	
2	1	1001	2500	10000	1/4/2016	100	
3	2	1007	2750	10000	2/5/2016	150	
4	3	1009	1100	5000	2/7/2016	458	
5	4	1010	100	5000	3/1/2016	147	
6	5	1011	0	7000	12/1/2015	10	
7	6	1012	2880	4000	3/10/2016	45	
8	7	1015	5000	5000	3/15/2016	740	
9	8	1017	40	5000	2/15/2016	250	

This is not data; these are just numbers in a model, including formulas and calculated results:

	A	B	C	D	E	F
1						
2			**2010**	**2011**	**2012**	**2013**
3		Revenue	1500	1540	1600	1650
4		Costs	855	924	864	925
5		Gross Profit	645	616	736	725
6						
7		Operating Costs	175	180	240	220
8						

Data Rule #2: Reference the entire column of data. If your data has 100 rows in column A, avoid formula references such as SUM(A2:A101). There is too much room for error in these partial references, especially if your data set grows with more rows. Better to reference SUM(A:A) for the entire column.

Date Rule #3: Use consistent formulas in a column. In other words, column C shouldn't have B2*2 in row 2, and B2*4 in row 3, or other types of variation. If necessary, add a new column to store your 2, 4, and other constants; the formula in column C should be consistent. Anchor references, add additional columns for calculations, or use OFFSET and IF and VLOOKUP where necessary. Avoid inconsistent formulas buried among your data.

Data Rule #4: Use pivot tables effectively. They are much more effective than simple filtering and sorting. You'll see what I mean shortly.

And as a bonus tip: **Learn keyboard navigation** to navigate between the beginning and end of your data set. You'll work much more effectively with large data sets if you don't get bogged down with mouse scrolling.

The rest of this chapter will be easier to follow if you have some data to work with. If you need an example data set, go to this book's website, www.AdvancedExcelBook.com, to download the Chapter 6 Sample Dataset.

Cleaning Up Data

When you receive a data set to analyze, you'll want to make sure it's cleaned and formatted correctly for proper analysis. Some common things to watch out for:

Headings in one row only. Your original data might contain column headings that span multiple Excel rows. They may look nicer, but multi-row headings will not work with data tables and pivots; hence, Data Rule #1. To convert headings in rows 1 and 2 into a single-row heading, follow these steps:

1. Insert a new row as row 3 ([ALT] [I] [R])
2. Use a formula in row 3 to concatenate rows 1 and 2 into one row, with a space in between: `=A1&" "&A2`
3. Fill this formula all the way right across the rows ([CTRL] + [R] with the proper range selected)
4. With the new row selected, copy and paste-special values to replace formulas with values ([CTRL] + [C] then [ALT] [E] [S] [V])
5. Delete the original header rows ([CTRL] + [-])

At this point you might object that the column headings are too wide, and would look better in two or three rows. No worries! While typing inside a header cell, use [ALT] + [ENTER] to force a new line break. You could also select the entire header row, press [CTRL] + [1] to open cell formatting, and select Wrap Text under the Alignment tab. Now your headings look better on the screen, but still only occupy a single Excel row.

No empty columns. Your data should be completely contiguous. That means you need to get rid of any empty columns between data columns. I've seen some people use blank columns as a means of spacing and formatting – it may look nice, but there are two significant drawbacks:

1. Using [CTRL] + [LEFT]/[RIGHT] gets stuck on the empty columns. This slows down your ability to navigate through the entire sheet.
2. Your data set will not work with pivot tables. Excel throws an error if you try to create a pivot table with empty columns.

No blank headings for any column. All data columns should have a header. Excel will not let you create a pivot table if you have a column without a header. The only exception would be for data used to create charts. For charts, leave the header of the first column blank; this way Excel knows that column is the X-axis and not a data set.

No skipped rows and/or subtotal rows. Some well-meaning folks might add in subtotals, totals, or other extra lines below or between your data. For example, you might have 100 transactions in 2014, followed by a subtotal line for the year 2014, followed by more transactions for 2015. You have to get rid of these extra subtotal lines! This is not data. If you add or count things in the entire column, you'll get the wrong answer. Subtotals belong in pivot tables or report summaries, not mixed in with data.

Check for typos and trailing spaces in data fields. Sometimes you'll have sloppily inputted data, especially if some of it was entered by hand (anytime humans manually enter large amounts of data, they will make mistakes; don't expect otherwise). In these cases, watch out for obvious typos and not-so-obvious ones. In particular, I recommend checking for trailing spaces such as "New York " instead of "New York". Use the Excel function TRIM to get rid of excessive spaces at the beginning or end of data fields. Then replace the original data with the trimmed data.

Use consistent formulas in each column. If a column has formulas, check to make sure the same formula is used for all rows in that column! This is a repeat of Data Rule #3. Don't intersperse any hard-coded numbers, formula exceptions, and so on, because it's very easy to miss these later on and make a mistake. You should be able to handle all special cases and exceptions with IF functions and additional columns for calculations. Fortunately, Excel is pretty good at warning you about inconsistent formulas with a little green triangle, like this:

	A	B	C
1	ID	Amount	Amount2
2	1	100	200
3	2	120	360
4	3	140	280

In this case, C2 and C4 multiply column B by 2, but C3 multiplies by 3. Excel is warning me to fix it!

Filtering and Data Tables

Most Excel users know how to filter data: click Data, then Filter. This adds little dropdowns in each column heading. Then you can choose to filter (show or hide) values by using check boxes or numeric filters such as "greater than 100".

We won't spend much time on filtering here; I prefer to show you pivot tables, which are more powerful in every way. There are two very significant problems with filtering you should beware of:

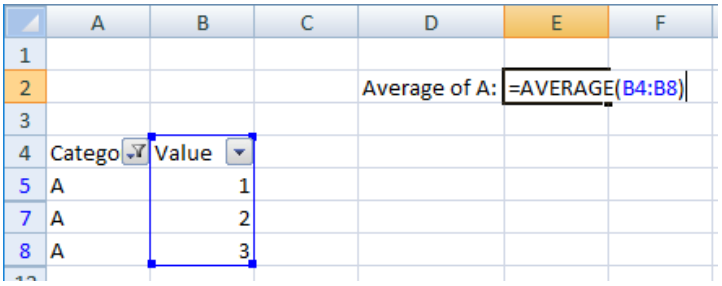
Advanced Tip

Be careful with formulas applied to a filtered column! Averages, sums, and so on will not exclude rows you filtered out. Your formulas will give the wrong result. Use Pivot Tables instead of filters.

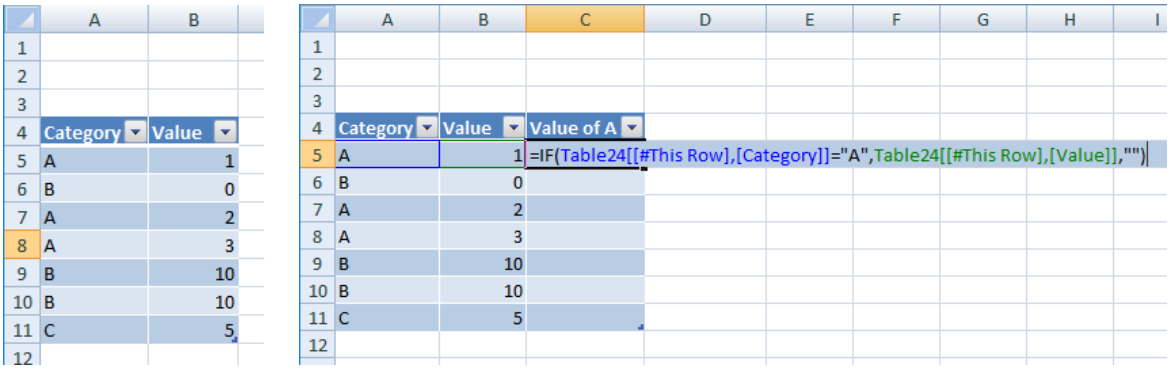
- Filtering does not recognize text with extra spaces as unique values. In your filter dropdown, "New York" and "New York " would appear to be the same value, although one has a trailing space. These values are actually different for the purposes of equals formulas, or COUNTIF. Pivot tables recognize them as distinct values.
- Hiding a row by filtering it out does not exclude it from SUM, AVERAGE, COUNT, or any other function on a range. This is an easy mistake to make! You want the average of items in category A, so you filter for that category and take an average of the values column. Wrong! The average

function does not depend on filters or hidden rows: it just takes the average of the range you provided, which includes all the rows in the data set.

Here's a screenshot to illustrate the second point: A4:B8 contains some data, filtered for category A. The other categories are now hidden, but they do have values. The function `=AVERAGE(B4:B8)` doesn't know and doesn't care that you've filtered out row 6! It returns the average of the entire range, not category A. The result is not 2.



If you followed Data Rule #1 (organize your data), you can convert your data into a **data table**. Select the entire data set (all the rows and all the columns, including headings), then click Insert, Table ([CTRL] + [T]) to convert your data into a data table. This makes everything easier to work with: **filtering** is automatically added, and these cells now belong together and are shaded blue.



Working within data tables has some convenient features. Above, you'll see in the second screenshot that I added a new column with heading "Value of A". As you type a formula into C5, references to A5 and B5 are replaced with things like `Table24[[#ThisRow],[Category]]`, which is Excel's notation for data tables. It's quite a bit longer than a regular formula with A5 and B5, but also very readable! As an extra benefit, when you press [ENTER] the entire column C fills with this formula.[31]

Now you can get the average of category A's value by taking an average of this entire newly created column. If you type the AVERAGE formula in a cell outside the data table, as soon as you start selecting the input data you'll once again get some data table notation:

```
=AVERAGE(Table24[[#All],[Value of A]])
```

I think this notation is a good feature. It takes care of anchoring automatically and helps avoid accidentally referencing the wrong column. You don't have to worry about remembering what data is in column A or B or C; the formula uses the data headings.

Don't be afraid to create several columns with filtered data using IF, like the above example. Excel gives you plenty of columns to work with.

---

[31] Excel knows you want a consistent formula for the entire column – Data Rule #3!

You might be wondering, how do you know whether the table you created is called Table1, Table20, or Table-whatever? A quick way to find out is to go to a blank cell and type a formula, referencing one of the cells in that table. Excel will use the name of the table in your formula.

If you have a formula with a table name but don't know where the table is, use Trace Precedents to find it! Press [ALT] [T] [U] [T] in order to draw the trace precedents arrows.

Duplicate Data

For the purpose of this section, consider a bunch of data in column A. How do you **highlight duplicates** in this data? In Excel 2007 and newer, you have access to some great Conditional Formatting options under the Home ribbon. Select your entire data column, choose Highlight Cells Rules, then Duplicate Values. You'll get a popup with a few choices of how to highlight. That's it!

Excel 2003 doesn't give you this built-in conditional formatting. You still can use conditional formatting with a formula. To use it, select just the first cell (it might be A2 if there's a heading in A1), choose conditional formatting, then choose Formula. The formula is this: `=COUNTIF(A:A,A2)>1`

Note on correct usage of this formula: A2 in the formula above is the first cell in the range you selected for conditional formatting. If you select just a single cell for conditional formatting, the formula will not work. If your selection starts with A1 or A3 or another cell, your formula will not work. Be careful to set up conditional formatting correctly!

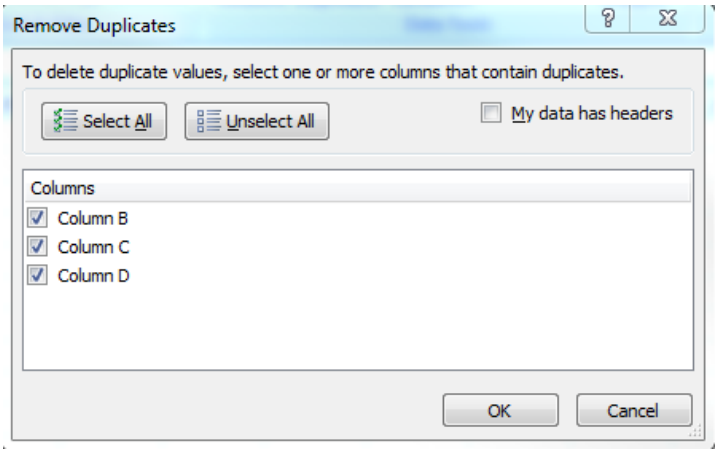
What's the fastest way to **see if a column of data contains any duplicates**? You probably don't want to use conditional formatting and then have to scroll through to find highlighted rows (assuming there are a few hundred entries).

I recommend using COUNTIF in another (adjacent) column. If column A has your data, use `COUNTIF(A:A,A2)` in row 2, and fill that formula down the entire column. Then select the entire column of COUNTIF results and check the max, min, and average results in the status bar. If your average and your max is 1, there are no duplicates! If you do all this with the keyboard, it shouldn't take more than 10 seconds total (see Chapter 2 for a refresh on keyboard navigation)!

Note: If you don't reference the entire column A:A but just a range, remember to anchor the rows like this: $A$2:$A$1000.

To **remove duplicates**, you could follow the same process with COUNTIF in another column. Filter out rows with COUNTIF greater than 1, and delete those rows. A **faster option** is this: go to the Data tab on the ribbon and find Remove Duplicates. Excel asks whether you want to Expand the Selection or Continue with the Current Selection.

Choose Expand the Selection to delete the entire row of data when a duplicate entry is found. Next, Excel asks which columns to remove duplicates from. I recommend selecting just one of these columns at a time. Excel will scan that column and remove any duplicate entries (keeping just one copy of those entries). Don't forget to check the box for My Data Has Headers if the first row of your selection is the header row.



When finished, Excel tells you how many duplicate values were removed and how many remain. For example, “9 duplicate values found and removed; 345 unique values remain.” This can be pretty helpful information, so don’t ignore it! The message might also say “No duplicate values found.”

Note: The remove duplicates feature doesn’t exist in Excel 2003; in that case, you’ll want to use the COUNTIF method above.

A few more points to keep in mind about removing duplicates:

- Sometimes you only want to remove duplicates if multiple columns all match. For example, column A has states and column B has cities. You only want to remove duplicates if A and B are both the same. The solution: create a new column that combines (with CONCATENATE) the value of column A and B. Then you can run Remove Duplicates on the new column.[32]
- Removing duplicates can be a nice quick way to delete blank rows in your data set. Let’s say you have hundreds of rows of data, with several empty rows scattered between them. It would be a huge hassle to delete these rows one by one! If there’s a column that shouldn’t have any duplicates in it anyway (besides multiple instances of a blank row), remove duplicates on that column. It gets rid of all blank rows except the first one. Now just clean up that last remaining blank row (use [CTRL] + [DOWN] to find it).
- Remove Duplicates is not case-sensitive. If you have entries with different capitalization, like “John” and “JOHN”, these are considered duplicates by Excel. The second one gets deleted.
- Remove Duplicates might skip cells if they appear to be duplicates, but are not exactly the same. Any leading or trailing spaces in the cell (such as “ John ”) cause the cell to not show up as a duplicate. The best solution is this: add a new column with the formula `=TRIM(B2)`, so the trim function gets rid of these excess spaces. Now run Remove Duplicates on the new column.
- What about identifying values that are “similar” but not exact duplicates? There is no easy way for Excel to do this – approximations and pattern recognition are quite difficult for computers. However, there are a number of techniques for performing what is called “fuzzy matching”. We’ll look at some alternatives in Chapter 9.

Pivot Table Basics

Pivot tables are awesome! If you work with data, pivot tables should be a regular part of your use of Excel. Here are a few things you can do with them, very quickly:

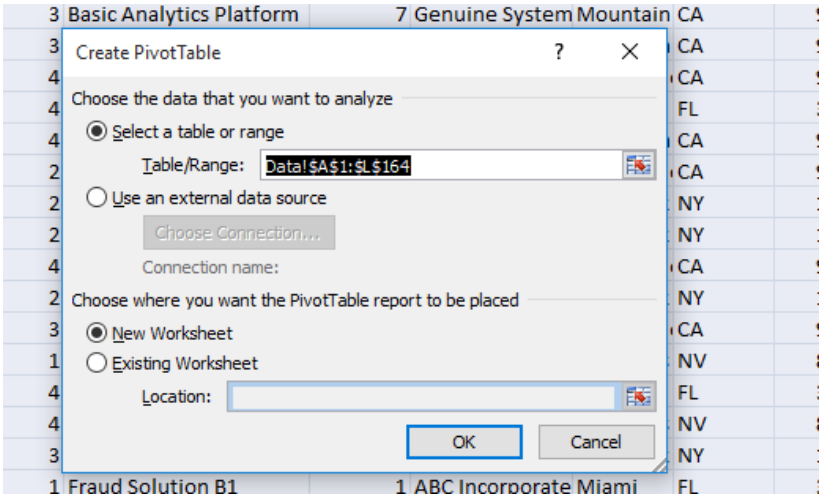
[32] This trick is similar to using VLOOKUP or MATCH on multiple columns. Create a new column with the combined data, and do your lookup on the new column.

- Filter data on one or more fields
- Calculate totals, averages, running totals, percentage distributions, minimums, and more, all with a few simple clicks
- Summarize data by one or more variables
- Organize daily data into months and years, and easily graph the result

Pivot tables themselves are also remarkably easy to use; you'll just need a data set to start! The examples and screenshots to follow use sample data you can download from www.AdvancedExcelBook.com. Each row of this data set represents a sales transaction with a sales rep, a product, and a company name and address, plus the dollar amount of the sale.

Pivot table source data must be organized according to Data Rule #1: a single row of column headings, and no blanks among column headings. The sample data set already meets these requirements.

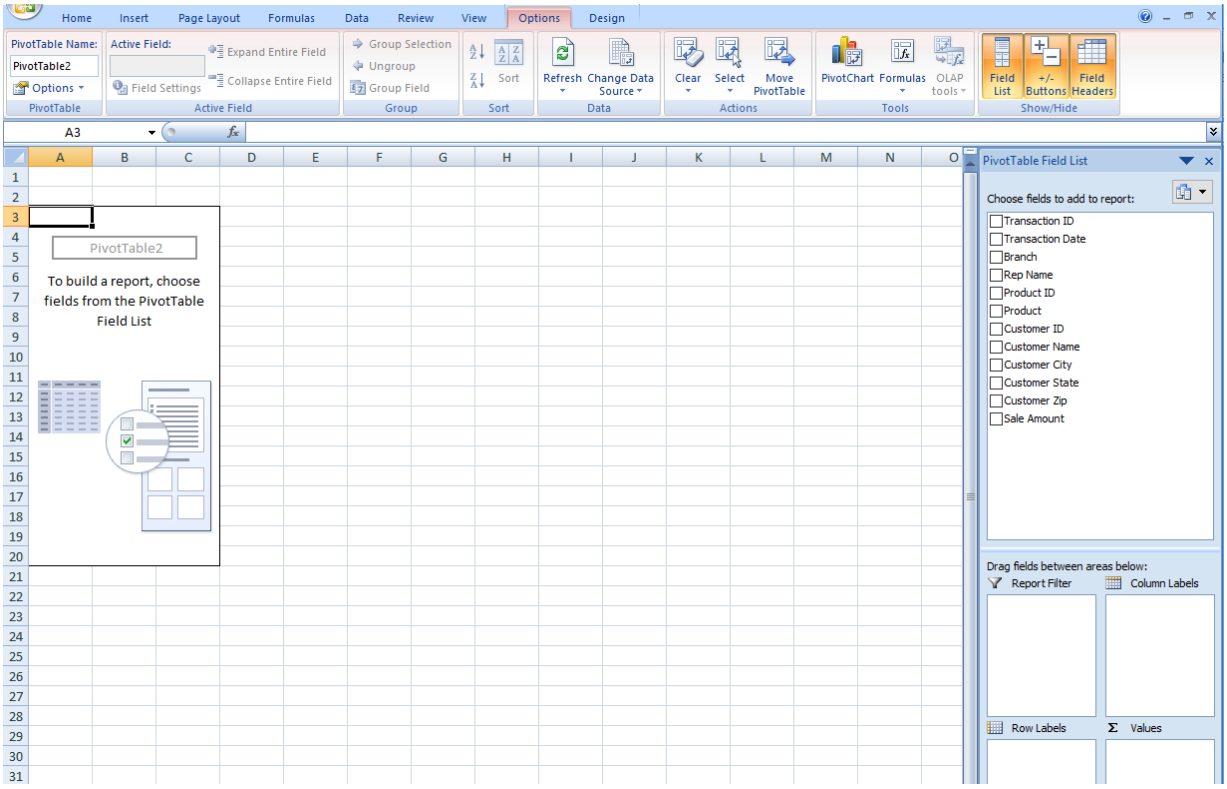
To create a pivot table, select the entire data range (including headings), and click Insert, PivotTable. Excel brings up a popup asking you two questions: the source of the data (Select a Table or Range), and the destination for the pivot (Choose Where you Want the PivotTable Report to be Placed). The source is the data you already selected, which is pre-populated in the popup. The destination can be anywhere as long as it doesn't overlap with other existing cells; usually New Worksheet is a good choice. Click OK to create the pivot.



You'll sometimes get this error:

"The PivotTable field name is not valid"

To fix it, check the first row of your selection, and make sure each column has a heading. Remove any blank columns. Then try adding the pivot table again. If you're now seeing something like this, you're in good shape:



The new pivot table starts out empty. You'll see a new section of Pivot Table Fields on the right, which lists fields (columns) available in the source data. Notice that the fields in this list are your data column headings. There are four areas at the bottom right: Filters, Columns, Rows, and Values. To manipulate a pivot table, you'll drag the available field names into these four areas. The actual pivot table results will show up on the left – right now it's just a big white box with some graphics inside.

Row Labels. Drag a field to the rows section, and it shows up vertically in the pivot table. Drag Rep Name to the Row Labels, and the pivot table lists all the reps along the left side.

You can have multiple fields within a pivot table area. Drag Customer Name to the Rows Labels, below Rep Name. Your pivot results will now be grouped by both rep and customer.

Values. It's time to see some actual results within the pivot table. Drag the Sale Amount field to the Values area. The pivot displays Sum of Sale Amount: the total of the Sale Amount values for each rep and customer.

Note: sometimes pivot values default to Count instead of Sum, such as Count of Rep Name. Count is the number of rows in the original data (same as the COUNTA function). It comes up by default when the data field has non-numeric values. Rep Name values are text, so you can't sum them, only count them.

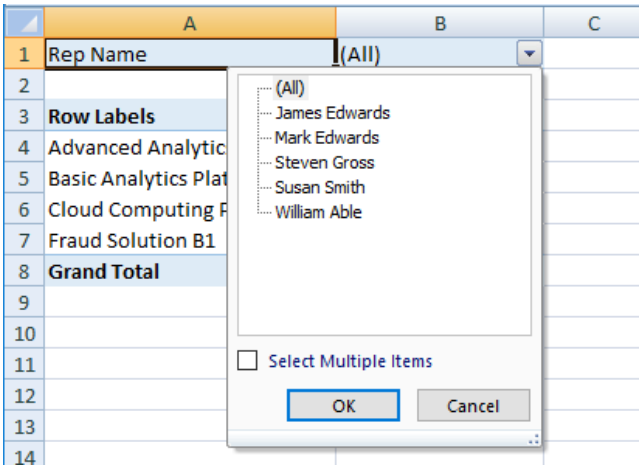
If you still want to change from Sum to a different measure, click on a field within the Values section, then click Value field settings. You can choose Sum, Average, Min, Max, Count, and a few others.

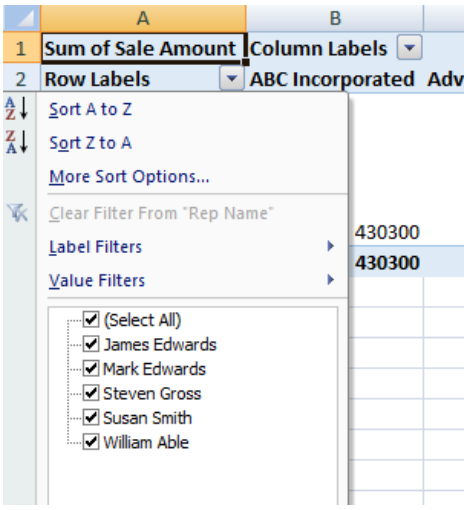
Column Labels. Columns work the same way as rows. Summarize data by columns along the top instead of along the left. You can create two-dimensional pivot tables by using one field along the rows and another field along the columns. Try moving the Rep Name from the Row Labels over to the Column Labels.

Filters. Use pivot table filters with caution. These allow you to restrict the entire data set based on the value of one or more fields. For example, you could create a pivot table that shows sales numbers for a single rep,

with the names of products along the rows. You would put the Rep Name field under filters, and the Product field under rows.

At first the Rep Name filter says “(All)”. Use the dropdown to actually filter for different names, or use Select Multiple Items to check them off one by one. The screenshot on the left shows filtering.





A particular field cannot be in both Filters and Rows/Columns at once. To show data by rep, but restrict the data displayed to only certain reps, keep the Rep Name field as a row label. Use the dropdown arrow next to “Row Labels” in the pivot to start filtering. That’s the screenshot on the right.

In general, you’ll be using the Field List section when you’re building the layout of the pivot table. Drag fields into any of the four areas: Filters, Column Labels, Row Labels, and Values. Then go to the pivot table itself to filter data and manipulate the way values are shown.

Pivot Table Error Messages

As you work with pivot tables, you’ll likely encounter a few common error messages. If Excel gives an error you don’t understand, I recommend just Googling it: “Excel <name of error I don’t understand>”. These are the three most common pivot table errors, and how to fix them.

PivotTable Field Name is Not Valid. This error comes up when your data headings have any blank cells. Add back the missing column headings, or change the data source by clicking the Options section on the ribbon, then Change Data Source. We’ll talk about Data Source later in this chapter.

Pivot Table Cannot Overlap another Pivot Table. The error is caused by a pivot table that is trying to refresh and no longer “fits” on the sheet because another pivot table is in the way. Any sheet that has multiple pivot tables can cause this error. Give your pivots more space (inserting more rows or columns between them), or move them onto a new sheet if they just won’t fit.

If your pivot tables have grouped rows or columns, sometimes Excel “forgets” the grouping when you refresh the data. This causes your pivots to have a lot more rows or columns than before, possibly causing overlap with another pivot.

Cannot Group that Selection. If you’re trying to group values for row or column labels, sometimes you get this very annoying error. The problem is that Excel doesn’t know how to automatically group the data in that column, usually because you have text or error data in that column. There are three possible solutions:

- If your data is not supposed to have text values, you might have to convert numbers stored as text into actual values.
- You can replace some of the text with blank cells; the blanks will have their own grouped category and the numeric values will be grouped properly.
- If you must retain text values, you won't be able to use automatic grouping in the pivot table. Instead, create a new data column with IF or VLOOKUP formulas to group into ranges or categories (we will discuss this later in this chapter under Grouping Labels).

Advanced Pivot Tables

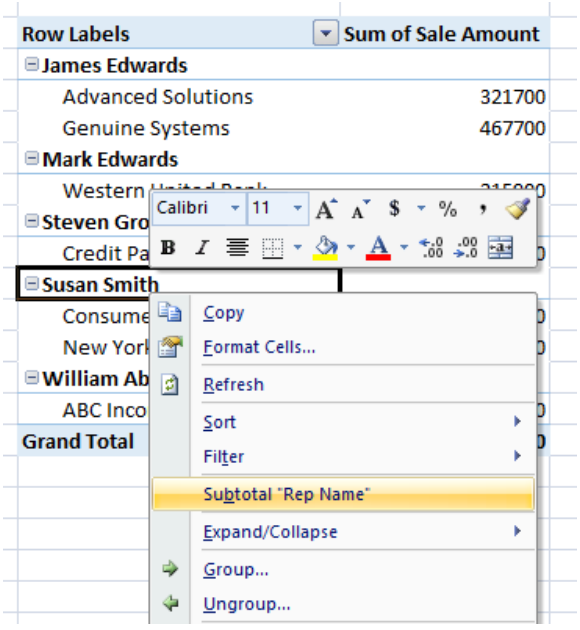
Let's move on to some more advanced things you can do with pivot tables.

Grand Totals and Subtotals

You can choose to **display or hide the Grand Total** from the last row/column. Access this setting by right-clicking the pivot table, and selecting the Totals & Filters tab. There you can un-check Show Grand Totals for rows or columns.

You can also toggle on/off subtotals. In the example below we have Rep Name and Customer Name as Row Labels. In the left screenshot, there are no subtotal values for Reps (you don't have the total sales for James Edwards).

To **add Subtotals**, right-click one of the rep names (such as the cell with "Susan Smith"), and click Subtotal "Rep Name". The subtotals show up in bold like the screenshot on the right. If you still don't see subtotals, you might be right-clicking the customer name instead of the rep, so try one level higher.



Row Labels	Sum of Sale Amount
James Edwards	789400
Advanced Solutions	321700
Genuine Systems	467700
Mark Edwards	215900
Western United Bank	215900
Steven Gross	283700
Credit Partners	283700
Susan Smith	643200
Consumer Technologies Inc	291500
New York Financial	351700
William Able	430300
ABC Incorporated	430300
Grand Total	2362500

Data Source and Refresh

Pivot tables save all the data behind them into a memory "cache". The actual data source behind your pivot is extremely important – that's the full extent of rows and columns that the pivot table uses. Unfortunately, the data source is not very transparent! You can easily make a pivot table mistake by having an incomplete data source, or by not refreshing your data.

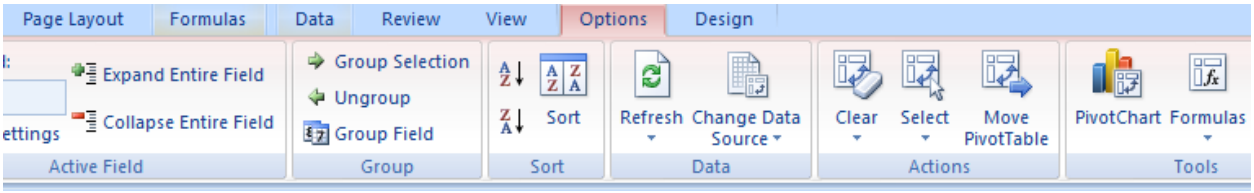
Advanced Tip

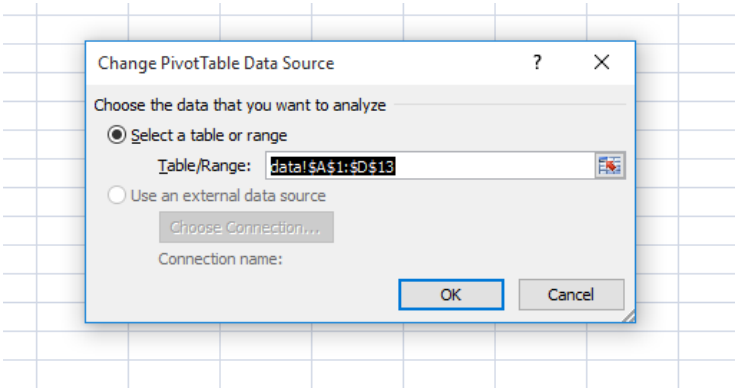
Don't forget to refresh your pivot tables! If anything changes in the underlying data, click the Data tab in the ribbon, then Refresh All.

Refresh. The pivot table doesn't know when you make changes to the underlying data, so you must specifically tell Excel to refresh it. Get used to right-clicking the pivot table and clicking Refresh, for each of your pivots. Alternatively, click Data tab in the ribbon and Refresh All to refresh all of your pivot tables in the workbook[33].

Data Source. You assigned a data source when you created the pivot table. You'll need to change that data source if you've expanded the underlying data with new rows or columns. Especially if you've added more rows to the underlying data, you'll need to add these new rows to the data source before the pivot knows about them!

To change the data source, go to PivotTable Options on the ribbon, and choose Change Data Source (if you don't see the Options tab on top, click on the pivot table first). Yes, this feature is pretty well hidden. Excel gives you a popup box with a one-line input box to edit the data source.





Working with this one-line input box can be tricky – you'll commonly see it with pivot tables and charts. It deserves its own short section.

Working With Excel's One-Line Input Box

When you start editing an existing formula in this input box, be careful not to use the arrow keys! They work the same as when you're editing a cell: [ARROWS] move around the worksheet and use the selected cells in the formula. Remember, though, that you can switch the behavior of the arrows back to the editing box: press [F2]!

[33] Refresh All will also refresh all external data connections (if you have any!). If you do have external data, it's best to use Refresh All twice: once to refresh the underlying data, and again to make sure the Pivot Tables that reference the data are refreshed. If you don't use external data connections, then don't worry about this; Refresh All once will be fine.

Even with the handy [F2] key, it's easy to make a mistake and turn your data source into a confusing jumble of references. Often the best thing to do is to clear out the existing reference, so you can start with a blank box. Then select the range you want to enter, either with the mouse or by using the arrow keys and [CTRL] + [SHIFT] to navigate the sheet (switch the arrows back to the sheet with [F2]).

When setting pivot table data sources, you have a few options:

- Set the exact range for the data source, columns and rows: `Data!$A$1:$D$13`
- Reference the entire columns: `Data!$A:$D`
- Turn your data into a data table, and reference that table for the pivot source: `Table1`
- Create a dynamic named reference for the pivot source

The **exact range** is the most straightforward. The only problem: if you add more rows of data to extend the data source, your pivot still only references the original rows of the data. That leads to wrong results! You have to be very careful to always update the correct data source if you change the underlying data.

> One of the most famous Excel mistakes in recent history concerns issues with data source references. Two economists from Harvard University, Reinhart and Rogoff, published a paper concerning economic growth and a nation's debt levels. The paper was cited by the media and politicians in defense of austerity policies; the trouble is, Reinhart and Rogoff made a simple Excel error that caused them to publish the wrong conclusions! The error concerned omitting five rows from the calculation of an average. [34]
>
> While this particular error concerned a simple formula rather than a pivot table data source, it's easy to see how simple reference errors can cause embarrassing problems. Be extra careful with pivot sources, because they are more difficult to troubleshoot than formulas!

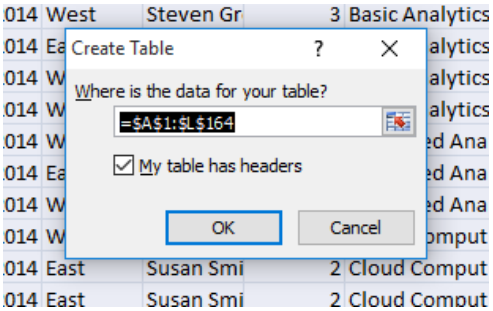
Entire column references are great for avoiding omitted data rows. All you have to do is make sure Row 1 in the data tab contains the actual data headings – you can't have empty rows at the top. You can be sure that all rows are picked up in the Pivot, even if you add more rows later (don't forget to Refresh if you do!). Your pivot will have some "(blank)" values (representing all the blank rows below the rows with data), but those can be filtered out.

	A	B
1		
2	**Row Labels**	**Count of Sale Amount**
3	James Edwards	54
4	Mark Edwards	13
5	Steven Gross	21
6	Susan Smith	46
7	William Able	29
8	(blank)	
9	**Grand Total**	**163**

You can also turn your data into a **data table**. Select the entire data set including the heading row: start in the top-left cell and [CTRL] + [SHIFT] + [RIGHT], then [CTRL] + [SHIFT] + [DOWN]. Then choose Insert,

[34] "FAQ: Reinhart, Rogoff, and the Excel Error That Changed History". Bloomberg. http://www.bloomberg.com/bw/articles/2013-04-18/faq-reinhart-rogoff-and-the-excel-error-that-changed-history

and Data Table or hit [CTRL] + [T] then [ENTER]. This turns your data into a data table, which you'll recognize by the blue coloring and the automatic filters.



	A	B	C	D	E	
1	Transaction ID	Transaction Date	Branch	Rep Name	Product ID	Product
2	1	1/3/2014	West	James Edwarc	4	Advanced
3	2	1/7/2014	West	Steven Gross	3	Basic Ana
4	3	1/11/2014	East	William Able	3	Basic Ana
5	4	1/14/2014	West	James Edwarc	3	Basic Ana
6	5	1/15/2014	West	James Edwarc	3	Basic Ana
7	6	1/16/2014	West	Mark Edwards	4	Advanced
8	7	1/21/2014	East	William Able	4	Advanced
9	8	1/25/2014	West	James Edwarc	4	Advanced
10	9	1/30/2014	West	Mark Edwards	2	Cloud Cor
11	10	2/3/2014	East	Susan Smith	2	Cloud Cor

Now that you have the data table, set it as your pivot source: select the entire table while editing the pivot data source. Excel recognizes the table by name.

One last solution for pivot data sources: this is my preference for big files that I do a lot of work in, especially if my data changes frequently. It takes a little setup, but the outcome is definitely worth it for ease of use! The idea is to use a **dynamic named reference as your Pivot Table data source.** Start by creating a named reference range, maybe called "Data1", that references a formula very similar to this:

```
=OFFSET(Data!$A$1,0,0,COUNTA(Data!$A:$A),COUNTA(Data!$1:$1))
```

This named reference uses the OFFSET function in a clever way (yes, you can use functions inside of named references). We take A1 from the Data tab, which is supposed to be the first data heading. Then we use OFFSET to expand the selection to the right and down. First, expand it down to include however many cells contain values in column A, which is determined by the function `COUNTA(Data!$A:$A)`. Then, expand right to include however many cells contain values in row 1, using another COUNTA function. The result is a range that includes every column with a heading and every row.

Note: this formula does not pick up all the rows if column A has blank cells! Pick a column that definitely does not contain blanks.

Finally, you want to assign this named reference "Data1" as the source for your pivot table. You now have a dynamic range as your pivot table source, which always references the correct number of rows and columns.

Grouping Labels

Grouping is very useful if you want to summarize your data over a certain variable. To illustrate grouping, I took some data from the World Bank on countries' per capita GDP and access to electricity.[35] You can download the Country Data in Excel from www.AdvancedExcelBook.com or replicate it on your own. Create a pivot table and use GDP Per Capita as the column labels and the averages of Access to Electricity as the values.

There are no two countries with exactly the same population, so this table is not useful right now (see the left screenshot below; every row has one unique GDP value). We should group the data labels on the left into GDP ranges. For example: under 2000, 2000-7000, 7000-12000, and so on, grouping by every 5000. Pivot tables can do this easily:

[35] "GDP per capita (current US$)" The World Bank. http://data.worldbank.org/indicator/NY.GDP.PCAP.CD/countries

1. Right-click one of the numbers in the left side column (the column labels)
2. Group
3. Specify minimum, maximum, and interval (in this case 2000, 30000, 5000) and click OK.

	Row Labels	Average of Access to Electricity
4	240.6	6.5
5	350.3	16.4
6	355.6	26.6
7	369.6	9.8
8	378.2	14.4
9	378.8	9.8
10	447.8	26.2
11	456.3	15.4
12	484.6	10.8
13	490.9	14.2
14	517	34.5
15	524.9	20.2

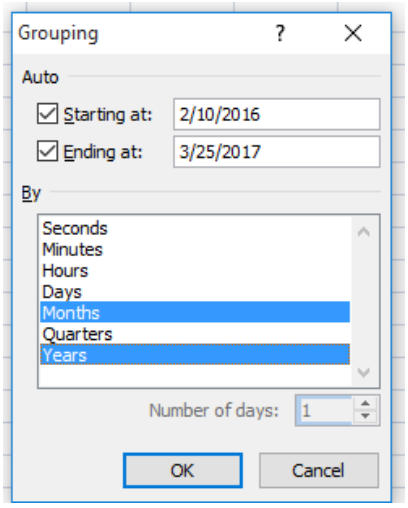
	Row Labels	Average of Access to Electricity
4	<2000 or (blank)	49.39295775
5	2000-7000	85.33137255
6	7000-12000	93.4
7	12000-17000	97.93571429
8	17000-22000	99.58333333
9	22000-27000	94.7
10	27000-32000	90.9
11	>32000	98.85142857
12	**Grand Total**	**77.99333333**

With your row labels grouped, the data starts to make sense: countries with less than $2000 GDP (or blank) have 49% access to electricity; then it starts to rise into the 90-99% range pretty quickly.

You might have noticed a limitation of grouping: the intervals have to be equal size! The only way to create custom, uneven intervals is by adding a new column to the original data, and using IF or VLOOKUP to group the GDP values. For example, you could segment out: blank, 0-2000, 2000-5000, and 5000+. This new column would replace the raw GDP values in the column labels.

To build these custom intervals, you'll probably want to use VLOOKUP with a lookup table. Go back to Chapter 4 if you need a refresher. Creating custom intervals is more time-consuming than right-clicking and grouping. I prefer the simple grouping method for quick analysis, and VLOOKUP for something more in-depth.

If your column labels are dates, there are special types of grouping available. You can group by days, months, years, and so on. Right-click and Group as usual, then select one or more types of grouping intervals. I like months and years together.



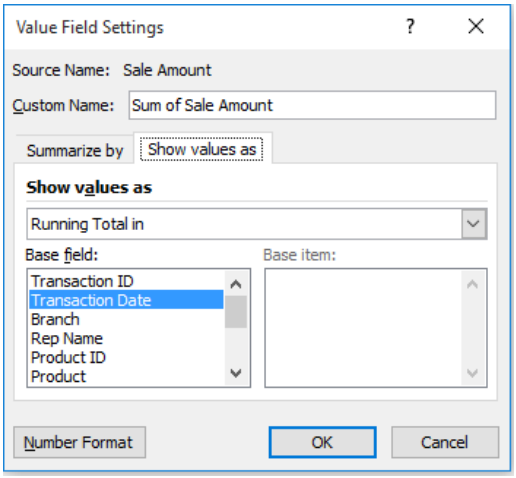
Sometimes grouping gives you the error “cannot group that selection”. We looked at ways to resolve this error earlier in this chapter.

Ways to Display Data

Pivots have a lot of useful ways to display data, besides just SUM, AVERAGE, and COUNT. To see some more options, let’s go back to the sales rep data. You should have Sum of Sale Amount in the values section. Pull up Value Field Settings (either by right-clicking one of the values inside the pivot, or by left-clicking that field in the pivot field list on the right).

Under Value Field Settings, click on the Show Values As tab. There are a lot of good choices here; my favorites are **% of Row**, **% of Column**, and **Running Total In**. Choose % of Row and click OK; the pivot table now shows the percentage of sales by category. Put the Rep Name along the row labels to see how the total sales break down by rep.

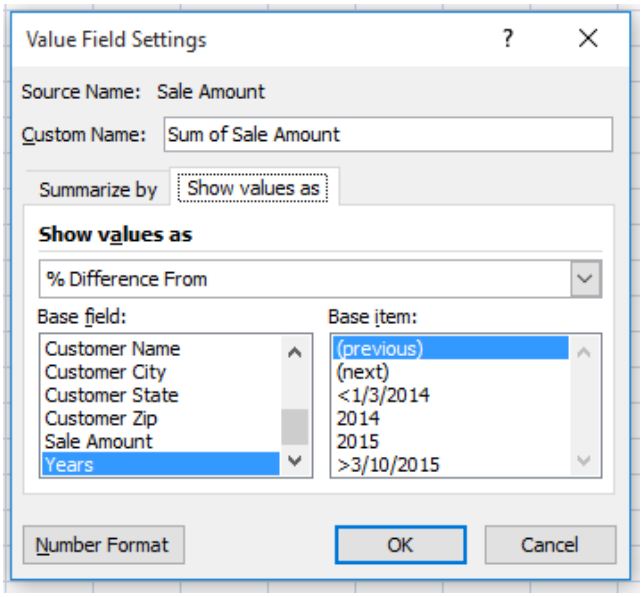
To illustrate **Running Total**, try this: clear out the pivot table. Put Rep Name as the column labels and Transaction Date as the row labels. Keep Sum of Sale Amount as the values. Open up the Value Field Settings and choose Running Total In from the dropdown. Your Base Field is the Transaction Date. Here’s a screenshot of those field settings, and the resulting pivot:

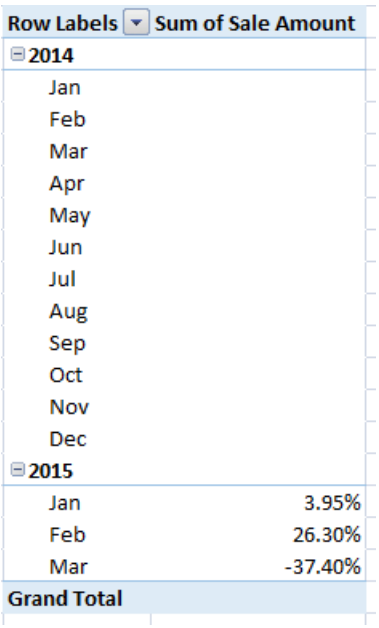


Sum of Sale Amount	Column Labels		
Row Labels	James Edwards	Mark Edwards	Steven Gross
1/3/2014	17400	0	0
1/7/2014	17400	0	5500
1/11/2014	17400	0	5500
1/14/2014	23600	0	5500
1/15/2014	28700	0	5500
1/16/2014	28700	18100	5500
1/21/2014	28700	18100	5500
1/25/2014	48600	18100	5500
1/30/2014	48600	38400	5500
2/3/2014	48600	38400	5500

How to read this output? Look along the columns; we are showing a cumulative amount (“running total”) of sales for each rep over time.

If your data is grouped by years and months (see the Grouping Labels section above), you can calculate **year over year percentage changes** directly in the pivot table. First, you’ll need to put Dates in the row labels, and group them by months and years. Then go back to Value Field Settings, the Show Values As dropdown, and choose **% Difference From**. The Base Field is Years, which is a field that was created by the date grouping. Base Item is just the previous year. You’ll know this worked if you see no values for the first year, and you see percentages in the subsequent year(s). The following screenshots should help:

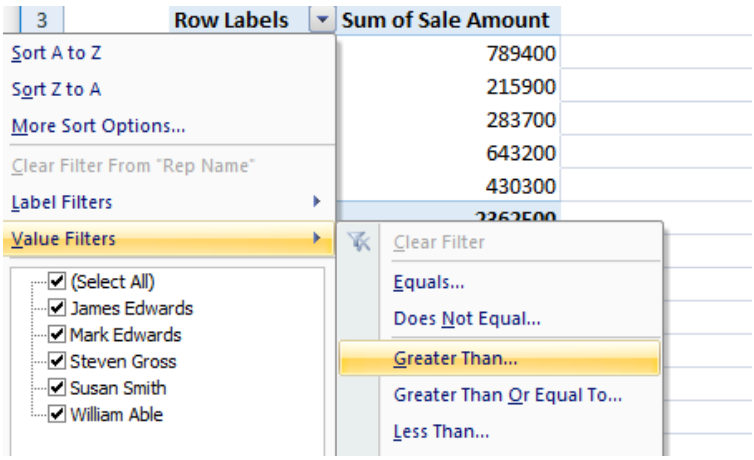


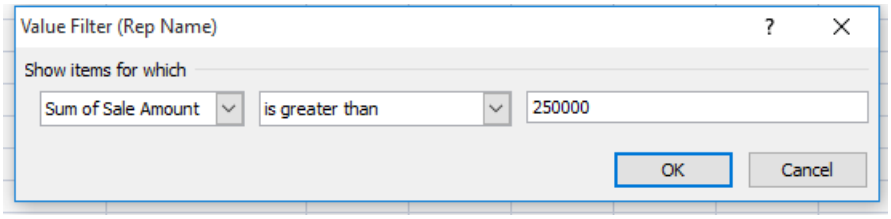


Filter by Values

Filtering pivots by values can be very helpful, and it's not obvious how to do it. The idea is this: I have a list of sales reps and their dollar sales. But I don't want to show any reps with less than $250,000 in sales. How do we filter out categories based on the sum of a field's value?

The answer: set up your pivot as normal, with Rep Names as the row labels. Click the filter dropdown next to the label, and choose Value Filters, Greater Than. Now you have the choice to filter based on the sum of sales, or on any other field in the pivot. Choose Sum of Sale Amount greater than 250000.





Text Values in Pivot Tables

Text in pivot tables doesn't behave very well. If you put any text fields in the Values section, all you'll be able to do is Count it. You obviously can't sum or average text, but you also can't display the text itself.

There is one way to trick the pivot table into outputting text values: put the text in the Rows as a subcategory, and switch to what's called Tabular Layout without subtotals. The end result looks like this:

Row Labels	Customer Name	Sum of Sale Amount
1	ABC Incorporated	430300
2	Credit Partners	283700
3	Western United Bank	215900
4	Consumer Technologies Inc	291500
5	Advanced Solutions	321700
6	New York Financial	351700
7	Genuine Systems	467700
Grand Total		**2362500**

Follow these steps to build a pivot table with multiple category items side by side:

1. Put Customer ID and Customer Name both in the Row Labels section
2. Right-click one of the Customer ID items and choose Field Settings (this has to be for the upper level category, not Customer Name)
3. Tab over to Layout & Print and choose Show items in Tabular Form instead of Outline Form
4. If you have subtotals, remove them by right-clicking one of the Customer IDs again, and unchecking Subtotal.

There you have it! Tabular form puts categories side by side. If your categories are one-for-one (as in this case, one Customer ID goes with one Customer Name), then they line up row for row. Use this trick to **display Text values in the pivot table** by putting them in the Row Labels section.

You can also use this trick to display a customer's name and address, or a rep's name and ID, on the same row of the pivot. Remember that text fields should go in the row or column labels; otherwise, their values will not show up.

Calculated Fields for Weighted Averages

Calculated Fields for pivot tables are very versatile; my favorite application is for outputting weighted averages instead of simple averages.

The distinction between simple and weighted average is very important. For example, consider some data on home listings, where price per square foot is an important metric. If you have three values like this, what is the average price per square foot?

Unit	Price	SquareFeet	Price/SqFt
1	$100,000	250	$400
2	$600,000	2500	$240
3	$750,000	3200	$234

Clearly unit #1 is an outlier; it's small, but the price per square feet is much higher. If you put these values in a pivot table, and get the Average of your Price/SqFt field, the result is $291 (the average of 400, 240, and 234). That's a very misleading result!

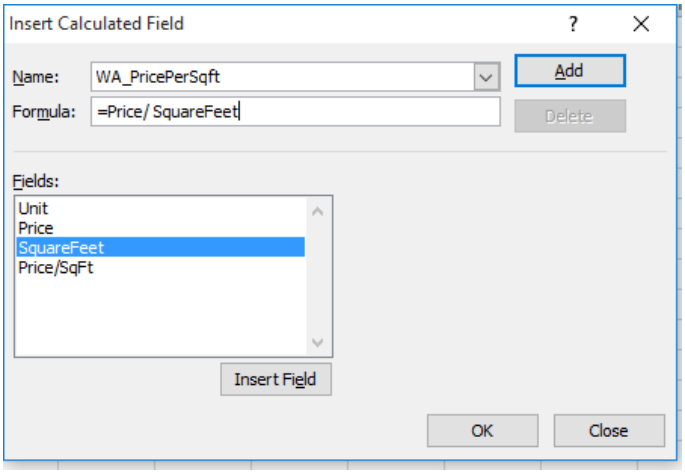
What you really want is a weighted average, weighted by the square feet. If this was an Excel formula, you would use: the sum of all the prices, divided by the sum of all the square feet. To do the same thing in a pivot table, you need a calculated field.

Select the Options ribbon (which is available when you click on a pivot table), then look for a button called Formulas (Excel 2007), or Fields, Items, & Sets (Excel 2010 and newer). Then click Calculated Field.

In the Insert Calculated Field box, you'll need to name the new field and give it a formula. Set it up like the screenshot on the next page; the name is WA_PricePerSqft, and the formula is =Price/SquareFeet.

A nice timesaver while entering these formulas: double-click a field in the list, and it gets inserted into the formula. When you click OK, there's a new field available to use in your pivot, and it has the correct value of $244. This is the weighted average price per square foot.

Note: the correct calculation is =SUM(Price)/SUM(SquareFeet), and you could enter the formula this way. It's technically more correct. Excel will assume you meant to SUM if you leave that out.



When you're using division in calculated fields, some of the cells in the pivot could have division by zero. Sometimes that's a legitimate output, but you'd rather not display nasty "#DIV/0" cells in your pivot table – someone might think there's a mistake! Here's an easy way to replace errors with blank cells:

1. Right-click inside the pivot and choose PivotTable Options (not Value Field Settings)
2. Under Layout & Format, check the box next to For Error Values Show
3. Put whatever you want in the adjacent text box as the error values result; I like to just keep it blank

When you click OK, all error values in the pivot are replaced by this value.

Show Underlying Data

Sometimes you'll see a value in your pivot table that doesn't make sense; or you're simply wondering what the original underlying data looks like for that output value. Here's an easy way to **see all the data behind a specific cell** in the pivot table: double-click the value itself! (Not the row or column label – click one of the values).

Advanced Tip

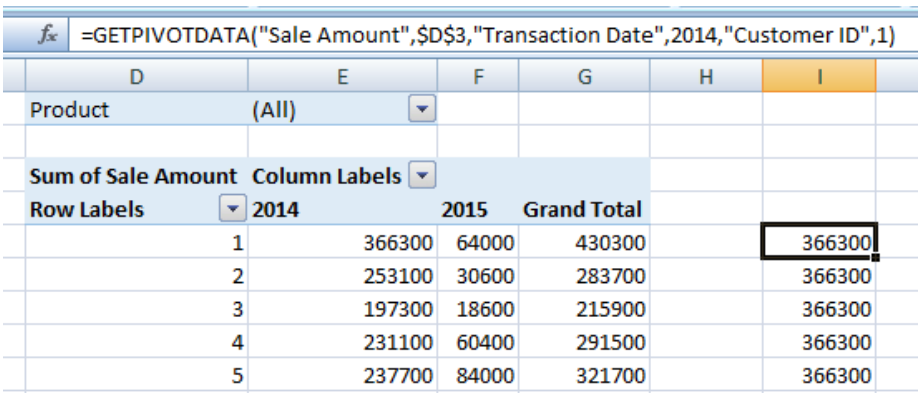
To see the underlying rows of data behind a pivot table cell's value, double-click that cell in the pivot.

Excel automatically creates a new sheet with all the rows of data behind this cell.

When you're done, feel free to delete this new sheet (Sheet3, Sheet4, or whatever name it auto-generates under). Nothing links to this sheet and you won't lose any of the original data.

Referencing Pivot Tables with and without GETPIVOTDATA

As soon as you start using pivot tables, you'll probably notice it's difficult to reference pivot table cells in formulas. Excel insists on using a function called GETPIVOTDATA instead of the normal "B10" format of cell reference. The problem is, you can't just "drag down" a GETPIVOTDATA formula, as seen below:



	D	E	F	G	H	I
	Product	(All)				
	Sum of Sale Amount	Column Labels				
	Row Labels	2014	2015	Grand Total		
	1	366300	64000	430300		366300
	2	253100	30600	283700		366300
	3	197300	18600	215900		366300
	4	231100	60400	291500		366300
	5	237700	84000	321700		366300

The formula looks something like:

```
=GETPIVOTDATA("Sale Amount",$D$3,"Transaction Date",2014,"Customer ID",1)
```

Dragging this down, the references stay anchored to the original fields. The result is the same in every row; how annoying! You need a way to change the second row to the next Customer ID like this:

```
=GETPIVOTDATA("Sale Amount",$D$3,"Transaction Date",2014,"Customer ID",2)
```

And change references to the next year (2015) like this:

```
=GETPIVOTDATA("Sale Amount",$D$3,"Transaction Date",2015,"Customer ID",1)
```

You certainly don't want to re-type these formulas for each cell. That's very slow and not at all productive. You have some better options:

- Manually type in the cell reference pointing to the pivot, like "E5". Don't click or use arrows to select the cell inside the pivot. The reference E5 can now be dragged down and behaves normally! It's a pretty easy work-around if you remember to do it.
- Modify the GETPIVOTDATA formula. Although the formula looks messy, it does have pretty consistent logic, so it makes sense to spend some time getting used to it – you might actually prefer it! I'll show you an example in a moment.
- If you really don't like GETPIVOTDATA, get rid of it forever! Go to Excel Options, then Formulas. You'll find a check box next to "Use GetPivotData Functions for PivotTable References". Un-check this box, and the GETPIVOTDATA feature will no longer come up.

Here's how to make GETPIVOTDATA work better: modify the formula to use cell references. Compare these two versions of the same function:

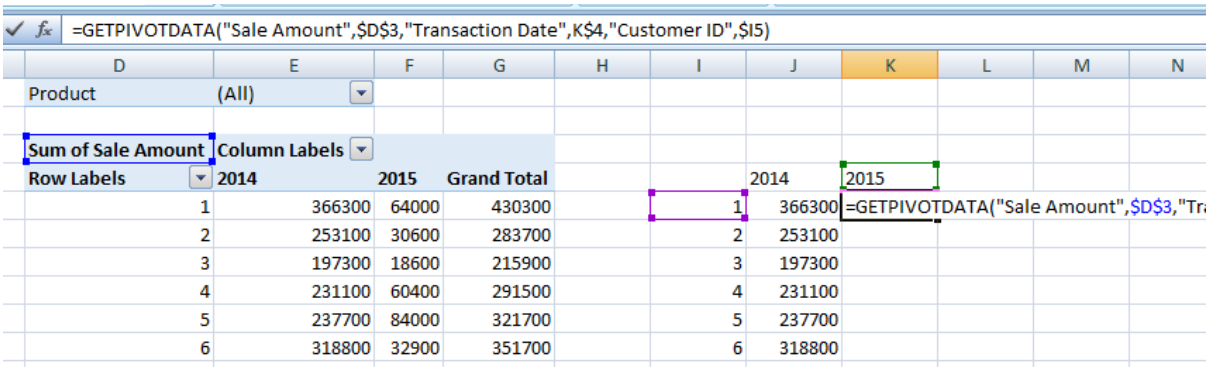
```
=GETPIVOTDATA("Sale Amount",$D$3,"Transaction Date",2014,"Customer ID",1)
=GETPIVOTDATA("Sale Amount",$D$3,"Transaction Date",J$4,"Customer ID",$I5)
```

The first function reads like this: take the Sale Amount value from the pivot table that occupies cell $D$3. Then filter for variable Transaction Date, value 2014. Then filter for Customer ID, value 1. The sum of sales for Customer 1 in 2014 is the result.

Advanced Tip

You can modify the automatic GETPIVOTDATA formulas to use cell references.

The second version is the same thing, but Transaction Date references cell J4 (with the row anchored) and Customer ID references cell I5 (column anchored). Put 2014 and 2015 in row 4, put the customer IDs in column I, and the result is a **GETPIVOTDATA formula that you can drag down and across**! See the screenshot below for the proper setup.



Charts

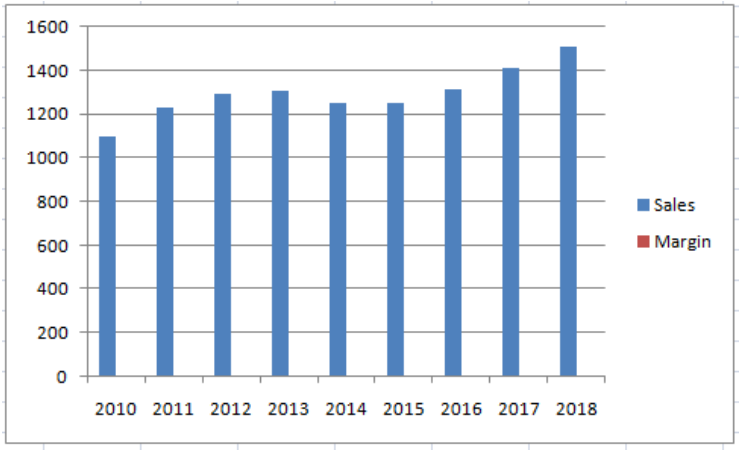
You probably already know how to make some basic charts in Excel. In this section, I'll show you how to make the most of these charts, as well as some hidden features.

Chart Formatting

We'll work through a long example to practice the following:

- Mix a line graph and a column graph on the same chart
- Use primary and secondary Y-axis
- Change bar widths and gaps in a column chart
- Apply number formatting to axis labels
- Modify grid lines and intervals

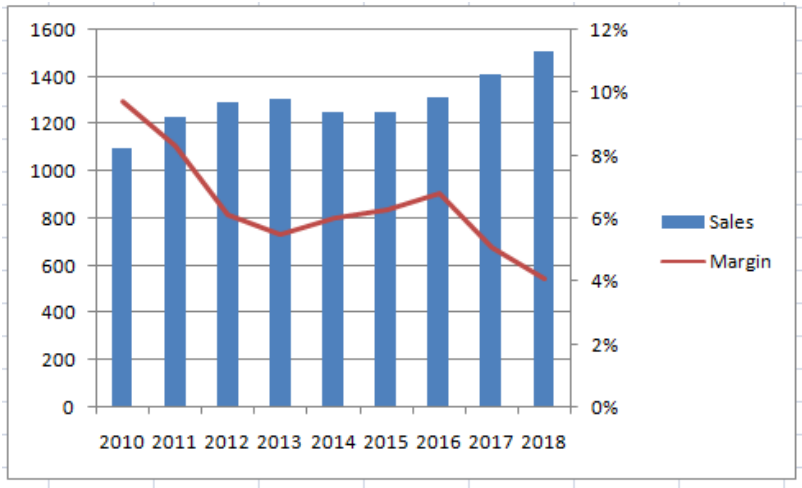
The data I'm using is available to download from www.AdvancedExcelBook.com or just make up a similar data set with years, sales numbers (with values around 1000-1200), and profit margin (percentages). The screenshot shows the example data as a column chart. We'll improve this chart step by step.



Put Margin on the secondary axis. The Margin values are very small (they're just percentages), you can't even see them! Since Margin is on a completely different scale from Sales, it needs to be on a secondary Y-axis. To do this, right-click the Margin data series within the plot area. It's pretty difficult in the example above because they are really small values! An easy trick is this: go inside the data, and multiply one of the Margin values by 1000. Now you'll see a big red bar for that data point, which is easier to right-click. Don't forget to remove the 1000 multiplier afterwards.

Right-click the Margin data series and choose Format Data Series. Find the option for Plot Series On, and switch to the Secondary Axis.

Convert Margin to a line graph instead of columns. I like mixing lines and columns on the same graph if the data represents different things, especially if one of those series is on a secondary axis. It helps visually separate the two types of data. Right-click the Margin series again and choose Change Series Chart Type. Select the first type of line chart (beware of the second line chart, the Stacked Line Graph; more on that later). Now the graph is starting to come together nicely.



The above modifications are the bare minimum to make this chart usable. Notice that we still have a lot of formatting work to do (the year labels are smashed together, the axes and the chart itself are not labeled, there are no units anywhere, the legend takes up too much space on the right, the bars are very skinny, the colors are not particularly attractive, and so on).

Move the legend to the bottom. That's easy: in the chart section of the ribbon (with the chart selected), click the Layout tab and the Legend button (Excel 2007 and newer). Show Legend at Bottom leaves more space for the chart itself.

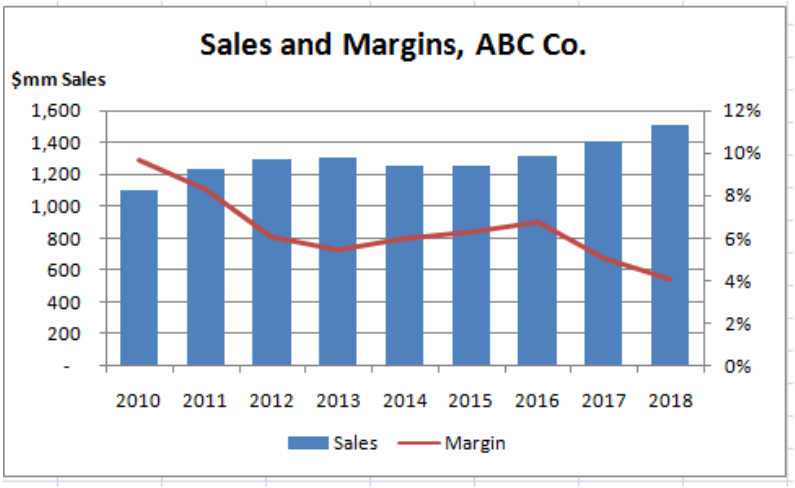
Number formatting. I like my numbers with commas after the thousands place. To change number formatting on the axis values, you have two options. Use number formatting on the source data itself, or right-click the axis and choose Format Axis. Then go to the Number tab.

Axis labels. Excel calls them “axis titles”. They’re found under the Layout tab of the ribbon, next to the Legend placement options. The axis on the left is the Primary Vertical Axis, and you have some layout options to choose from. Vertical Title looks pretty bad in my opinion, Rotated is a bit hard to read, and Horizontal takes up a lot of space. Of three imperfect choices, I recommend either the Rotated version or the Horizontal one. With Horizontal, you’ll want to rearrange things a bit so as not to leave so much white space. Feel free to click the axis label and drag it around.

Chart title. Add titles to your charts under Layout in the ribbon. Sometimes I like to change the font size of my chart title, making it one notch smaller than the default.

Change column width and gap. Right-click on the column series and choose Format Data Series. You have two sliders to adjust: Series Overlap and Gap Width. Try a gap width of 75%; it should look quite nice.

Below is a little update to show where we are so far. One more thing you should consider fixing: notice that the right axis hash marks (at 10%, 8%, 4%, 2%) don’t line up with the main horizontal gridlines. This could be confusing to people looking at the graph.



Line up horizontal gridlines for left and right axes. Right-click the right axis and choose axis options. We can choose our own maximum cutoff and/or major units to force the hash marks to line up correctly. In this case, choose 0.16 as the maximum.

Different colors for projected values. 2016-2018 are projected revenues, while earlier years are actual revenues. You might want to make a visual distinction by coloring 2016-2018 differently. Right-click the 2016 bar (not the entire data series this time) so you have the option to Format Data Point. In the popup box you’ll go to Fill, and choose a lighter shade of blue. This changes the color of just that one bar. The good news is that there’s a quick way to do this with the next bar: click on the bar for 2017 and press [CTRL] + [Y] for redo. The same setting now applies to the 2017 bar.

Finally, you might choose to change the label 2016 to “2016E” for “expected”, and do the same with 2017-2018. Go back to your source data on the sheet to make this change.

Here is the final chart after all of the above changes. Isn’t it pretty?

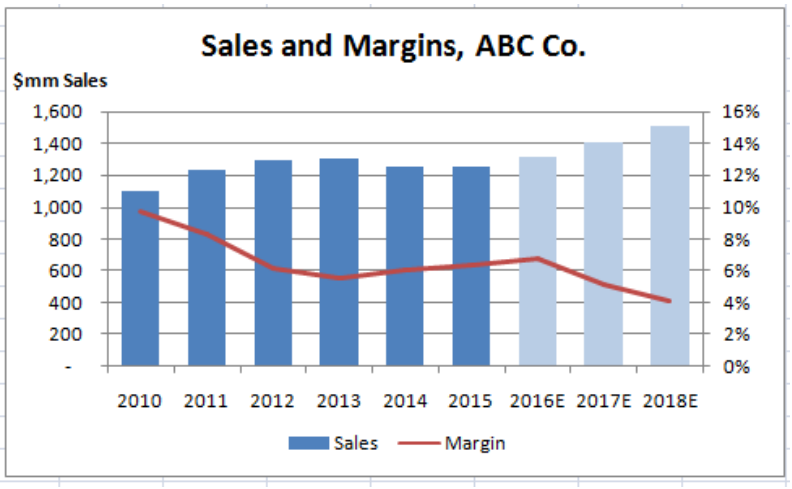


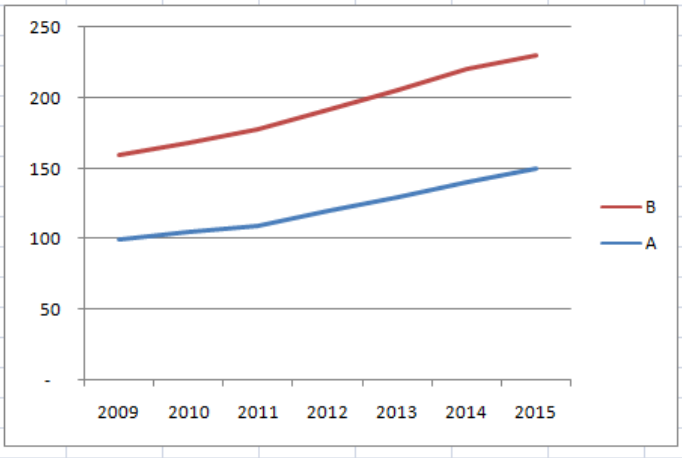
Chart Types

Excel has a lot of chart types. A few are truly useful; many are very specific and you'll only use them once in several years. The 3D charts look weird and are difficult to read accurately, so I would discourage using them. The **four core chart types** you'll use again and again are:

- 2D Column, not stacked
- 2D Line, not stacked
- 2D Pie, regular or "exploded"
- Scatter, without lines

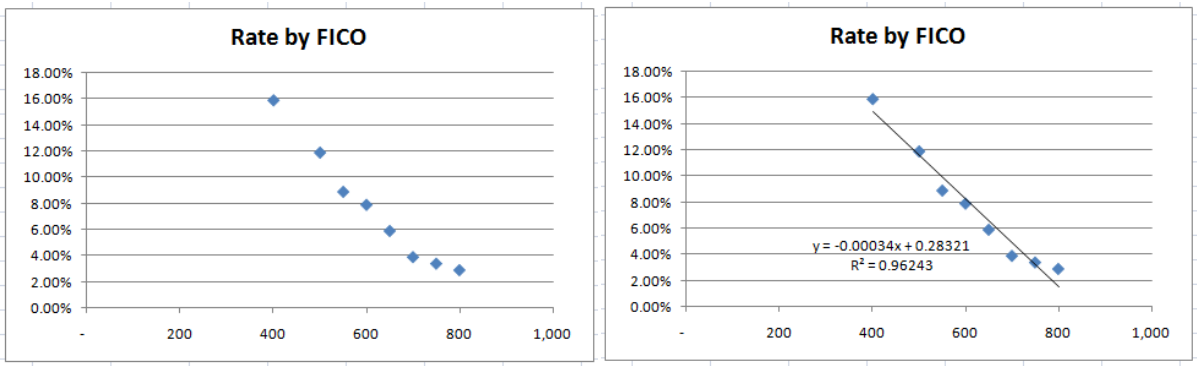
Beware of stacked graphs such as the Stacked Line Graph. It can trick you into making a huge mistake. For example, let's say you're graphing the sales growth of two companies. Company A has sales growing from 100 to 150. Company B has sales growing from 60 to 80. Now make a Stacked Line chart, which is the second choice under line charts. You'll get something like this, with company B showing more sales than Company A:

	A	B
2009	100	60
2010	105	64
2011	110	68
2012	120	72
2013	130	76
2014	140	80
2015	150	80



Do you see how the values for the "B" line are really "A+B", meaning "B" stacked on top of "A"? **This graph is misleading and you should never use it**. If you must make a stacked graph, Stacked Area is a much better option. Even Stacked Column will likely trick some people. Don't trick people with graphs.

Scatter plots are great for understanding relationships between data sets. For this example we'll look at data on FICO scores and interest rates. For an X-Y scatter plot we'll use FICO as the X-axis and Rate as the Y-axis; select just those two columns of data and insert the scatter plot graph. It looks something like the graph on the left, with individual dots representing each data point:



You can use **trendlines** to measure a linear, exponential, polynomial, etc. relationship between these two data sets. Linear should be the most common. To add a trendline, right-click the data set and choose Add Trendline. I recommend checking Display Equation on Chart and checking the R-Squared Value (which measures how closely the trendline fits the data). A linear trendline equation is the same as a single-variable regression (just more visual and easier to do).

Do you ever wish the trendline equation had more digits? My equation shows as y = -0.000x + 0.283, which is nice, but I need more digits before the x! The solution: right-click the label itself and click Format Trendline Label. The first tab is number formatting; choose Number instead of General and specify the number of decimals you need. In this case I'll use 5. The equation now shows as y = -0.00034x + 0.28321, which is much more helpful.

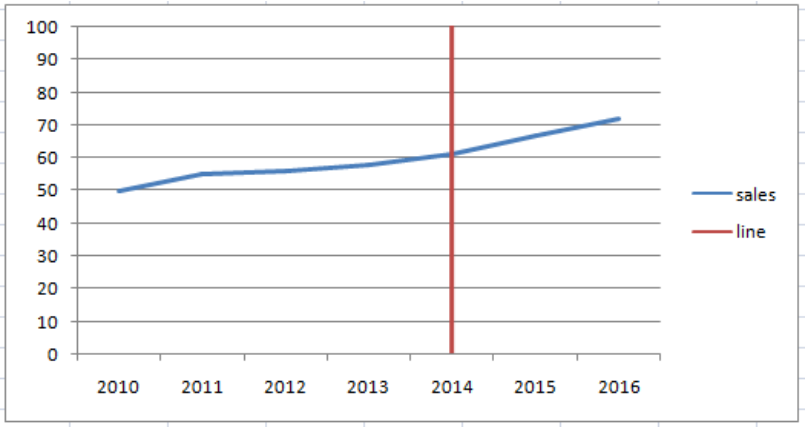
Do you ever have to **graph three-dimensional data**, meaning something with X, Y, and Z properties all at once? I like the **bubble chart** for this purpose. It's a normal X-Y scatter plot, but you can add a third dimension as the size of the bubbles. Large circles are large values, and small circles are small values. Try it out!

How about showing a 2D relationship changing over time? To do this I like to combine macros, the OFFSET function, and any kind of 2D chart. The idea is this: set up each time period's data side by side, and use OFFSET to pull in time period 1, 2, 3, etc. By changing the offset parameters, you can pull in different sets of data. The chart itself only references a single set of X-Y data points. And you'll have a macro that steps through different values of OFFSET, which dynamically moves your chart forward in time! You'll know how to do this once you finish Chapter 8.

Other Chart Topics

Excel's charts are good at making basic graphs quickly; but you'll probably run into limitations if you want to be creative with graphs. Sometimes you have to trick Excel a little bit. Below are two examples.

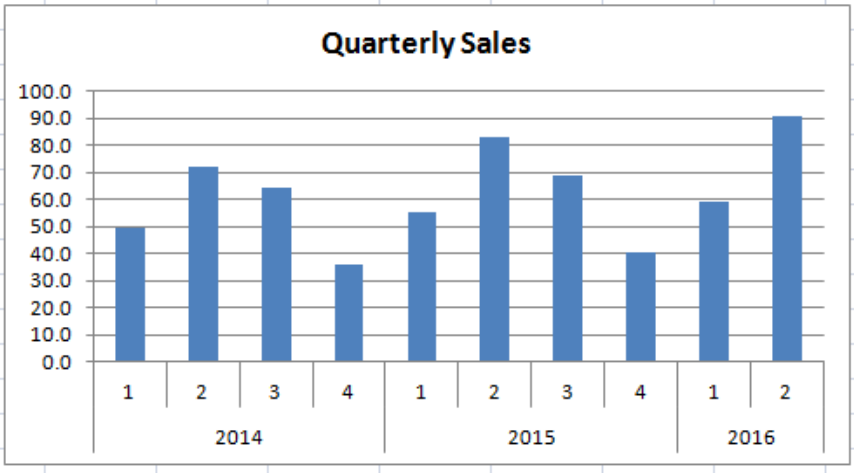
Graph a vertical line. There's no way to show a perfectly vertical line in a graph, but you can create that illusion. The idea is to make a line series go from 0 to a very large positive number (or from a very large negative number to a very large positive) at the point you need the line. "Very large" depends on the scale of your graph; if your numbers go from 0 to 100 then 1,000,000 should be plenty. A sloped line from 0 to 1 million looks like a perfectly vertical line when your axis only goes to 100. You'll just need to hard-code the maximum and minimum values for your Y-axis; In this example, I cap the axis at 100:



Lastly, you'll want to remove "line" from the legend. You could delete the entire legend, or click inside the legend on just the word "line" and press [DEL]. Now the legend only shows the real data series.

Multiple columns for X-axis labels. You can use two columns to serve as your X-axis labels, which is a very nice presentation for something like years and quarters. Take a look at this chart:

	A	B	C
1	Year	Quarter	Sales
2	2014	1	49.7
3		2	72.3
4		3	64.0
5		4	36.0
6	2015	1	55.0
7		2	83.3
8		3	68.7
9		4	40.3
10	2016	1	59.0
11		2	91.0
12			
13			
14			

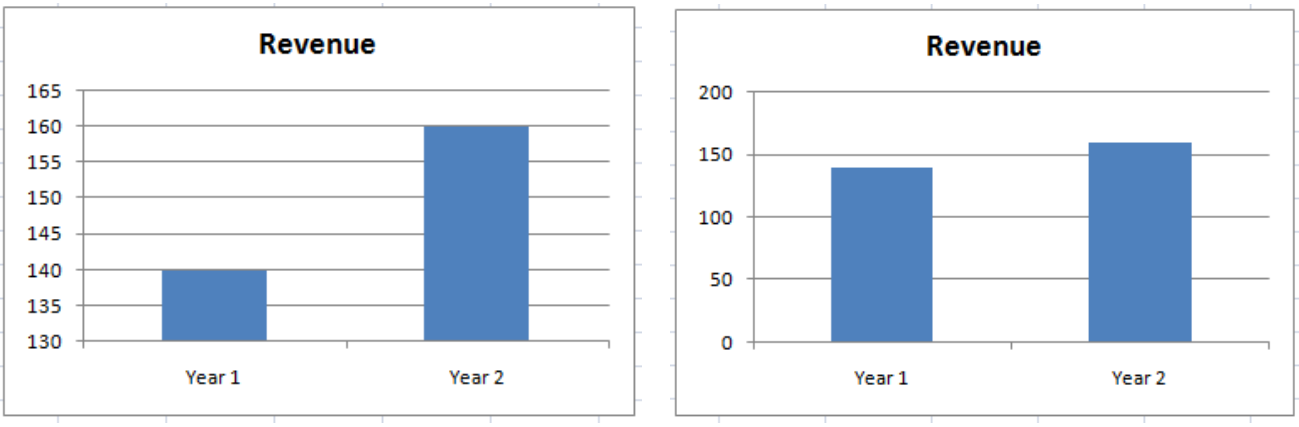


Year and Quarter are both axis labels. To achieve this, start with a regular column chart. Then right-click the chart and choose Select Data. Under Horizontal (Category) Axis Labels, choose Edit and select both columns of labels, year and quarter (A2:B11 in this example). You'll have just one data series called Sales.

Avoid starting your Y-axis at a value greater than zero; it's too easy to misread a chart that way! For an illustration, take a look at the column charts on the next page showing revenue growth.

It sure is more exciting to see the rate of growth from the first chart! But look closer and notice the two charts show the same data: revenue increasing from 140 to 160. If your boss or client is busy they will miss

something like this and draw the wrong conclusions. It's your responsibility to make charts easy and intuitive to read.



When you're making charts you'll sometimes run into this error: "**A formula in this worksheet contains one or more invalid references.**" This one can be a real brainteaser, because it comes up when seemingly all the formulas are fine! The cause of it is almost always an **invalid reference in a chart's source data**. The solution: go through your charts one by one. Change the data sources, and delete any unnecessary data series. If this approach fails, delete charts one at a time until the error stops coming up; this way you'll narrow it down to just one chart to fix. Worst case scenario: you have to recreate that chart from scratch. Sorry! Be careful with deleting rows or columns that are source data for charts, as this error can come up.

One last cool thing about charts: **you can copy-paste formatting from one chart to another**, which is a huge time saver when you have multiple charts and want to keep their formatting consistent.

Advanced Tip

You can copy and Paste Special the formatting from one chart to another.

Select the first chart and click Copy, or use [CTRL] + [C]. Paste Special over the target chart using [ALT] [E] [S] (or find Paste Special under the Paste dropdown on the Home ribbon). You get a mini version of the Paste Special popup. Your choices are: All, Formats, Formulas. Choose Formats to paste the formatting from the first chart to the target chart!

Cleaning Up Data Using External Tools

To conclude this chapter on data analysis, I want to mention two great tools outside of Excel that could make your life easier. Excel is not particularly good at cleaning up messy data such as inconsistent text and number formatting. Try out these two tools instead:

- Open Refine (www.openrefine.org) is an open source program, originally developed by Google as "Google Refine".
- Data Wrangler (vis.stanford.edu/wrangler) is a web-based tool developed at Stanford; it's no longer supported, but was still working at the time of this writing.

7: Excel Modeling

In this chapter we'll talk about building good Excel models. You've already learned the essential building blocks: keyboard shortcuts, functions, charts and analysis tools, error handling, and so on. The mechanics of modeling are not complicated. Most financial models (the kind that analysts get paid hundreds of thousands per year to build!) use very simple formulas: basic math (add, multiply, subtract, some parentheses), some aggregation functions like sum or average, and the occasional OFFSET or IF for scenario analysis.

Good modeling means building an **organized** and **easy to understand** spreadsheet, which **reflects the reality** you are modeling and is **free of calculation errors**.

I'll begin this chapter with a review of Excel modeling mistakes. Then we'll talk about best practices to avoid these mistakes and create Excel models that are functional, easy to use, accurate, and beautiful.

Modeling Mistakes and Risks

I've mentioned a few famous Excel errors throughout this book. There are plenty more where those came from: I particularly enjoyed reading a list of "horror stories"[36] maintained by an organization called the European Spreadsheet Risks Interest Group.[37]

For a more thorough look at Excel errors, consider two academic papers on the subject: "What We Know About Spreadsheet Errors" by Raymond Panko,[38] and "Errors in Operational Spreadsheets" by three professors at Dartmouth.[39] Here are their main findings:

- About 90% of spreadsheets had errors causing wrong results to be obtained (there were additional instances of "poor practices" on top of these).

---

[36] "EuSpRIG Horror Stories". Patrick O'Beirne. 2003-2004. http://www.eusprig.org/horror-stories.htm

[37] Note that most of the Excel mistakes I've referenced in this book are also listed on this web page. I have cited the original sources of those stories, although their summaries can be found on several online lists including this one.

[38] "What We Know About Spreadsheet Errors". Raymond R. Panko
http://panko.shidler.hawaii.edu/SSR/Mypapers/whatknow.htm

[39] "Errors in Operational spreadsheets". Journal of Organizational and End User Computing, 21(3), 24-36, July-September 2009. Stephen G. Powell, Kenneth R. Baker, Barry Lawson.
http://tuck-fac-cen.dartmouth.edu/images/uploads/faculty/serp/Errors.pdf

- The most common types of errors were logic errors (incorrect formulas), reference errors (referencing the wrong input cells), copy-paste errors and hard-coding numbers in formulas.
- Approximately 1% of spreadsheet cells contained some kind of error – either wrong results or poor practices.
- Most model creators were over-confident in their ability to produce error-free spreadsheets.

Don't get discouraged by these findings; the errors cited by these authors and covered by the media are generally simple mistakes caused by lack of familiarity with Excel, disorganization, or lack of documentation. With good modeling habits and effective troubleshooting, I believe you will significantly cut down on mistakes.

The prize for the costliest of all spreadsheet errors in history has to go to JP Morgan's "London Whale" – a trader who amassed massive positions in complex securities known as Credit Default Swaps. The loss was $5.8 billion dollars, a massive amount of money even for a global bank (though likely only a small fraction of this sum is related to Excel). Congressional as well as internal investigations followed, revealing the role of Excel spreadsheets in the debacle (as first reported by financial blogger James Kwak).[40]

JP Morgan was using a model that "operated through a series of Excel spreadsheets, which had to be completed manually, by a process of copying and pasting data from one spreadsheet to another." They identified a specific formula error whereby "the spreadsheet divided by the […] sum [of two values] instead of their average, as the modeler had intended."[41] A trivial mistake perhaps, but a very expensive one!

Golden Rules of Modeling

When building an Excel model, keep one guiding principle in mind: when you finish the model, and deliver it to your boss or your client, would you trust him or her to figure out what to do and use the model as intended? How would you make your spreadsheet easy to use (for someone who hasn't seen it before and doesn't have time to study every formula)? The answer lies in the golden rules of modeling:

Rule #1: Don't mix constants and formulas.

Rule #2: Keep formulas consistent.

Rule #3: Label things.

Rule #4: Simplify!

Advanced Tip

Follow these rules! Your models should be clear and easy to understand

The importance of these rules becomes easier to understand when comparing screenshots of a "bad" and a "good" model. These are different versions of the same Excel model, showing revenues and profits of a company:

[40] "The Importance of Excel". James Kwak. Baseline Scenario. Feb 9, 2013. http://baselinescenario.com/2013/02/09/the-importance-of-excel/

[41] "Report of JPMorgan Chase & Co. Management Task Force Regarding 2012 CIO Losses". January 16, 2013. http://files.shareholder.com/downloads/ONE/2261602328x0x628656/4cb574a0-0bf5-4728-9582-625e4519b5ab/Task_Force_Report.pdf

	A	B	C	D	E
1					
2		Year 1	Year 2	Year 3	
3	Revenue	10	11	12.21	
4	Costs	7	7.26	7.69	
5					
6	Profit	3.00	3.74	4.52	11.26
7		1.95	2.43	2.94	7.32
8					
9		5.0%			
10		35%			
11					
12					

	A	B	C	D	E
1					
2		**Year 1**	**Year 2**	**Year 3**	**Total**
3	Revenue	$10.0	$11.0	$12.2	
4	*Growth*		*10.0%*	*11.0%*	
5					
6	Costs	7.0	7.3	7.7	
7	*Costs % of Rev*	*70.0%*	*66.0%*	*63.0%*	
8					
9	Profit Pre-Tax	3.0	3.7	4.5	11.3
10	*Tax Rate*	*35.0%*	*35.0%*	*35.0%*	
11	**Profit Post-Tax**	**$2.0**	**$2.4**	**$2.9**	**$7.3**
12					

I'm drawing these examples from past experience. Common mistakes in the "bad model" above are:

- Poor and inconsistent formatting
- Not labeling constants in blue (the good model correctly has blue formatting for constants, although that's a bit hard to see in black and white)[42]
- Mixing constants and formulas in the same cells
- Inconsistent formulas column by column
- No labels for some rows and columns
- Assumptions in random cells below the model, with no label

We'll apply the Golden Rules of Modeling to improve the model.

Rule #1: Don't Mix Constants and Formulas

This practice is so important that it will make the difference between models that are clunky and make your Excel experience terrible, versus models that are streamlined symphonies of clarity and ease of use. Not kidding.

If you're an experienced modeler in Excel, hopefully you already do this every time:

1. Put hard-coded constants (assumptions) that drive your model into cells by themselves. Color the cells with blue font.[43] Label what each cell means.
2. Use regular black font for formulas (the inner workings of the model).
3. Don't mix #1 and #2! A cell should either contain a constant (an assumption you can change as you use the model), or a formula that is an integral part of the model. In formulas, incorporate constants by referencing their cells (C3*C4) instead of hard-coding their values (10%*5000).
4. Don't repeat assumptions in multiple places; when you use the same value in multiple places, always reference back to the one original assumption cell that drives your model forward.

If the above makes perfect sense to you, you can skip this section and move on to Rule #2. If you don't already do this, but know that you're supposed to: please try to stick with it. This habit makes you a better modeler. If you've never heard of this and/or you aren't convinced that it matters: keep reading for an illustration!

---

[42] Percentages that represent growth rates or margins look better italicized, but this is not a strict requirement.

[43] Blue font for constants is standard in financial modeling. It's a very good practice that you should start using immediately if you don't use it already. By the way: newer Excel versions have built-in formatting styles on the Home ribbon for things like "output" and "input". Input is a box with orange background and a border. Sorry, but the Excel toolbar is wrong: orange is not the standard color to indicate inputs (at least in the financial world). Stick to blue.

Let's look at the example from the screenshots above. To project out revenues, you need assumptions about annual growth rates. If you know Year 1 revenues were $10 in cell C3, and you assume next year will grow at 10 percent, you might create the "bad model" using this Year 2 revenue formula:

```
=C3*(1.1)
```

Don't do this! How does someone understand what the 1.1 means? Can it be changed to 1 or 0.9 or 150? If your growth assumptions change, how do you know where to change them? Be kind to your colleagues and fellow Excel users: separate out these constants and never mix them into formulas.

The "good model" instead adds a new row labeled "Revenue growth" with 10% in Year 2. The font is blue for constants. It's a value that can be changed to model different scenarios. The Year 2 revenue formula becomes:

```
=C3*(1+D4)
```

Much better. You'll never change the formulas in row 3, which calculate revenue based on growth assumptions. To change your assumptions, you change the blue numbers in row 4. Just tell your boss or client to modify all the blue cells, and leave the black cell formulas alone.

Notice that the "good model" above has sets of assumptions in each column. You might also have certain model-wide assumptions that don't change year to year. For example, the tax rate is the same each year, and we're using a formula in D10 and E10 to reference C10; you know that C10 is the only constant because it's the only blue cell. You might want a special "assumptions" section of your model, perhaps on another sheet or at the top of this one. There you can add things like tax rate, interest rate, or stock price; these are constants that don't change year over year.

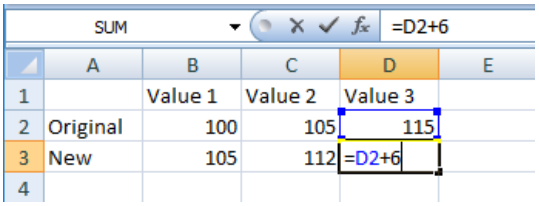
There are great benefits to spelling out your assumptions in one place: it's easy to see what can be changed in the model; you won't lose track of any particular assumptions; and you can easily use OFFSET to model different scenarios (more on OFFSET later in this chapter).

There is a common-sense exception to using constants (numbers) in formulas: only use numbers that will never be changed. To convert months to years, you don't need to put "12" in an assumptions cell and label it "months in a year". That's just silly.

However, what if your company always pays out 10% of its income in dividends? Would you hard-code that 10% for all future dividend calculations? I wouldn't! If anyone else is looking at your model, they should understand that the 10% is an assumption driving dividend payouts, rather than a part of the model. While it may be a longstanding policy, the 10% dividend payout rule shouldn't be fixed and embedded in formulas. It would be a pain to track down all the places you hard-coded "10%" if you ever had to change them. Better to be able to change just one cell in the assumptions section.

Another example: start with a row of constants that drive a model (they could be growth assumptions, prices, weights, whatever). Now, suppose you want to add a new feature to this model by showing some kind of change to these constants; maybe you want to increase everything by 2 or 3, depending on the column.

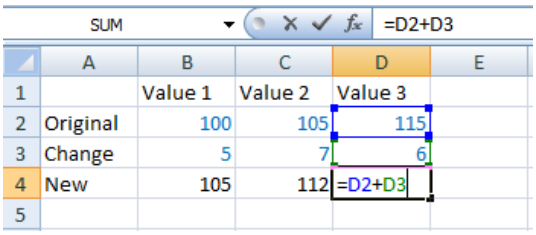
Here's what I mean:



You should avoid at all costs the following, which is what the above screenshot is doing:

```
B3 = B2 + 5
C3 = C2 + 7
D3 = D2 + 6
```

… and so on. This approach mixes constants and formulas! It will be very hard to maintain these formulas if they ever change, and very hard to update or even see the original values (5, 7, 6). Instead, create a new row for the original constants, and another row for the change factors. The final results should go in the third row. There is no excuse for not adding additional rows for constants. If you absolutely don't want your constants to be visible, you can hide the extra row or put it somewhere lower down the sheet.



Rule #2: Keep Formulas Consistent

This rule is related to the previous one, and should be common sense when I show you an example. In the "bad model", the costs formula in Year 1 is:

```
=B3*(65%+B9)
```

And the costs formula for Year 2 is:

```
=C3*66%
```

Yikes. Obviously we're violating Rule #1, but there's another problem. You want to create consistent formulas that you can drag across (or down) as you fill in the rest of your model. All Excel users will expect your models to work this way. Any exceptions to this rule, such as a different formula in Year 2, should be avoided at all costs!

In the "good model," we instead reference row 7, which contains the cost percentages. The new formula is:

```
=B3*B7
```

This way, the additional 5% cost for year 2 is just built into the 70% cost assumption. If you want to keep the 5% as a separate cell, add it in a new row 8 (special costs), and change the total costs formula to:

```
=B3*(B7+B8)
```

I can't tell you what specific formula to use; the point is to use the same formula in all columns. Any differences from year to year should always be driven by changes in constants (which are clearly visible) rather than changes in formulas (which people will, and should, assume are always the same).

Rule #3: Label Things

This one should be obvious: don't put numbers in random places. Even if you're just putting something together "real quick". When you put a number in a cell, take an extra moment and type next to it (or above it) what that number means. At minimum, labeling will reduce the cognitive load on your brain when you're trying to interpret what you're seeing – and someday, you might avoid a major mistake caused by misinterpreting a result.

In the "good model", we labeled "Growth" and "Costs % of Rev" in the appropriate rows, and removed the stray cells from rows 9 and 10.

Rule #4: Simplify!

Make your formulas simple and easy to understand, so you can avoid mistakes and have an easier time troubleshooting. Consider this big ugly formula:

```
=IF(B5=B6,0,IF(B4=B6,0,IF(C4=4,C1+D4,IF(C4=5,C1+E4,C1+F4)+IF(ISNA(VLOOKUP(I9,K:M,3,
FALSE)),"Error", VLOOKUP(I9,K:M,3,FALSE)))))
```

It's very hard to understand something like this! Look for ways to simplify formulas and break them apart into multiple cells. Here, the first few IF conditions could be handled with a series of interim calculations in three or four new cells.[44]

Advanced Tip

If a cell's formula takes up more than one line in the formula bar, seriously consider breaking it into two or three parts.

With IF functions, shorten your formulas by looking for repeated code:

Complicated: `=IF(A2=0, B2+C2+D2+F2-G2, B2+C2+D2+F2)`

Simple: `=B2+C2+D2+F2+IF(A2=0,-G2,0)`

Use the MAX or MIN function to replace certain types of IF statements:

Complicated: `=IF((A2+B2)*C5<0,0,(A2+B2)*C5)`

Simple: `=MAX(0,(A2+B2)*C5)`

Use VLOOKUP to replace nested IFs. We covered this in Chapter 4, in the VLOOKUP section.

Use interim calculations:

Complicated: `=((A2+B2)*C2)/D2 + IF((A2+B2)*C2>100,10,5)`

Simple: first set a new cell A4 `=(A2+B2)*C2`

Then the simple formula is: `=A4/D2+IF(A4>100,10,5)`

Use the IFERROR function:

Complicated: `=IF(ISNA(VLOOKUP(I9,K:M,3, FALSE)),"Error", VLOOKUP(I9,K:M,3,FALSE))`

Simple: `=IFERROR(VLOOKUP(I9,K:M,3, FALSE),"Error")`

Making Models Look Better

By following these Golden Rules, your models will be easier to understand. Now I'll show you how to make them look even better. A good Excel model is a good-looking Excel model.

Hide Gridlines. This is a simple setting: click on View and uncheck the box for Gridlines. Your sheet looks cleaner without these lines:

[44] Don't worry about using too many cells in your models. You have over 16 *billion* cells to work with in Excel! Use as many extra cells and interim calculations as you need.

	A	B	C	D
1				
2		Year 1	Year 2	Year 3
3	Revenue	$10.0	$11.0	$12.2
4	*Growth*		*10.0%*	*11.0%*
5				
6	Costs	7.0	7.3	7.7
7	*Costs % of Rev*	*70.0%*	*66.0%*	*63.0%*
8				

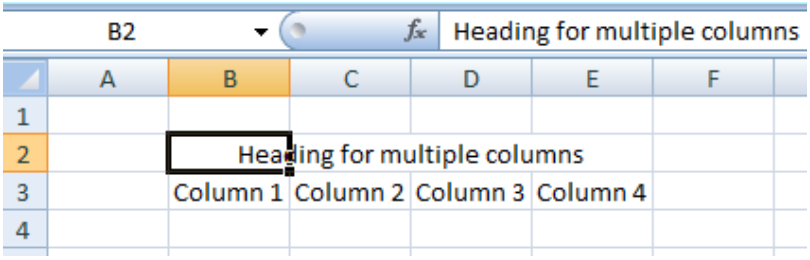
	A	B	C	D
1				
2		Year 1	Year 2	Year 3
3	Revenue	$10.0	$11.0	$12.2
4	*Growth*		*10.0%*	*11.0%*
5				
6	Costs	7.0	7.3	7.7
7	*Costs % of Rev*	*70.0%*	*66.0%*	*63.0%*
8				

Don't merge cells. Avoid using the Merge & Center button (on the Home ribbon) to center text across columns. I know what you're thinking: "I want to center my heading across multiple columns." The issue is the merging part. Merged cells make your model more annoying to work with for several reasons:

- Merged cells interfere with [CTRL] + [SPACE], because they force you to select all the merged columns
- If you're deleting or moving cells around, you'll get warnings all the time
- If you try to insert a cell and shift the rest of the column down, Excel will need to unmerge anything in that column
- When copy-pasting over merged cells, you'll get errors saying "you can't paste this here".

Instead of merging, you should use **Center Across Selection**. Select the range of columns, with the first cell containing the text you want to center. Open Format Cells with [CTRL] + [1] and tab over to Alignment (or press [A]). Under Horizontal alignment, choose Center Across Selection.

When you Center Across Selection, the heading gets correctly centered as before. But you don't get any of the annoying side effects of merged cells!



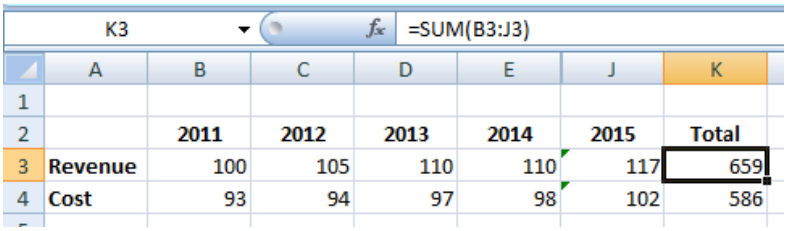
Use good number formatting. For dates, if you enter "Jan 2016", Excel converts it to a default "16-Jan". That's really bad formatting – what the heck is 16-Jan? For dates, make your own number formatting. Press [CTRL]+[1] to format cells, and go to the Numbers tab. I prefer the Custom number formats for dates, because you can specify your formatting with combinations of m, d, y, spaces, and special characters. I like the following custom formats (in each case, the actual date value is Jan 10, 2016):

mmm yyyy	Jan 2016
m/d/yyyy	1/10/2016
mm/dd/yyyy	01/10/2016

For number formatting other than dates, use consistent formatting. Dollar amounts should have a consistent number of decimals throughout the model. Don't overuse dollar signs – a good idea is to dollar format the first row and the totals row of numbers, and omit the dollar sign for other rows. Row after row of dollar signs looks bad.

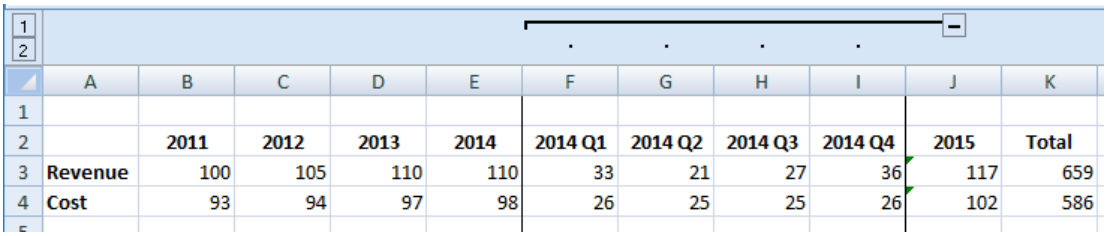
Take advantage of these three keyboard shortcuts for number formatting: [CTRL]+[SHIFT]+[1] for a numeric format with thousands commas; [CTRL]+[SHIFT]+[4] for currency formatting with thousands commas; [CTRL]+[SHIFT]+[5] for percent format. You might still have to adjust the number of decimals, but these are quick shortcuts to get decent number formatting. You can use [ALT] [H] [9] to reduce decimals.

Never hide columns or rows. There's a much better way to hide rows and columns than the conventional way (right-clicking the heading and choosing Hide). The problem with hiding is this: at some point, you will trick people with hidden columns. They'll get a calculation wrong because of hidden columns, which could be disastrous if unnoticed. Take the example below: K3 has the sum of 2011-2015 revenues. Did you notice the resulting total is wrong?



	A	B	C	D	E	J	K
1							
2		2011	2012	2013	2014	2015	Total
3	Revenue	100	105	110	110	117	659
4	Cost	93	94	97	98	102	586

The problem is caused by hidden data in columns F through I. If you look closely at the headings you'll see these columns are hidden, but it isn't obvious! The solution: **use grouped columns/rows**. We covered grouping in Chapter 3; as a reminder, use [ALT] [D] [G] [G] to group selected columns/rows, or find Group in the Data ribbon. Grouped columns look like this:



	A	B	C	D	E	F	G	H	I	J	K
1											
2		2011	2012	2013	2014	2014 Q1	2014 Q2	2014 Q3	2014 Q4	2015	Total
3	Revenue	100	105	110	110	33	21	27	36	117	659
4	Cost	93	94	97	98	26	25	25	26	102	586

Notice that nothing's changed from the above screenshot, except that the previously hidden columns are now grouped. Click the little box to hide or unhide columns; grouped hidden columns are pretty intuitive.

Use comments. Add comments to a cell by right-clicking and Insert Comment. Use comments to make notes to yourself and others on why you did something in your model. Use them to explain complex formulas or certain assumptions. It's better to have more comments than less! As a bonus, you can also copy-paste comments using paste special.

The keyboard shortcuts for working with the keyboard are: [SHIFT] + [F2] to add or modify a comment, [ESC] to close it, and [DEL] (after [ESC]) to delete it. No need to use the mouse at all!

Other Modeling Advice

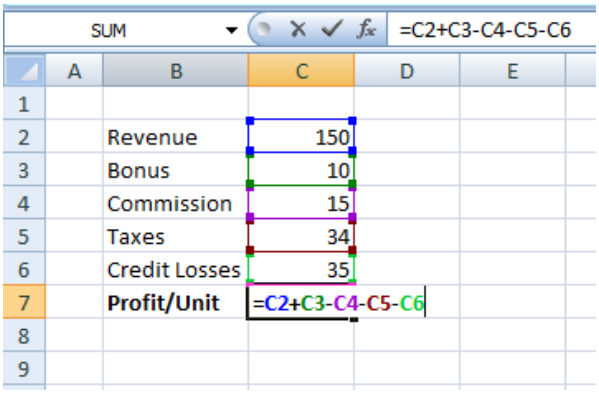
Let's quickly review some more good modeling advice. These are not as critical as the Golden Rules of modeling, but they are still good practices. We will cover:

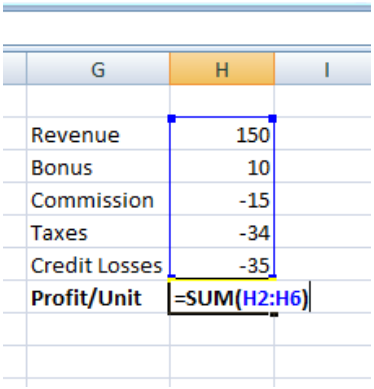
- Consistent sign convention
- Check your formulas, especially totals
- Building error checking cells
- Anchoring and named references
- Using comments
- Handling circular models with iterative calculations (if you must!)
- Testing, testing, testing

Particularly for financial models with cash flows, you'll want to use **consistent sign convention**: positive values mean cash coming in, and negative values mean cash going out. Don't use positive values for everything!

> Consider the example of Fidelity's Magellan Fund from way back in 1995 (when it was the largest mutual fund in the world). In a calculation used to determine year-end distributions to shareholders, at one point the fund's gains and losses were copied to a spreadsheet. But apparently some minus signs were missing: "the accountant mistakenly transcribed a $1.3 billion loss as a gain."[45]
>
> I don't know whether consistent sign convention (or just a newer version of Excel!) would have prevented this error, but I know it's easy to make a similar mistake if you omit the negative sign from losses, costs, or other types of negative cash flows.

Using negative and positive numbers correctly will help you avoid missing inputs into formulas. Consider the example below (left screenshot), where we're trying to calculate net profit based on some positive and negative items:





The formula on the left is cumbersome! You have to remember which items are positive or negative, and it's easy to make a sign mistake when calculating the final result. A much better way is shown in the right side screenshot above: the inputs have the correct signs and the profit calculation is a simple SUM.

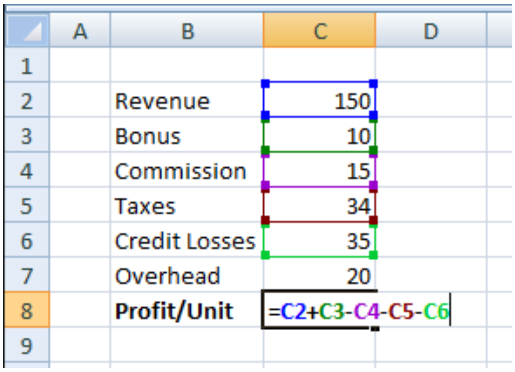
Let's add a new cost called "overhead". Once you've done this, you need to remember to include overhead in your complex Profit/Unit formula. But you do get a little help from Excel if you just use SUM. Notice the two different results on the right side screenshot below; their profit formulas are:

=SUM(H2:H7) and =SUM(K2:K6)

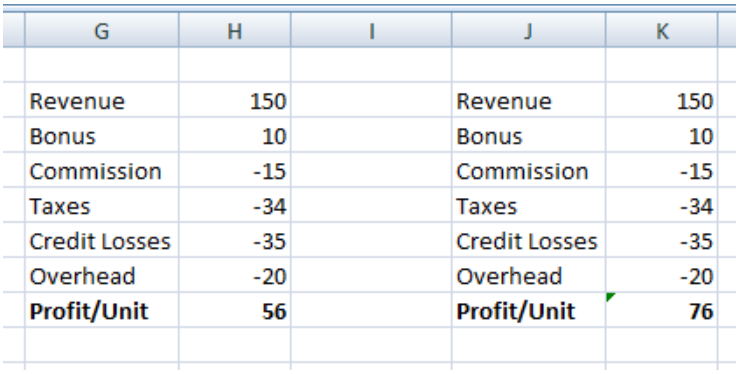
[45] "Magellan Error Is Explained". The New York Times. http://www.nytimes.com/1995/01/04/business/magellan-error-is-explained.html

In column H, Excel automatically updated the total to include the new item. That gets the correct result of 56. But it's dangerous to assume this will always happen! Be careful not to end up with the result in column K, where the Profit/Unit is wrong because the formula wasn't updated.

Notice the little green triangle in cell K8 below. It comes with a hint that says the "Formula omits adjacent cells". That's a nice catch by Excel, but you shouldn't rely on it. Make a habit of **checking your formulas** to avoid potentially serious errors. Remember the alternating [F2] and [ESC] technique from Chapter 2!



	A	B	C	D
1				
2		Revenue	150	
3		Bonus	10	
4		Commission	15	
5		Taxes	34	
6		Credit Losses	35	
7		Overhead	20	
8		**Profit/Unit**	=C2+C3-C4-C5-C6	
9				



G	H	I	J	K
Revenue	150		Revenue	150
Bonus	10		Bonus	10
Commission	-15		Commission	-15
Taxes	-34		Taxes	-34
Credit Losses	-35		Credit Losses	-35
Overhead	-20		Overhead	-20
Profit/Unit	**56**		**Profit/Unit**	**76**

For complex models, try setting up **error checking cells** in a prominent place (near the top of the model). The idea is this: maybe some values in the model must always add up to 100%, or two cells must be equal to each other. Put a formula like `=SUM(B10,D13,Sheet2!E5)=100%` into an error checking cell. You don't even need an IF formula around it; the `=100%` is a logical test that returns either TRUE or FALSE. If the result is FALSE, you'll immediately see that something is wrong.

In academic studies of Excel errors, cell reference errors are identified as the second most common mistake. To help avoid reference errors, use **named references** for your assumptions. These two formulas do the same thing, but the second one is much easier to troubleshoot:

```
=C3*(1+assumptions!B4*assumptions!C7) / B3
=C3*(1+inflation*multiplier) / B3
```

We covered how to name cells and manage your named references in the beginning of Chapter 5. They make formulas more readable, and also help avoid problems when you drag formulas down or right: these named cells are always anchored to the original cell! Of course, there should be a common-sense limit to naming cells, because it is a bit time-consuming.

Remember to also use proper **anchoring** with [F4] to avoid formula reference problems.

Sometimes your model has **circular logic**, whether intentionally or unintentionally. We looked at dealing with circularity in Chapter 5. Remember that you'll need to either eliminate the circularity, or turn on **iterative calculation** (under Excel Options, Formulas, Calculation Options) to converge on a solution.

Use round numbers and limit the digits in models. Yes, it's very impressive that your company's Year 5 revenues are projected to be $12,789,292.37, but there's no point in showing that level of detail! It's very difficult to read, especially at a management level. Financial models should be denominated in thousands (or millions) of dollars. Label near the top that you're using $000s (thousands of dollars), and your cell's value should appear as $12,789.

When you've finished building your model, all the formulas are consistent, and everything links back nicely to some clearly labeled assumptions – are you done? Not so fast! Don't forget to **test your model!**

- Enter various assumptions and make sure the output results make intuitive sense. If your sales go up, does your income go up? It probably should!
- Try outlandish assumptions like negative numbers, very large values, and decimals. Do they break your model?
- Pay particular attention to complicated formulas such as IF, MAX/MIN, and VLOOKUP. Try passing various values to these functions to test that they work correctly; it's easy to make a mistake! Don't forget to use error handling with VLOOKUP in case the input doesn't match anything in the lookup table.

Quickly Understanding How an Excel Sheet Works

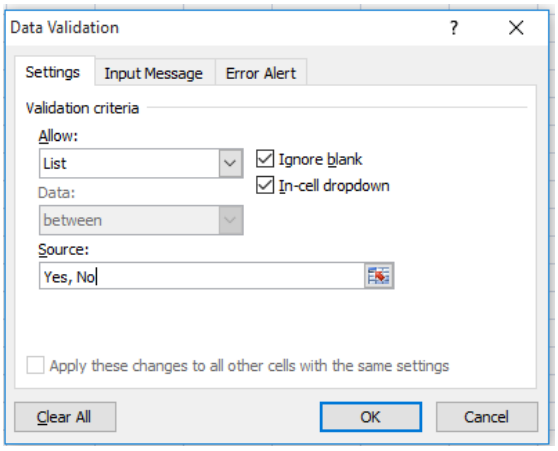
Somebody sends you a model that you've never seen before. How can you quickly understand what it does and how it works? (Hopefully the model's creator followed the three golden rules of modeling!)

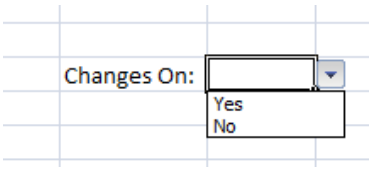
Do the following as you figure out all the ins and outs of someone else's Excel sheet:

- **Check for hidden rows or columns**. Usually there aren't any hidden rows or columns, but some people have a nasty habit of using them. You might notice column D right after column B – that's your giveaway that column C is hidden! To quickly unhide all columns and rows, press [CTRL] + [A] to select everything on the sheet (you might have to do it twice). Then right-click one of the columns and choose "Unhide". Do the same for rows by right-clicking the row headings on the left.
- **Check for any hidden sheets**. Another rare and bad habit is hiding entire sheets. Right-click any one of the sheet tabs at the bottom of the window, and see if Unhide is grayed out. If it isn't grayed out, there are hidden sheets, so unhide them!
- **Trace dependents and trace precedents.** These two are critical, and probably the most important tools for figuring out how a model works. Use trace dependents in particular to follow how a particular assumption, constant, or formula flows through the rest of the model.
- **Use [F2] and [ESC] to look inside cells and check formulas for consistency.** This is a great way to catch any formula mistakes: open up a cell's formula with [F2], check the color-coded cells that are the precedents of this cell, hit [ESC] to back out, and use the [ARROWS] or [ENTER] / [TAB] to move to the next cell. Then repeat the same [F2] and [ESC] process. You can go pretty fast like this.
- Alternatively, **use [CTRL] + [~] to show all formulas and check them for consistency.** This shortcut converts all cells to show their underlying formulas, and you can easily compare many formulas at once. Don't freak out when you see this! Press [CTRL] + [~] to toggle back.
- **Look for named references.** If the author of the model was considerate, they named some key assumptions and cells. Check the name manager (Formulas, Name Manager) to look for any clues.
- **Check for macros, especially ones that auto-execute.** This one's probably overkill, but to be thorough you'll want to look for code hidden in VBA. We'll talk about VBA in the next chapter; for now, just keep in mind that macros can have a major impact on how a model works.

Data Validation: Restricting Inputs

Data validation is a good way to make your models "idiot-proof" and potentially easier to use. The idea is to restrict what kinds of values can go into a cell. To apply data validation to a given cell, select the cell and choose "Data Validation" under the Data ribbon. The Allow dropdown gives you a lot of useful data validation options.





Whole Number or **Decimal** restricts to specific numeric inputs, including minimum and maximum values.

List lets you specify a specific list of values that this cell can take – for example, Yes or No. You can specify the possible values in the Source input box in one of two ways: either type in the actual values separated by commas (Yes, No), or select a range of cells that contains the values (C1:C3).

With List validation, keep the In-Cell Dropdown option checked. This creates a dropdown menu within the cell, which is a very nice way to show users which inputs are allowed. You can use [ALT] + [DOWN] to use the dropdown menu with just the keyboard.

Finally, you might find yourself applying the same data validation to multiple cells. Don't set data validations one cell at a time – it's time-consuming. Instead, you can **copy-paste data validation from cell to cell**. Use paste special and select Validation as the thing to paste.

Protect Sheet: Locking Cells

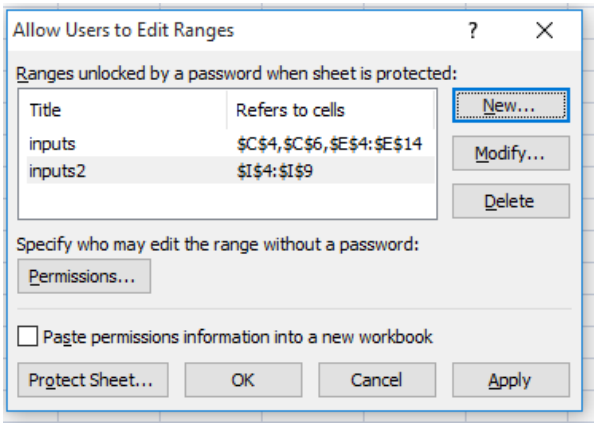
To go a step beyond data validation, you can lock certain cell ranges, and even entire sheets, so they cannot be edited. As a practical use of this feature, you might have a pricing model you want to give to your employees. You don't want them to mess with any of the formulas – you just want them to enter the inputs. In this case, I highly recommend making the inputs section prominent and different from other sections; you might give these cells a light blue background, and label them "inputs".

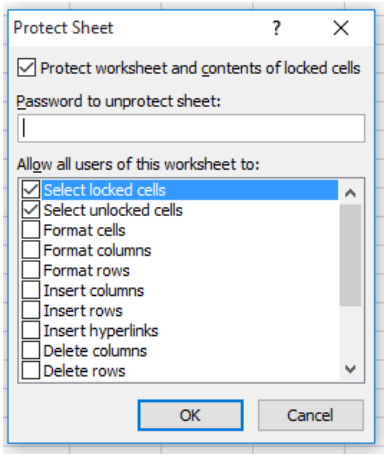
> To illustrate the importance of locking cells and correctly identifying inputs, consider a mistake made by the Hungarian government when they sent out a spreadsheet for contractor bids. The file they sent required contractors to input numbers into one column, while the second column was automated. Apparently "25 of the 28 bidders misunderstood the spreadsheet and manually filled in the second column."[46]
>
> Don't let this happen to you! To ensure that user inputs stay within your set boundaries, you can lock the rest of the sheet from all editing.

Start with **specifying which cells can be edited** (once the sheet is locked, you can't change which cells are editable, and you would need to unlock it again to change them). Under the Review ribbon, click Allow Users

[46] "KDB fines Central Services Directorate for misleading spreadsheet". Budapest Business Journal. June 28, 2010. http://bbj.hu/economy/kdb-fines-central-services-directorate-for-misleading-spreadsheet_53252

to Edit Ranges. Click the New button to add the first range of editable cells. Select single cells, ranges of cells, multiple ranges – whatever you want to keep editable. In the screenshot below I created two ranges; the first one has a mix of individual cells and ranges.





Once you're satisfied with the editable cells, go back to the Review ribbon and choose Protect Sheet. You'll see a password input box, followed by a long list of checkboxes for what to allow. The default setting works well: allow users to select locked or unlocked cells.

You can choose a password, which will be required for unlocking the sheet. **Assigning a password does not mean your sheet is securely protected**. Anyone could break this password easily; I'll show you how in Chapter 9. I would add a password only as a deterrent against casual tampering. Ignore when Excel says "Caution: if you lose or forget the password, it cannot be recovered" – that's just not true.

Once the sheet is protected, you can't do much to it. You can change the editable cells, but everything else is locked down. Click Unprotect Sheet (and enter the password) to go back to a normal unlocked sheet.

There's also a button to Protect Workbook. A protected workbook doesn't lock any sheet's cells, but it does prevent adding, deleting, hiding, and un-hiding sheets. This feature can also have a password, and that password can also be broken within minutes.

Scenarios, Sensitivity Analysis, and "What-If" Analysis

Once your model is fully set up, you'll want to use it to understand how different assumptions affect your outputs (that's kind of the point of building a model!). The skill of advanced modeling, beyond the mechanics of building good formulas and keeping things organized, lies in **understanding the true relationships underlying the model**. Use scenario analysis and sensitivity analysis effectively to better understand what's going on.

We looked at the OFFSET function specifically for scenario analysis in Chapter 4. As a reminder, the idea is this: stack up your key assumptions in one place (one column or one row). You can specify several sets of assumptions; for instance, you can add an optimistic, pessimistic, and middle scenario. Then use the OFFSET function to switch the entire set of inputs between these scenarios, based on a single input value of 1, 2, or 3.

Sensitivity analysis looks at how an output changes with respect to two of the inputs. For instance, show total profit as it relates to annual growth and profit margin, or show price as it relates to quantity and location. Most good models have two key inputs, and you'll want to see how those inputs affect the overall result.

We'll build a sensitivity table together to do that. The end result looks like the table on the right side of this screenshot, starting in column F:

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2		Growth	0%			*Change in Operating Profit*						
3		Price Increase	10%					**Price Increase**				
4		Base Price	200			**Units**	29%	**-20%**	**-10%**	**0%**	**10%**	**20%**
5		Base Units	100			**Growth**	**-20%**	-88.6%	-65.7%	-42.9%	-20.0%	2.9%
6							**-10%**	-72.9%	-47.1%	-21.4%	4.3%	30.0%
7			**BaseLine**	**New**			**0%**	-57.1%	-28.6%	0.0%	28.6%	57.1%
8		Revenue/Unit	$200	$220			**10%**	-41.4%	-10.0%	21.4%	52.9%	84.3%
9		Units Sold	100	100			**20%**	-25.7%	8.6%	42.9%	77.1%	111.4%
10		Revenue	$20,000	$22,000								
11												
12		COGS/Unit	$50	$50								
13		COGS	$5,000	$5,000								
14		SG&A Fixed	$8,000	$8,000								
15												
16		Op Profit	$7,000	$9,000								
17		*Change*		***29%***								
18												

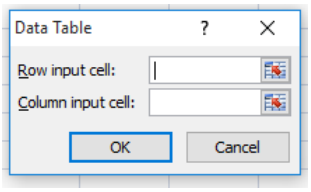
The sensitivity table's output shows changes in operating profit based on price changes and unit sales changes. A 20% price increase and a 10% units growth results in an 84.3% profit increase. And a price cut of 10% that increases units by 20% increases profits by 8.6%.

To start building this table, you'll need to identify **three key cells in your model**: two input cells and one output cell. In my example these are: growth in C2, price increase in C3, and change in operating profits in D17.

To build your sensitivity table, start in a blank area of the sheet so you have some space. The top-left corner of the table must be a cell reference to the model output; in the screenshot, it's the cell in the corner with 29% value (my G4 references D17). From this cell, go down and to the right to fill out possible input values for each input. You can use as many rows and columns as you want; between 4 and 10 rows and columns is the sweet spot. The actual values are easy to change; I used percentage changes from -20% to +20% for both variables.

So that's the setup: a row and a column of input values, intersecting at a corner cell that is the output value. Here's the screenshot so far:

29%	**-20%**	**-10%**	**0%**	**10%**	**20%**
-20%					
-10%					
0%					
10%					
20%					



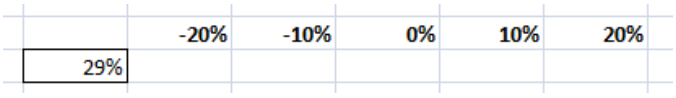
Next, select this entire 6x6 area starting with the top-left cell. Then go to Data ribbon, What-If Analysis, and choose Data Table. The Data Table popup asks you for a Row Input Cell and a Column Input Cell. This step connects the data table to the two input cells we identified before (C2 and C3 in my example). We'll have

price increases along the rows, so the row input cell is C3. Growth is along the columns, so the column input cell is C2.

When you click OK, the sensitivity table values are populated. Now we just need to clean up the number formatting and label the rows and columns (otherwise, who knows what the numbers mean?). You'll end up with a sensitivity table like the example above.

To spice up this table, I like to apply some conditional formatting to make the numbers stand out better. To do this, select the entire range of output values (without the headings), and under the Home ribbon, Conditional Formatting, try one of the Color Scales (available in Excel 2007 and newer). You'll end up with a colorful sensitivity table that's even nicer than the original screenshot above.[47]

It's possible to make a one-dimensional sensitivity table as well, using one input and one output. The setup for this is slightly different: use a single row or column for headings, and offset the output cell by one. In this case we'll have row labels, and the next row will be the output cell.

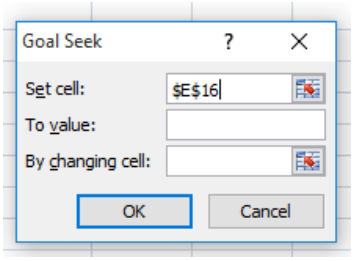


In the Data Table popup, you'll just choose a cell for Row Inputs, and leave Column Inputs blank.

What about sensitivity tables on three variables? There's no good way to do that. You could maybe set up multiple two-dimensional data tables, then use a macro to iterate a third variable and then copy-paste the table results each time. Honestly, though, you're much better off narrowing down to the two most important inputs and sticking with a two-dimensional sensitivity table.

Goal Seek is interesting. It's another option under "What-If Analysis". The popup itself is more or less self-explanatory: set a cell to a value, by changing another cell.

Example: you can tell Excel to set the change in profits (E16) to a value of 30% by changing cell C3 (the price increase). Excel will solve for the price increase you need for profits to increase by 30%. Use this instead of trial and error if you have a specific target for the output.



If you've followed along with me so far and you're using these Excel techniques in your day-to-day work, you are easily among the top 1% of Excel users. You can navigate and manipulate spreadsheets efficiently with the keyboard; you're using advanced formulas to your advantage; you can analyze and graph data; and your models are clear, accurate, and meaningful. Great job so far!

The next chapter will be a little different, as we dive into the world of programming. Using macros to your advantage adds a whole new level to your work. Keep reading to become a top 0.1% Advanced Excel user!

---

[47] I would have loved to show you a screenshot of the colorful sensitivity table, but color ink doesn't come cheap!

8: Macros and VBA

Do you find yourself regularly doing repetitive, time-consuming work in Excel? Let's fix that!

With Macros and VBA, you'll achieve major productivity gains in your routine work. You'll get more done, in less time, more accurately. In this chapter you'll learn the building blocks for automating tasks within Excel, such as:

- Send an email with an attachment to everyone on a list
- Add a specific formula to a cell on every sheet (even if you have dozens of sheets)
- Change pivot table filters
- Delete empty rows

Macros are programs that run on top of an Excel sheet. They let you manipulate the contents of cells, execute every available Excel command, work across multiple sheets and files, and so on. Just about anything you can do manually in Excel, you can also run with macros.

VBA (Visual Basic for Applications) is the programming language for building macros.

In this chapter we'll work through several examples together, so you get comfortable with VBA syntax as well as the logic of solving problems with macros. There will be a lot of typing! The best way to learn is to type the actual code into Excel yourself, rather than copy-pasting code from the web; the act of typing code makes it easier to remember the syntax. You'll probably make mistakes along the way, which is OK! Everyone does; mistakes and typos are a natural part of programming.

If You Need Something, Google It!

This advice applies to Excel in general, but especially to VBA. Whatever you're trying to do in VBA, chances are someone else has solved the same problem already. So open up your favorite web browser and search for "Excel VBA <something>". Examples:

- Excel VBA save file as
- Excel VBA paste special
- Excel VBA run macro in another workbook
- Excel VBA insert row

You may not find a solution to the entire macro you want to build, but if you break it down into individual tasks, you'll definitely find the answer. If your goal is to be productive and solve problems, the easiest way is to copy someone else's solution![48]

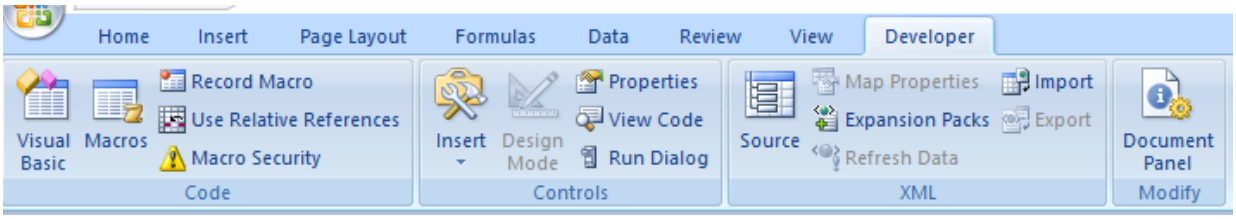
Quick Start: Recording Macros

We'll start the easy, lazy way: recorded macros. Actually writing code will be far more powerful and versatile, but recording is a nice introduction.

Macros are on the Developer tab on the ribbon. If you don't see a Developer tab, turn it on in Excel Options. For Excel 2010 and newer, look for the heading Customize Ribbon. It'll show a list of tabs, with Developer unchecked. Check the box and your developer tab is ready to use.

For Excel 2007, go to Popular under Excel Options. Check the box for Show Developer Tab in the Ribbon.

The Developer tab looks something like this (it varies slightly depending on your Excel version):



We're now ready to record a macro. As a gentle introduction, this first macro will take a column of data and make two copies of it to the right. My example sheet has the numbers 1, 2, 3, 4, 5 in a column starting in cell B3 (follow along by setting this up yourself):

	A	B	C
1			
2			
3		1	
4		2	
5		3	
6		4	
7		5	
8			

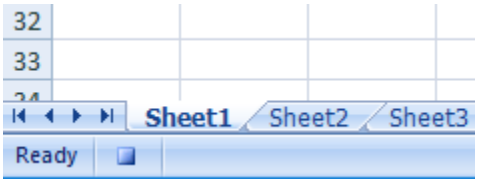
We'll record a macro that does the following: select this data, copy it, and paste it into columns C and D. To record a macro, click Record Macro in the Developer tab ([ALT] [L] [R]). You'll get a popup asking for the Macro name and some other settings. Let's name it "CopyData2x". Macro names are not allowed to have spaces.

Still in the Record Macro popup, you can assign a shortcut key which lets you run the macro with a keyboard shortcut of your choice. This is a great feature, but be careful with it! Don't assign one of the (numerous) existing Excel shortcuts. I recommend using [CTRL] + [SHIFT] + a letter; there are very few built-in shortcuts like that. Let's assign [CTRL] + [SHIFT] +[C].

For the Store Macro In dropdown, This Workbook is fine. The macro will work within this particular Excel file, which is all we want. Description is optional; I normally just leave it blank. Ready? Click OK.

[48] Feel free to copy the examples in this chapter for personal or business use, but don't publicly display them as your own.

Now what? Nothing happened – except Excel is recording a macro now. You can tell by the little “stop recording” icon that appears at the bottom left of the screen.



When you see the square “stop” icon, the macro is recording. If you click somewhere, move around with the keyboard, execute a command, or delete something, all of these actions get stored within your macro. Don’t make your recording too long; it will generate a lot of code behind the scenes! If you click the stop button, it ends the recording.

Right now, let’s keep recording and tell Excel what the macro should do. Follow these steps exactly:

1. Click on B3 (which contains the first cell of data)
2. Use [CTRL]+[SHIFT]+[DOWN] to select down to the end of your data (don’t do this with the mouse; that actually means something different for the macro. I’ll explain shortly)
3. [CTRL]+[C] to copy
4. [RIGHT] to go to the next column
5. [CTRL]+[V] to paste
6. [RIGHT] to go to the next column
7. [CTRL]+[V] to paste
8. [ESC] to unselect the copied area
9. Click the stop button to stop macro recording

Let’s try out our new macro. First, clear out the copied columns of data. We want the macro to fill these out again. To run the macro, go back to the Developer tab and click Macros (or use the [ALT] + [F8] shortcut). You’ll see a list of all your macros, including “CopyData2x”. Select it and click Run.

	A	B	C	D	E
1					
2					
3		1	1	1	
4		2	2	2	
5		3	3	3	
6		4	4	4	
7		5	5	5	
8					

Boom! The macro just ran, and it did exactly what you walked through in the eight steps above. You can clear out columns C and D and run the macro again to get the same result.

Now, remember we assigned a shortcut for this macro: [CTRL]+[SHIFT]+[C]. Clear columns C and D again and hit the shortcut to run the macro. I hope this gets you excited about the new possibilities for keyboard shortcuts: you can run any Excel command(s) with the keyboard using macros!

Save this Excel file so you can use the macro later. To save a file containing a macro, you’ll need to save as Macro-Enabled Workbook. Hit [CTRL]+[S] to save (if saving for the first time), or use Save As. In the Save As Type dropdown choose Excel Macro-Enabled Workbook.

It's very important that you save as Macro-Enabled Workbook. If you save as a regular workbook, Excel gives you a message that the VB Project features cannot be saved in macro-free workbooks. They're not kidding – if you save this file as non-macro-enabled, then close it and re-open it, your macro will be gone!

The error message says "to continue saving as a macro-free workbook, click Yes." You'll want to **click No** – and then save as macro-enabled. With the macro safely saved, let's keep going.

M**acros do exactly what you instruct them to do**. No more, no less. They won't anticipate or understand your true intentions; they just copy what you do. To show you what I mean, try this: move your original data over one column, from column B to column C. Now run the macro.

Oh no!

Here's what happened: Excel still copied data from B3 (empty) through the end of the contiguous range downward (which is the entire empty column). Then that empty column got pasted over columns C and D. The result is that the data in column C is gone.

And it gets worse… undo doesn't bring the data back! **There's no undo for macros**. The best you can do at this point is close the file without saving, and re-open it (hope you had it saved!). Otherwise, you will need to re-create the lost data.

Macros are **sheet-dependent**. Switch to another sheet and fill it up with some content (an empty sheet won't do much). If you run the macro now, unexpected things will happen. It will apply the same copy-paste steps to this sheet, not the original one. Later in this chapter we'll learn macro code that specifies the sheet you want to use.

Let's modify the macro so it's a little more flexible. Forcing it to always start on cell B3 is not ideal; let's make the starting cell selectable.

There's only one thing we need to do differently this time: have cell B3 selected before you start the macro. Record a new macro (maybe called CopyV2), but follow steps 2-8 from above (skip the first step).

What's different? The macro's frame of reference changed. The new macro works regardless of where the data is, as long as the first cell is selected when you run it. Try this out by moving your data to column C (or anywhere else). Select the first cell of data, run the macro, and it works!

So, let's recap the main lessons for recorded macros:

- The macro copies exactly what you do! Take deliberate actions while the macro is recording
- Don't forget to stop recording when you're done
- Save as a macro-enabled workbook so you don't lose your macro
- There is no undo for macros, so save often!

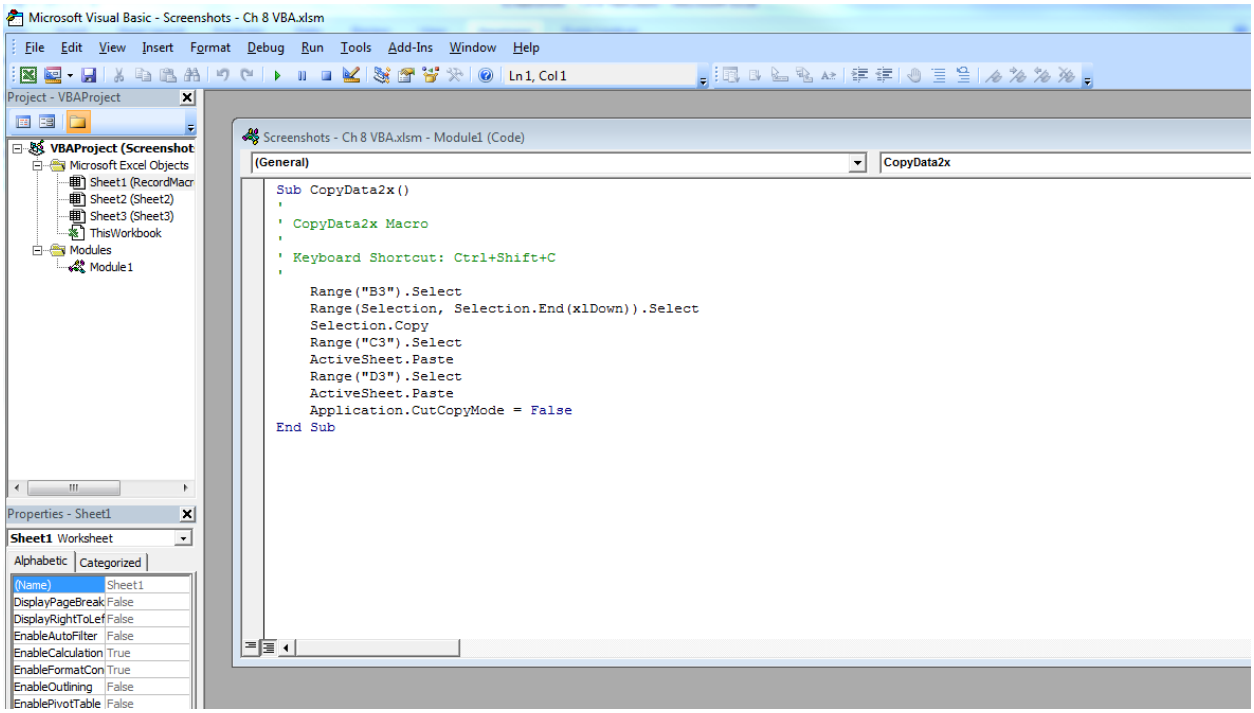
When you open a macro-enabled workbook, Excel gives you a little warning. It asks if you want to **enable macros**. For your own Excel files and ones that you trust, it's fine to enable macros. But be careful with strange files from strange people! Macros can do all sorts of nasty things, including deleting files on your computer. Macros can even run by themselves. For security reasons, if you don't trust the file, don't enable macros.

Quick Start: VBA Code

VBA stands for Visual Basic for Applications; it's the language of macros. When you record a macro, Excel generates VBA code behind the scenes. If you followed along and recorded a macro earlier in this chapter, you can now take a look at the code behind it (if you didn't do that, just quickly record a macro that does something.)

Press [ALT] + [F11] to open the VBA editor. This is your new favorite shortcut for macros! Use it to toggle back and forth between Excel and VBA. You'll see a screen like this (although the main window with code might not be showing yet):

> **Advanced Tip**
>
> Use [ALT] + [F11] to switch between VBA and the Excel sheet.



The essential parts of this screen are:

- At the top left, you have a project tree diagram of your current workbook(s). It contains Microsoft Excel Objects, which are your worksheets. If your file contains macros, then you'll also see a Modules section (if you don't see any modules yet: Right-click, Insert, Module). If you have multiple Excel files open, you'll see all of them in this list.
- Below the project tree is a list of Properties. You will rarely use these.
- The main area of the screen starts out blank. When you double-click Module1 in the project tree, its code opens up. The screenshot above shows the code window for Module1, which is the VBA code from your previously recorded macro!
- At the top of the window, you have a toolbar specific to VBA.

Note: when you have multiple workbooks open, make sure you're editing the correct module within the correct file! Look for the name of the file at the top of the code window; in my screenshot it is "Screenshots - Ch 8 VBA.xlsm".

What does the code inside Module1 mean? By the end of the chapter, you'll understand every line of it. For now, here's a quick rundown:

- The whole thing is bracketed by `Sub` and `End Sub`. Sub stands for subroutine, but that doesn't matter: Subs are macros. Your macro code starts with `Sub` and the name of the macro, followed by parentheses. It ends with `End Sub`.
- The first few lines are green and begin with an apostrophe. These are comments, meaning they are just notes for you, the human looking at the code. Excel doesn't do anything with commented lines; they're explanations of what the macro is doing. You can put commented lines anywhere in the macro. Deleting them won't make any difference.
- Then you have the actual VBA code. If you read this line by line, you can probably understand what's happening. For instance, `Range("B3").Select` selects the cell B3. And `Selection.Copy` copies whatever is selected. All this macro does is select something, copy, and paste.

Recording a macro is a nice way to fast-track creating macros. As you've just seen, a recorded macro generates the building blocks of code, which you can then edit yourself. When I work on a complicated macro, I often record several little macros just to remind myself what the VBA code is for the command I need. What's the VBA code to run Goal Seek in Excel? Record a macro to see – there's no need to memorize obscure commands!

Now that you've seen some macro code in action, we'll start learning to write new code from scratch.

VBA Interfacing with the Excel Sheet

Go to the VBA editor with [ALT] + [F11].

Add a new module: right-click in the tree structure on the left side, and choose Insert, Module.[49] You'll have a new module to work in (Module1 or Module2).

I'll make a new macro called NewMacro which will do the following: take whatever value is in cell A1, multiply it by 2, and output that value into cell A2. Start typing into the VBA editor:

```
Sub NewMacro
```

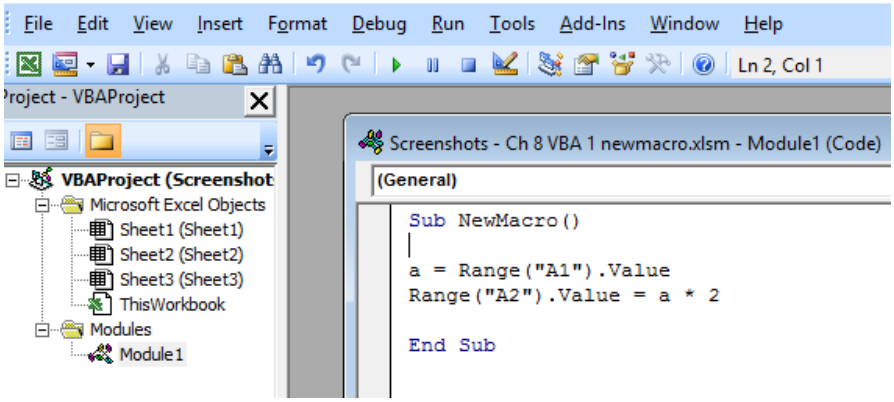
Excel enters the little parentheses after the name of the macro, and also fills out the `End Sub`. Convenient!

Now, type in the rest of the code, just these two lines below (make sure they're before `End Sub`, otherwise they're not actually part of your macro):

```
a = Range("A1").Value
Range("A2").Value = a * 2
```

Run this macro! From the VBA editor, you can run the macro by clicking the little green triangle icon on the toolbar, which is like a play button (it's below Debug). **Make sure your text cursor is inside the macro you want to run.** Or use the [F5] keyboard shortcut within VBA.

[49] If you've accidentally closed the project tree, you can open it back up with View, Project Explorer.



After the macro runs, switch back to the Excel sheet with [ALT]+[F11]. If you had some value in A1, you now have twice that value in A2. If A1 was blank, A2 contains 0.

If you had text as opposed to a number in A1, VBA gave an error saying "Type mismatch." For now, don't worry about this error. Just click End to get rid of it. If you clicked Debug, that's fine too: Excel then highlights the row of code that is causing the problem. You'll need to stop the macro (use the stop button on the VBA toolbar, to the right of the play button). To fix the problem, change cell A1 on the Excel sheet to contain a number; then the macro will run properly.

Let's look at the VBA code. Each line in VBA consists of a separate command; a new line means we're executing a new command. The first line is:

```
a = Range("A1").Value
```

The equals sign means we change the value of the thing on the left to be equal to the value of the thing on the right. Here we set the value of a **VBA variable** called `a`. VBA variables are different from Excel named references, which are names of cells and can be used in Excel formulas. VBA variables only exist in VBA.

We are assigning `Range("A1").Value` to the variable `a` using the `=` sign. `Range("A1")` is the correct way to reference cell A1 in VBA (cell A1 on which sheet? The sheet you are on currently).

Notice that you can't do this: `a = Range("A1")`. The range called "A1" is an object in VBA, and objects themselves cannot be assigned to variables. Cell A1 has a value, but it also has formatting, a formula, and even data validation. All of these are properties of cell A1. To extract the value of A1, use `.value`.

Moving on to the second line:

```
Range("A2").Value = a * 2
```

This time, the reference to a cell's value is on the left side of the equals sign. We are assigning a value to this cell, the value of `a*2`. The asterisk means "multiply", just like in Excel formulas. With that covered, you now have a simple macro that pulls data from the Excel sheet and makes changes to it.[50]

[50] I like using this example as an interview question for those who know VBA: "Build a macro that takes a value from the sheet, multiplies it by two, and returns the value to another cell." You'll ace this question now; lots of people don't.

VBA Variables and Simple Math

Let's look at variables and simple math operations in VBA. Type in the following code exactly as below (yes, I'm going to ask you to do a fair amount of typing in this chapter; it's the only way to learn):

```
Sub math()
Dim a As Integer, b As Integer, c As Integer
Dim n As Double, p As Double, q As Double

a = 5
b = a + 5
c = 5 / 2

Range("A1").Value = a
Range("A2").Value = b
Range("A3").Value = c

n = 2.5
p = 3
q = ((n + p - 0.5) / 10) ^ 2

Range("B1").Value = n
Range("B2").Value = p
Range("B3").Value = q

End Sub
```

Run this macro, and you'll get some numbers output on the Excel sheet in columns A and B, rows 1-3. Are the results what you expected? Let's look at the code line by line and see.

- `Dim a As Integer` is used to declare the variable `a` of type `Integer`. You are telling the computer that you will be using variable `a` in the macro, and that it can only store integers (meaning positive or negative whole numbers, no decimals).
- `Dim n As Double` declares the variable as `Double`. The `Double` data type can store all real numbers, positive or negative, including decimals.

Data types determine what kinds of values your variables are allowed to contain. Integer and double are the most common for numbers.[51] You will also use the `String` data type for storing text.

Declaring variables with `Dim` is optional but recommended. You could skip those two lines and the macro would still run, but sometimes it would return a different result! We'll see why in a moment.

- The next three lines set the value of `a`, `b`, and `c` as 5, 10, and 2.5.
- Next, the macro outputs the values of `a`, `b`, and `c` into cells A1, A2, and A3.
- The second half of the macro works with the three variables declared as type `Double`. The only complicated part is the line that sets the value for `q`: we have parentheses, subtraction, division, and even an exponent! They're the same as calculations in Excel formulas. We are working with real numbers and not integers, so the result for `q` has no unexpected surprises.
- Finally, the macro outputs the values of `n`, `p`, `q` in B1, B2, and B3. The last line ends the macro.

Do the output results make sense? Why does A3 have 2 and not 2.5? Because we declared `c` to be an integer, `c` is not allowed to store any decimals. So the computer drops the 0.5 part and stores just the integer 2 (it doesn't round numbers up or down, it just chops off decimals).

[51] Possible data types for numbers in VBA are: integer, long, single, double. Integer is common for things like row numbers or loop counters. Use double when you might need decimals, such as dollars and cents or general numeric inputs.

Let's run this macro again, but without declaring the variables. Put an apostrophe in front of both lines starting with `Dim` to comment them out. They become green to indicate these are comments that will not be read by the macro.

The result of this second run is slightly different, because A3 is now 2.5! We removed the line that forces `c` to be an integer. The macro automatically assigns a data type to `c` the first time we give it a value. In this case, `c = 5 / 2` is the first time the macro sees this variable, and it knows to make it a real number, using the value 2.5.

What did we learn from this exercise?

- Watch out for data types; they affect the results of your calculations in the case of fractions and decimals
- You can use variables inside macros that don't exist anywhere in the Excel sheet. These are not cells or named references, but VBA variables
- You can do math calculations in VBA and output results to the Excel sheet

In real life, don't use VBA for long calculations. It's much easier to do your math calculations on the Excel sheet with formulas (in fact, this is my Golden Rule #3 for VBA – more on that later). But you'll certainly use named variables and basic math within your macros.

VBA Math Functions: Min, Max, Average

Using math functions within VBA is slightly more effort than using them in Excel. VBA will not recognize something like `Max(1,2,3)` because MAX is a "worksheet function" and not a VBA function. Worksheet functions include MIN, MAX, SUM, AVERAGE, VLOOKUP, ROUNDUP, and so on – basically, every function we covered in Chapter 4.

To use MAX in VBA, use this code:

```
Application.WorksheetFunction.Max(1, 2, 3)
```

There's a slightly abbreviated version that also works:

```
Application.Max(1, 2, 3)
```

And here's how you use SUM:

```
Application.Sum(1, 2, 3)
```

Note: some commands do exist in VBA, such as IF, AND, OR. These have different structures in VBA, compared to the corresponding functions on the Excel sheet. We'll look at IF logic in VBA shortly.

You could technically do a VLOOKUP, or perform other advanced functions, within VBA:

```
Range("D5").Value = Application.VLookup(3, Range("A1:B4"), 2, 0)
```

That's a lot of typing! It's also difficult to read and understand this code, so don't use complicated formulas within VBA. Instead, put the calculation within a cell on the sheet and reference its value in VBA.

Variables Versus Ranges

Let's clear up any possible confusion about variables and ranges before we dive deeper into VBA. You'll want to get comfortable with the distinction between these five concepts:

`X`	Refers to a VBA variable called X. X can have any value, or no value at all.
`"X"`	Refers to the single character “X”.
`Range(X).Value`	Refers to the value of a cell or range on the Excel sheet, depending on the value of the VBA variable X. If X has the value “A1”, then this refers to the value of the cell A1. If X has the value “input_cell”, then this refers to the value of the named range input_cell from the Excel sheet.
`Range("X").Value`	Refers to the value of the Excel range “X”. If you don’t have a named range “X”, then this reference throws an error.
`Range("X1").Value`	Refers to the value of the cell X1 on the Excel sheet.

Examples:

`Range("x").Value = x*2`	Take a range called x on the Excel sheet, and assign it 2 times the value that the VBA variable x has.

`Dim x as string`
`x = "input_cell"`
`Range(x).Value = "x"`
	Declare a VBA variable x as a string. Set that value of that variable to “input_cell”. Then set the value of the Excel named range input_cell to the character “x”.

`A1 = "Hello"` `Range("A1").Value = A1`	Set the VBA variable called A1 to value “Hello”. Then assign that value to the Excel cell A1 so it says “Hello”.

Debugging VBA Code

Once you start working with VBA, you’ll realize it runs into errors all the time. The key to a happy relationship with VBA is to accept that it doesn’t always do what you want the first time, and to troubleshoot your code efficiently when things go wrong.

There are two types of errors you’ll run into: syntax errors and logic errors.

Syntax error means VBA doesn’t understand your code or can’t do what you asked for. It could be because you mistyped something, you’re using symbols (such as periods, equals signs, or parentheses) incorrectly, or something is missing. In each case, it’s relatively easy to find out where the problem is: Excel pops up an error message, offering to End or Debug. This is a common error message:

Run-time error ‘1004’: Method ‘Range’ of object ‘_Global’ failed

What to do about it? Click Debug and Excel will highlight for you the line of code that is causing a problem. Now you know which line to fix. Once you get more familiar with VBA, you’ll recognize exactly what’s wrong. With a “Method ‘Range’…” error like above, your code is referencing a named range that doesn’t exist. We’ll review other common VBA error messages at the end of this chapter.

Logic errors are more difficult to fix. In this case, the macro executes without throwing an error, but doesn’t do what you intended. Maybe the output number is wrong. Maybe you’re stuck in an endless loop (if you ever have a macro going endlessly, **press [ESC] repeatedly to stop the macro**). Fortunately, VBA has tools to help you debug the problem.

To illustrate these debugging tools, we'll start with an example Excel sheet and a simple macro that has some logic errors. The macro is supposed to generate a Fibonacci sequence from two starting values on the Excel sheet. We'll need to fix the errors in the macro. You can download the associated Excel sheet from www.AdvancedExcelBook.com or follow these instructions to recreate the situation:

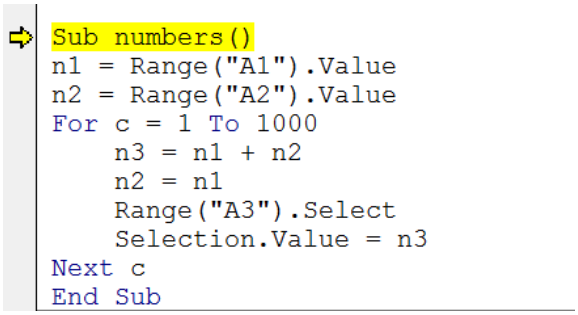
1. Set up the Excel sheet. Start with a blank Excel file, and type "1" in cells A1 and A2. These are the first values in the Fibonacci sequence.[52] We'll use the macro code to complete the rest of the sequence, until we reach 100 numbers.
2. Next, create the macro in VBA. Instead of recording a macro, use [ALT] + [F11] to open the VBA editor, then create a new module. Inside the module, add the following code (if you're experienced with VBA, you'll notice there are some serious errors – that's the point! We'll fix them soon):

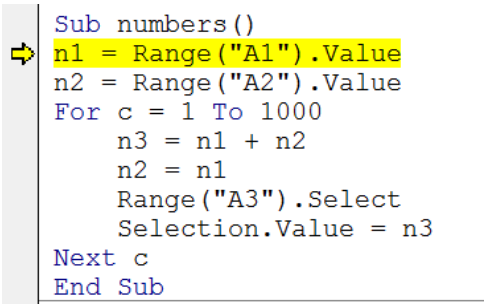
```
Sub numbers()
n1 = Range("A1").Value
n2 = Range("A2").Value
For c = 1 To 1000
    n3 = n1 + n2
    n2 = n1
    Range("A3").Select
    Selection.Value = n3
Next c
End Sub
```

Press [F5] to run the macro (you'll need to have the text cursor somewhere inside the macro code) and check the results. You'll notice it's not working correctly. A3 looks fine, but we were expecting values for A4 and on. We'll use our debugging arsenal to figure out why these cells are empty.

With the macro stopped, make sure your cursor is once again inside the macro somewhere. Now press [F8] instead of [F5] to run the macro one line at a time. With the first press of [F8], the first row gets highlighted in yellow. This is the next step that's about to be executed.

Press [F8] again, which highlights the next command. The macro is now paused. These two screenshots show how you step through with [F8]:

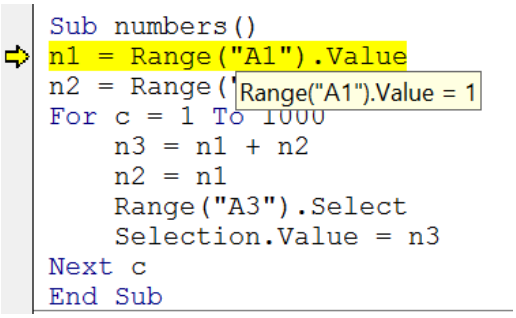


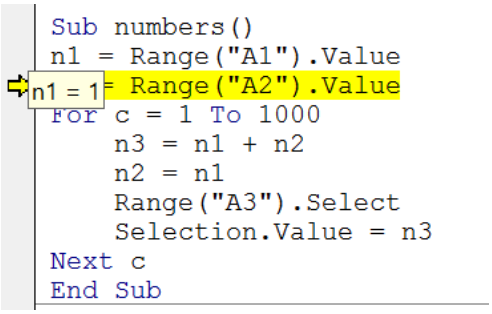


Let's pause at this highlighted line. We are about to assign the value of cell A1 to variable `n1`, but this command hasn't run yet. How do you know what the value of A1 is? You can look at the Excel sheet with [ALT]+[F11] and see that the value is 1. Alternatively, use your mouse to hover over `Range("A1").Value` and VBA will tell you its value! You can hover over any variables and range references in VBA while the macro is paused. Hover over `n1` and you'll see it says "Empty", because we haven't assigned it any value yet.

[52] The Fibonacci sequence starts with 1, 1, and then each subsequent number is the sum of the two previous numbers. So it goes 1, 1, 2, 3, 5, 8, 13, etc. It has some very interesting properties. Read about it on Wikipedia: http://en.wikipedia.org/wiki/Fibonacci_number

Press [F8] again to execute the highlighted step. Hover over `n1` again; it has the value of 1 now. These screenshots show how hovering works at each step:





Press [F8] again as you continue stepping through the code the same way the computer reads it. This process is a great way to understand exactly what the macro is doing. First, we assign values for `n1` and `n2`. Then comes a loop within VBA, called a `For` loop.

The `For` loop starts with the line `For c = 1 to 1000` and ends with `Next c`. Everything between these two lines gets executed over and over, as the variable `c` automatically iterates from 1 to 1000. The first time the loop runs `c` is 1, the next time `c` is 2, and so on.

Keep pressing [F8] to see how this happens. Looping takes a long time, so hold down [F8] to go faster! You'll see the yellow highlight go through the loop over and over. If you switch back to the Excel sheet, you'll notice that cell A3's value is always 2. Clearly we've got a couple of problems: the macro isn't stepping down to the next cell, and it isn't iterating the Fibonacci values properly. Let's fix it.

Inside the loop, instead of always selecting cell A3, we should select A3 first, then A4, then A5, and so on. We need to use the variable `c`, which changes each time the loop runs. Replace `Range("A3").Select` with:

```
Range("A" & c + 2).Select
```

Just like when we concatenated text inside Excel cells, we can now use variables and the & character to concatenate text in VBA. The above code is the same as `Range("A4")` when `c` has the value 2.

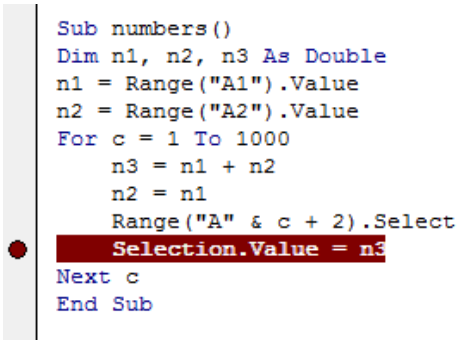
Run the macro again. Below, I'm highlighting the only row that has changed:

```
Sub numbers()
Dim n1, n2, n3 As Double
n1 = Range("A1").Value
n2 = Range("A2").Value
For c = 1 To 1000
    n3 = n1 + n2
    n2 = n1
    Range("A" & c + 2).Select
    Selection.Value = n3
Next c
End Sub
```

There's some progress now; at least the macro outputs 100 numbers. But all the resulting values are 2, so we still have some logic errors. It must be a problem with n1, n2, and n3 in the loop.

Use one more debugging tool to finish the job: [F9]. When your text cursor is inside the macro, press [F9]. It highlights the current row in red and puts a red circle in the margins. This is a break point for the macro (where we'll pause and take a break). As an alternative to going step by step with [F8], the macro can pause at these specific break points.

Put a break point at the last line before the end of the loop, `Selection.Value = n3`. We want to see what's going on just before the cell's value gets output. Get rid of any unwanted break points by pressing [F9] again. (If you prefer the mouse, click on the margin where the red dot appears. You can toggle break points on or off.)



```
Sub numbers()
Dim n1, n2, n3 As Double
n1 = Range("A1").Value
n2 = Range("A2").Value
For c = 1 To 1000
    n3 = n1 + n2
    n2 = n1
    Range("A" & c + 2).Select
    Selection.Value = n3
Next c
End Sub
```

Now press [F5] to start the macro. It runs until it reaches the break point, then it highlights that line in yellow (it's next to be executed) and pauses. Now we can look around and see what's going on. Hover your mouse over the values for n1, n2, and n3. They are all what they should be: the first two are 1s, and the third number is a 2.

Press [F5] again to continue the macro from the break point; you'll come to a break again at the same line. If you switch to the Excel sheet, you can see we've written a 2 in A3. What are the values of n1, n2, and n3 now? The same as before, unfortunately: n1 and n2 are 1s, and n3 is a 2. And the cell A4, which is now selected, is about to get assigned the value of 2. That's the wrong value.

If you keep going with [F5], you'll notice that n1, n2, and n3 never change. We're not iterating the sequence properly. Instead of this line:

```
n2 = n1
```

We should have these two lines:

```
n1 = n2
n2 = n3
```

This way we are properly setting up the next iteration. The current second number becomes the new first number. The current third number becomes the new second number. With these changes, the next loop will calculate the correct new third number.

Let's put it all together and run the macro! Keep the break point in place and step through with [F5] to verify that everything is working. Use [F9] again to remove the break point once the macro is working. This is the correct code:

```
Sub numbers()
Dim n1, n2, n3 As Double
n1 = Range("A1").Value
n2 = Range("A2").Value
For c = 1 To 1000
    n3 = n1 + n2
    n1 = n2
    n2 = n3
    Range("A" & c + 2).Select
    Selection.Value = n3
Next c
End Sub
```

By the way, Fibonacci numbers increase very rapidly. The last value (which is the 1002nd in the series) starts with a 1 and has 209 digits (the name for that number is one hundred octasexagintillion).

To recap, use the following tools for troubleshooting macros:

- [F5] to run a macro until the next break point (or until it finishes)
- [F8] to execute the next step of the macro
- [F9] to insert or remove break points in the code
- Hover over variable names in the code (when the macro is paused mid-execution) to reveal their value at that point in time

There are four more debugging techniques that you might find helpful:

- **Comment blocks.** We saw earlier that you can comment out a line of code by typing a single apostrophe character at the beginning of the line. The text turns green, and the computer knows not to execute that line of code. If you want to remove code, it's often better to comment it out rather than delete it, in case you want to add it back later!
  You can comment or un-comment several lines at once. In the VBA window, click View, Toolbars, Edit to turn on the edit toolbar. Find the two icons called Comment Block and Uncomment Block. Now select multiple lines of code and click Comment Block or Uncomment Block to turn comments on/off.
- **Status bar updates.** If you have a very long-running macro (you could have loops that run for minutes or hours), it's nice to know what's going on while the computer is busy executing code. You can output any custom text to the status bar at the bottom of the Excel window. One of the example exercises in this chapter will show you how.
- **Output values to the Excel sheet.** While you're building your macro, designate an area of your Excel sheet for macro outputs. In the Fibonacci example, you could choose to output n1 and n2 into columns B and C in each output row. This lets you keep track of each variable each time through the loop. You can comment out this extraneous code once the macro is working correctly.
- **Message boxes.** You can create pop-up messages at various points during your macro. These could either serve as very in-your-face status updates, or as decision points driven by the user. In the Fibonacci example, you might pop up a message after every iteration that contains the values of n1, n2, and n3. We will look at message boxes later. I avoid them because they remind me of Windows error messages, which nobody likes to see.

Golden Rules of VBA

We looked at the Golden Rules of Modeling in the previous chapter. I also have three Golden Rules of VBA that will make your life much easier (and make your code more likely to work):

Rule #1: Avoid repeating code

Rule #2: Use named references instead of cell references

Rule #3: Do your calculations on the Excel sheet, not in VBA

Rule #1: Avoid Repeating Code

Macros are great for running similar commands over and over. Take advantage of loops to avoid typing repetitive code. For example, avoid doing this:

```
Range("A1").Value = 1
Range("A2").Value = 2
```

```
Range("A3").Value = 3
Range("A4").Value = 4
Range("A5").Value = 5
```

This is repetitive code: it's boring to enter and a pain to change later. Instead, replace these with a single `For` loop:

```
For r = 1 To 5
    Range("A" & r).Value = r
Next r
```

If you want to keep going with this code 100 times, you just replace the 5 with 100. Or change the middle line if you want to enter a different value in these cells.

Even for more complex cases, if you find yourself repeating a block of code (perhaps with small modifications), there is usually a better way. Use loops or split off the repetitive code into a function that you can call multiple times. In my experience, more than ten lines of repetitive code will become painful to manage and are better handled with loops or functions.

Rule #2: Use Named References Instead of Cell References

Whenever possible, avoid using cell references in your macros like:

```
c = Range("B7").Values
```

The reason: if you move cell B7 anywhere else on the sheet, the macro will reference the wrong cell! Inserting or deleting rows or columns, or moving things around on the sheet, are likely to cause these references to break. Excel updates formula references on the worksheet automatically, so you normally don't have to worry about adding or deleting rows or columns. But **VBA does not update cell references**!

The solution: name the cell! Give B7 a named reference such as "main_input". Your macro code is now:

```
c = Range("main_input").Value
```

This way, the original cell B7 can move around, but the reference "main_input" will always point to the correct cell. Get in the habit of naming every cell you reference within VBA.

If you have to reference a range of cells that all belong together (such as the A1 through A5 example above), you could do the following: name the first cell, then use `Offset` in VBA to reference the others in relation to the first cell. The loop from Rule #1 should be replaced with the following, to avoid the hard-coded reference to column A and row 1:

```
For r = 1 To 5
    Range("start_cell").Offset(r - 1, 0).Value = r
Next r
```

Rule #3: Do your calculations on the Excel sheet, not in VBA code

I mentioned this point earlier regarding worksheet functions within VBA. You'll want to limit any calculations you do within VBA itself to very simple things: addition or subtraction, MIN or MAX, comparing greater than or less than.

The reason: calculations within VBA code are difficult to understand, and difficult to debug. You can clearly see the results of formulas within the Excel sheet, but you need to go through a tedious debugging process to see what VBA calculations are doing.

For instance, take this complicated VBA calculation:

```
d = Application.Max(a + b, 0) / Application.RoundUp(c + b, 0)
```

We should do this calculation on the Excel sheet. So let's push the values of a, b, c to named references on the sheet, and read back d as an output. Assign A1, B1, C1, D1 to have named references macro_a, macro_b, macro_c, macro_d. The new macro code is this:

```
Range("macro_a").Value = a
Range("macro_b").Value = b
Range("macro_c").Value = c
d = Range("macro_d").Value
```

The VBA part is now very simple and easy to understand. Where's the calculation itself? You need to type it into cell D1 (named macro_d), which gets read back into the macro. In Excel this formula is:

=MAX(macro_a + macro_b, 0) / ROUNDUP(macro_c + macro_b, 0)

While this formula may not look simpler than the VBA version, having it located on the Excel sheet makes it very transparent and easy to troubleshoot!

Example VBA Exercises

The rest of this chapter consists of several example exercises to hone your VBA skills. We'll cover some common basic tasks with VBA that should immediately pay dividends in your day-to-day work. Follow along with these examples in Excel to better understand how they work!

Copy Values from One Cell to Another

This basic task will be a part of almost every macro you see going forward. We'll take a value from one cell and copy it to another cell.

Here's how a recorded macro would do this task:

```
Sub Macro1()
Range("C5").Select
Selection.Copy
Range("C8").Select
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
    :=False, Transpose:=False
Application.CutCopyMode = False
End Sub
```

What does this code do? Let's read it line by line:

- Sub Macro1() just marks the start of this macro
- Select cell C5
- Copy the selection
- Select cell C8
- Now for a big long bit of code that says we are using Paste Special to paste Values. The other special modifiers such as transpose or skip blanks are off. These are all settings from the Paste Special dialog box, which you've seen before
- `Application.CutCopyMode = False` is the VBA way to turn off copying the selection. It's what happens after you've copied something and then press [ESC]
- End Sub ends the macro

The first major problem with this code is the violation of **VBA Rule #2: Use named references instead of cell references**. C5 and C8 may be the correct cells now, but if you insert rows or change anything on the Excel sheet, your macro code will break. So the first thing you'll want to do is replace C5 and C8 with named references such as source_cell and destination_cell.

With those changes, Macro1 above works correctly. But there's a much better and shorter way to copy values from one cell to another:

```
Sub Macro2()
Range("destination_cell").Value = Range("source_cell").Value
End Sub
```

That's it! A one-line macro which has significant advantages:

- It's shorter to type.
- It executes much faster. That may not matter if you do this one time, but if you have a macro that executes thousands of these commands, you'll save some serious time by avoiding copy-paste.
- By not using the clipboard (for copying and pasting), you can still do other copying and pasting on your computer while this macro is running. If this macro were part of a very long loop that ran for ten minutes, you would not be able to use your computer's copy-paste ability while your macro was running! No need to hijack your computer with inefficient code.

Build a Delete-Row Shortcut Macro: IF Statements and Shortcut Keys

Have you ever deleted a row you thought was empty, but you ended up deleting something important on your sheet? Wouldn't it be cool to have a delete command that first made sure the whole row was empty? Let's build such a command. We'll assign a keyboard shortcut so you can easily use it.

We need a way to see if a row is empty. These steps will work:

1. Start at the selected cell, and use [CTRL]+[Left] to go left to the end of the contiguous range
2. If you end up on a cell that is not empty, you have a non-empty row; don't delete the row
3. Otherwise, use [CTRL]+[Right] to go right to the end of the contiguous range
4. If you end up on a cell that is not empty, you have a non-empty row; don't delete the row
5. Otherwise, we can be certain that this row is empty; delete the row.

Now let's turn this logic into VBA code. As always, start by creating a module and opening up the code for that module. The full code is very short, so I will give you the whole thing here, and then we'll look at it line by line:

```
Sub delete_row()
Selection.End(xlToLeft).Select
If Selection.Value = "" Then
    Selection.End(xlToRight).Select
    If Selection.Value = "" Then
        Selection.End(xlToLeft).Select
        Selection.EntireRow.Delete
    End If
End If
End Sub
```

Now let's break this down:

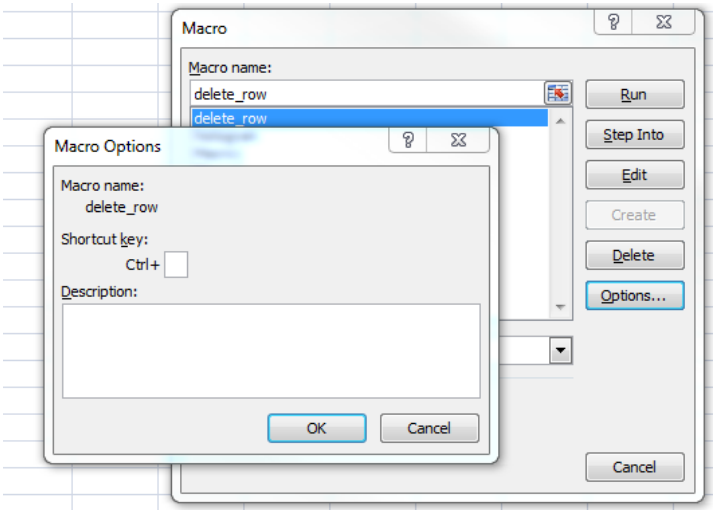
- `Selection.End(xlToLeft)`. Selection always means whatever cell or cells are currently selected. This command starts with the selected cell, and goes to the end of the contiguous range to the left (`xlToLeft` goes left, `xlToRight` goes right, `xlUp` goes up, `xlDown` goes down[53]). You need to add `.Select` to actually select the cell to the left.
- Now we have an `If` statement. We're evaluating whether the value of the new selection (`Selection.Value`) is equal to two double-quotes, which means it is an empty value. If it is empty,

[53] There is no such thing as `xlLeft` and `xlRight`; you must use `xlToLeft` and `xlToRight`. Weird, I know.

we execute the next step inside the `If`. If it isn't empty, we skip the `If` and move all the way down to the `End If` statement, just before the end of the macro.

- If the first `If` is true, we keep going. The next line is indented purely for ease of reading. I like to indent with Tab everything inside the `If` statements to show that this block of code belongs together. The next line selects a new cell to the right.
- Next is another `If` statement. The new selection is all the way to the right, but is it empty? If it's not empty, skip down to the next `End If`. Otherwise, continue inside the `If`.
- Next we tell Excel to go all the way back to the left. You need this so you don't end up at the very right side of the sheet when the macro is finished (in column XFD, who wants to be there?).
- Now, the most important part: delete the row. `Selection.EntireRow.Delete` means you take the Selection, expand it to the entire row, and delete.
- Finally, close each `If` statement with a corresponding `End If`. The inner `If` and `End If` go together first, then the outer ones go together (similar to sets of parentheses).

You may want to assign a keyboard shortcut so you can run this macro easily, and here's how you do that: Go to Developer, Macros. Find your macro called "delete_row" in the list, and click Options. This is the same window you see when you record a new macro.



Press the desired shortcut key, such as the letter d, or number 5, or anything you want. You can also hold down [SHIFT]. Pressing [SHIFT] + [D] means the keyboard shortcut for your macro will be [CTRL]+[SHIFT]+[D].

Be careful not to assign a commonly used keyboard shortcut to your macro, such as [CTRL]+[S] which is already assigned to Save. You really don't want these common shortcuts to launch your custom macro (unless you want to play a prank on someone)!

There's one more way we can improve this macro: let's show an error message when you're trying to delete a non-empty row. This way you can be certain the macro is running correctly, instead of just doing nothing with non-empty rows. The code for that is very simple:

```
MsgBox "Row is not empty!"
```

Of course, you can write anything you want within those quotes. We'll just need to execute this `MsgBox` statement when the `If` conditions are false. Use an `Else` statement: everything between the `Else` and `End If` will execute only if the `If` statement is false.

Here's the full macro with the new changes highlighted:

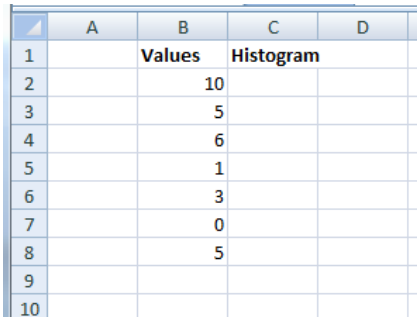
```
Sub delete_row()
Selection.End(xlToLeft).Select
If Selection.Value = "" Then
    Selection.End(xlToRight).Select
    If Selection.Value = "" Then
        Selection.End(xlToLeft).Select
        Selection.EntireRow.Delete
    Else
        MsgBox "Row is not empty!"
    End If
Else
    MsgBox "Row is not empty!"
End If
End Sub
```

Make a Histogram: Loops and Variables

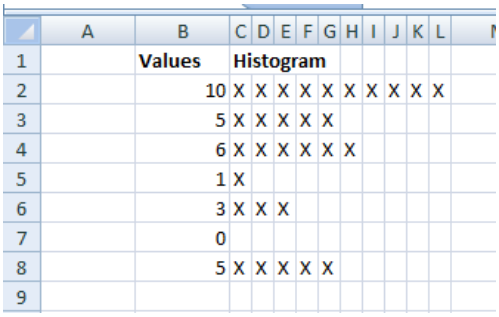
The following example illustrates the power of multiple iterative loops.

Suppose we have a sheet with some numbers in a column. We want a macro that makes a simple histogram by putting a certain number of X values to the right of these numbers. If I see a 5, I want five X values in the columns to the right. If I see a 10, I want ten X values.

Here's what the beginning setup and the end result look like:



	A	B	C	D
1		Values	Histogram	
2		10		
3		5		
4		6		
5		1		
6		3		
7		0		
8		5		
9				
10				



	A	B	C	D	E	F	G	H	I	J	K	L
1		Values	Histogram									
2		10	X	X	X	X	X	X	X	X	X	X
3		5	X	X	X	X	X					
4		6	X	X	X	X	X	X				
5		1	X									
6		3	X	X	X							
7		0										
8		5	X	X	X	X	X					
9												

Let's think logically about how this macro will work:

1. We'll go to the first value and see what it says (in this case 10)
2. Then we'll take the ten cells to the right of this value and fill them with X
3. Then move on to the next row, and repeat step 2
4. Stop if we get to a blank row (it has to stop at some point!)
5. Format columns C, D, E, and so on, so they are only as wide as an X character

First, a little prep work: your macro needs to know which cell to start in. And don't just say B2, because that violates VBA Rule #2! Instead, we need to name this cell something like first_value.

Go to the VBA editor with [ALT]+[F11] and add a module. Then type the following into VBA, which creates the initial structure for this macro:

```
Sub histogram()
Range("first_value").Select
v = Selection.Value
If v > 0 Then
    For c = 1 To v
    Next c
End If
End Sub
```

Looking at this line by line:

- Select the first value, which is 10 in this example. If you forgot to name a range first_value, VBA gives an error because it can't find first_value.
- `Selection` always refers to the selected cell. We put that cell's value (`Selection.Value`) into a variable called `v`. This variable will later determine how many X characters to output.
- Check if `v` is greater than 0. If it is greater, then we go on to the next line. If it's 0 (or negative) we won't do anything. Instead continue from the `End If` that is several lines below.
- Now comes the `For` loop. It repeats everything between the `For` and the `Next` part over and over, a specified number of times. We'll use it to output X values, `v` number of times. `For c = 1 To v` means you take a variable called `c`, set it equal to 1, and keep looping until it reaches the value of `v`. The line `Next c` increases the value of c by 1, and then loops back to the line with `For`.

So far, so good; next we'll put X values into several cells to the right of the selection. We need to put "X" in the cell that is one to the right of the selection, then two to the right, then three to the right, and so on, until we reach the limit set by `v`. We can do all this with just one line of code:

```
Selection.Offset(0, c).Value = "X"
```

This command takes the selected cell (the one named first_value) and then offsets it. `Offset` in VBA works similarly to the regular OFFSET worksheet function from Chapter 4: it starts with one cell, and then moves some number of rows down and some number of columns to the right. As usual for VBA, you add this on with a period. We want to stay in the same row, so the first input for `Offset` is 0. But we want to go right by `c` columns. How much is `c`? It's determined by the `For` loop we already set up: first 1, then 2, then 3, and so on until it reaches `v`.

Here's the main logic so far:

```
If v > 0 Then
    For c = 1 To v
        Selection.Offset(0, c).Value = "X"
    Next c
End If
```

If you've been following along, go ahead and run this with [F5]. You should get ten X's next to the 10.

The output would look much nicer if these columns were narrower, so let's auto-fit the columns using VBA. We can put this inside the `For` loop as well, such that every time we place an X, we make sure the column is the correct width.

Add this line inside the `For` loop:

```
Selection.Offset(0, c).Columns.AutoFit
```

You've seen the first part of this line: offset the selection by the correct number of columns. Then add `.Columns` to reference the column of that cell (you can't auto-fit a cell, but you can auto-fit a column). Finally, we add on `.AutoFit`.

The last thing we need to do is move on to the next row and repeat the same task of placing X's. And we want to keep doing this until we get to a cell that is empty. We'll use a different kind of loop for this: a `While` loop.

The code for just the loop looks like this:

```
Do While Selection.Value <> ""
Loop
```

It means we'll keep looping as long as ("while") the selected cell's value is not blank. The less-than and greater-than signs mean "not equal to", and the double quotes mean blank.

At this point, make a note for yourself that somewhere within the loop, the Selection needs to be changed to the next row; otherwise the `While` statement will always be true, and the macro will be stuck in an infinite loop!

Note: If you run into an infinitely looping VBA macro (or just a macro that keeps going too long), hit [ESC] a bunch of times to make it stop.

Here's how to shift the selection to the next row:

```
Selection.Offset(1, 0).Select
```

This line takes the selection, offsets it by 1 row and 0 columns, and selects that new cell. In other words, you move down one cell. Putting all the lines of code together, we get:

```
Sub histogram()
Range("first_value").Select
Do While Selection.Value <> ""
v = Selection.Value
If v > 0 Then
    For c = 1 To v
        Selection.Offset(0, c).Value = "X"
        Selection.Offset(0, c).Columns.AutoFit
    Next c
End If
Selection.Offset(1, 0).Select
Loop
End Sub
```

Go ahead and run this macro – it should work! The result should match the histogram screenshot from a few pages back. Are we done? Not quite!

The macro works well the first time, but there's nothing to clear out those Xs in case you change the values. If you change the first value to 5 and rerun the macro, you're still stuck with those same ten Xs. Let's incorporate some code at the beginning of the macro to clear everything out.

Once again, we need to be very specific on how to do this. You can't just tell Excel to "clear all the X values from my histogram". Excel isn't smart enough to do that. Instead, we need to break this down into simple steps:

1. Select the first value
2. Use [CTRL]+[SHIFT]+[Down] to select the entire column of values
3. Offset that selection to the right, and expand it to at least 20 columns (or more, if that's not enough)
4. Clear everything in those cells

The above method isn't perfect (what if you have more than 20 columns of X? What if the 20th column contains important data you shouldn't be deleting?), but it will work for this simple example. As you learn more about macros, you will probably figure out a better solution to these potential issues.

The code to accomplish the four steps above is this:

```
Range("first_value").Select
Range(Selection, Selection.End(xlDown)).Select
Selection.Offset(0, 1).Resize(, 20).Select
Selection.ClearContents
```

And the step by step analysis:

- You've seen the first line before – it selects the first value.
- The next line takes a range that is defined from the Selection to the end of the contiguous range down. It's the same as [CTRL]+[SHIFT]+[Down], and it's a good one to remember.

- The third line takes the selection (the column of values), offsets by 1 column to the right, then resizes the whole selection. Resize is similar to offset, but changes the row and column size of the selection. The first input for Resize is the number of rows; we don't want that to change, so leave it empty, followed by a comma, followed by 20. That resizes the selection to 20 columns wide.
- The last line clears out any values in the selected range. It's the same as pressing the [DEL] key.

And we are done! The final code for this macro is:

```
Sub histogram()
Range("first_value").Select
Range(Selection, Selection.End(xlDown)).Select
Selection.Offset(0, 1).Resize(, 20).Select
Selection.ClearContents

Range("first_value").Select
Do While Selection.Value <> ""
V = Selection.Value
If V > 0 Then
    For c = 1 To V
        Selection.Offset(0, c).Value = "X"
        Selection.Offset(0, c).Columns.AutoFit
Next c
End If
Selection.Offset(1, 0).Select
Loop
End Sub
```

Reference Sheets and Files in VBA

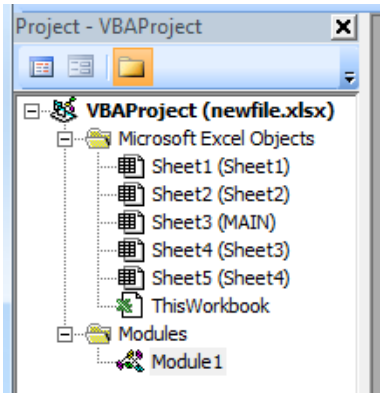
VBA gives you three ways to reference a particular sheet. The first two have potentially serious flaws, so the third way is my preference:

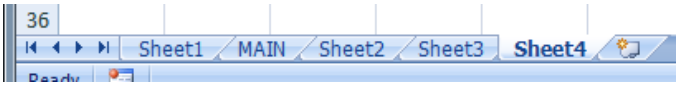
`Worksheets("new").Activate`
or
`Sheets("new").Activate`
	Activate the sheet that has the label "new". The downside: If there is no such sheet, you get an error. If you change your sheet label, your code breaks.

`Worksheets(2).Activate`
or
`Sheets(2).Activate`
	Activate the second sheet, in the order your sheets are arranged. The downside: if you rearrange your sheets, your code ends up referencing the wrong sheet!

`Sheet3.Activate`	Reference a specific sheet with the identifier of Sheet3. It doesn't mean this sheet's tab is labeled "Sheet3"; we're talking about the original Sheet3.

The third option uses the sheet's identifier instead of its label. When you rename a sheet in Excel, you're changing its label. Changing a sheet's identifier is done in VBA only. To explain sheet identifiers versus sheet labels, consider a file with this VBA tree and these tabs:





The first sheet was always labeled Sheet1, and that hasn't changed. But the original Sheet3 was renamed MAIN and moved before Sheet2. Then we added two new sheets at the end.

MAIN is the sheet's label, found in parentheses in the tree. The sheet's identifier is Sheet3, which is before the parentheses. You can select this sheet with `Sheet3.Activate` or `Sheets("MAIN").Activate`. I recommend the former; the identifier Sheet3 is a more permanent way to reference this sheet, and will continue to work even if you change the sheet's label later.[54]

Now let's incorporate references to different files. Try this:

```
ThisWorkbook.ActiveSheet.Range("B2").Select
```

As usual with VBA, we're stringing together code with periods.

- This line starts with an item called `ThisWorkbook`, which is always the Excel file that is running the macro.[55]
- Next is `ActiveSheet`, which refers to the sheet that is currently selected within the workbook.
- Then we reference a specific cell on this sheet with `Range("B2")` and select that cell.

Note that `ThisWorkbook` and `ActiveSheet` are optional and are used to clarify which workbook and which sheet you're referencing. You could also use the shorter version:

```
ActiveSheet.Range("B2").Select
```

Or the shortest version:

```
Range("B2").Select
```

To make your code reference a specific sheet in a specific file, include the workbook and the sheet reference. Here's how to reference B2 on a different sheet in a different workbook:

```
Workbooks("file.xlsx").Sheet3.Range("B2").Select
```

It's the same structure as before: specify the workbook, then the sheet, then the range.

---

[54] You can change a sheet's identifier as well, but it's rare to do so. Click on the sheet in VBA and look for the Properties box below the tree. The first property for this sheet is labeled "(Name)", and you can change it from Sheet3 to anything else. To confuse things further, there is another property called "Name" further down, which is really the label. I recommend you leave both of these properties alone and rename sheets the traditional way, using the tabs.

[55] With one minor exception: you can use macros to start running *other* macros in *other* workbooks! While another macro is running, `ThisWorkbook` becomes the workbook containing that other macro.

Here's how you switch to a different Excel file:

```
Workbooks("file.xlsx").Activate
```

If the file named "file.xlsx" is already open, this switches to (activates) that file. If such a file is not currently open, Excel returns the error "Subscript out of range." You'll need to open the file first.

To delete a sheet within the current workbook use `.Delete`:

```
ActiveWorkbook.Sheet1.Delete
```

`ActiveWorkbook` is the file that is currently active. It's not necessarily the same as `ThisWorkbook`, which is the file where your VBA code lives. You can be running a macro in oldfile.xlsx which activates the file newfile.xlsx (like we did above), then deletes a sheet from the `ActiveWorkbook`. In this case the active workbook is newfile.xlsx.

Here are some practical examples to copy data and formulas between sheets and files.

Note: all of the commands in the left column belong on a single line, they just didn't fit. To tell VBA to continue the same command onto the next line, put an underscore (_) character at the end of the first line.

`Workbooks(A).Sheet1.Range("A1").Value = _` `Workbooks(B).Sheet1.Range("A1").Value`	A and B are string variables representing file names. Take Sheet 1, cell A1's value from file B, and copy that value in Sheet 1, cell A1 in file A.
`Sheet2.Range("B:B").Formula = _` `Sheet1.Range("A:A").Formula`	Take the formulas in all cells of column A from Sheet1, and copy those formulas into Sheet2's column B.
`ActiveWorkbook.Sheet3.Copy _` `Before:=Workbooks("target.xlsx").Sheets(1)`	Take the entire Sheet3 from the active workbook and copy it into another workbook called target.xlsx, before the first sheet in that workbook.

Now for an interesting example – what do you think this macro does?

```
Sub newmacro()
For Each WB In Workbooks
    For Each WS In WB.Worksheets
        WS.Range("A1").Value = WB.Name & "-" & WS.Name
    Next WS
Next WB
End Sub
```

This is a convenient use of a new type of loop, the `For Each` loop (twice!). It loops through each item in a set of items. Let's read the macro step by step, line by line:

- The outer loop goes through each Workbook that is open (it could be one, or several), and stores the workbook's name as the variable named WB.
- Then, within each workbook, loop through each sheet and store it in the variable WS.
- For all of these workbooks and all of the sheets, we set the value of A1 to a text value that consists of the name of the workbook (WB.Name), followed by a hyphen (-), followed by the name of the worksheet (WS.Name).

Try this macro when you have several workbooks open!

Open, Close, and Save Files

To **open an Excel file** with VBA, use one of these commands:

```
Workbooks.Open "filename.xlsx"
Workbooks.Open "C:\Users\chris\Desktop\filename.xlsx"
```

As long as the target file is in the same folder as the file with the macro, you can skip the full path and just use the first way with filename plus extension (don't forget the distinction between .xlsx and .xlsm). Otherwise, you should use the full path.

Working with full file paths is tricky, so I recommend assigning the path of the current file to a variable. You can assign the full path to the current file to a variable:

```
filepath = ThisWorkbook.Path
```

To **close an Excel file** with VBA, use one of the many ways to reference workbooks, followed by `.Close`.

```
Workbooks("Newfile.xlsx").Close
```

I recommend referencing the name of the file when closing it. The reason: chances are you opened the file using VBA, and you used the filename to do that. You should save that file name to a variable, and then use that variable to close the file. Like this:

```
FileName = "filename.xlsx"
Workbooks.Open FileName

'<some lines of code go here>

Workbooks(FileName).Close
```

If you run the above code and close the file, you might get a popup asking if you want to save changes. This popup is helpful, but we should try to avoid manual interaction – let's have the macro take care of saving changes!

You can save the file just prior to closing it. To **save a file** with VBA:

```
ActiveWorkbook.Save
```

You could also save changes at the same time as closing the file:

```
ActiveWorkbook.Close SaveChanges:=True
```

Or you can close the file without saving changes:

```
ActiveWorkbook.Close SaveChanges:=False
```

And finally, one more option for avoiding the save changes popup. You can use VBA to **disable Excel alerts** (such as the one about saving changes), then close the workbook, then re-enable alerts. It goes like this:

```
Application.DisplayAlerts = False
ActiveWorkbook.Close
Application.DisplayAlerts = True
```

Disabling and enabling Excel alerts can be useful in other parts of your macros. If you ever want to keep Excel from asking questions, use this code above.

You can also Save As and specify a folder path, file name, and file format. Use the file format option to save a file as a Macro-Enabled Excel workbook or even as a CSV. Here's a simple example of saving the file in the current directory, under the name newfile.xlsx:

```
filepath = thisworkbook.path
```

```
newfilename = "newfile.xlsx"
ActiveWorkbook.SaveAs Filename:= filepath & "\" & newfilename
```

Note: you need a space before and after the & sign, otherwise Excel will complain.

If you want to save as a macro-enabled file, specify it with the `.SaveAs` command:

```
ActiveWorkbook.SaveAs Filename:= "newfile.xlsm", FileFormat:=52
```

Note: we didn't set a file path this time, just the file name. In this case, the file is saved in the active Excel file's folder. Notice that you need a comma after the filename and before the FileFormat.

The following are the most common file formats:

- file format 51 is a regular Excel sheet, which needs extension .xlsx (or .xls)
- file format 52 is a macro-enabled Excel sheet, which needs extension .xlsm (but it accepts .xls)
- file format 6 is a CSV (comma separated values) file, which needs extension .csv

If you get a run-time error pop up when running a macro with Save As, chances are the problem is with the file path. Use your debugging tools to verify that you're using a valid folder.

Finally, let's put everything in this section to use with a working macro. This macro will read the name of a file from a cell (without the file extension), open that file, save it as a macro-enabled workbook, and close it. You could use this code as part of a loop to convert several regular Excel files into macro-enabled files.

```
Sub save_as_macro()

ThisWorkbook.Activate 'not a bad idea to include this
fn = Range("input_file").Value
fn_open = fn & ".xlsx"
fn_save = fn & ".xlsm"

Workbooks.Open fn_open
Workbooks(fn_open).SaveAs Filename:=fn_save, FileFormat:=52
Workbooks(fn_open).Close

End Sub
```

There is nothing in this macro you haven't seen before.

- I added a comment after the first command, just as a note to myself. That line sets `ThisWorkbook` (the file running the macro) as the active workbook – it's an optional command, but it's a good habit to include it.[56]
- Next we read the name of a file from the sheet into the variable `fn`. Then we set `fn_open` as the .xlsx version of the file, and `fn_save` as the .xlsm version.
- Open the file `fn_open`. That file, with extension .xlsx, needs to exist in the same folder as the file that is running the macro. If there's no such file, you get an error and the macro ends. Otherwise, we move on to the next line.
- Save the newly opened file as a .xlsm (with the same name).
- Finally, we close the new file we just saved.

[56] If you have multiple workbooks open, you can easily lose track of the active workbook while you're in the VBA editor. You could run a macro in one file and have it end up affecting a different file. Adding `ThisWorkbook.Activate` to the beginning of the macro prevents this issue.

Note: the last two lines specifically named the workbook called fn_open when executing their commands. We could have also used ActiveWorkbook.SaveAs, which would work but might be more difficult to understand. The macro above makes it more clear which file each command refers to.

Copy Named References from One File to Another

If you've been working with named references long enough, you've probably run into this situation: you have a range of cells you want to copy from one file to another. The range includes some named references, but when you copy everything over, the cells don't keep their names! You would have to go back one at a time and recreate the named references in the new file. Is there a better way?

Fortunately, we can write a quick macro that goes through each named reference in one Excel file, and replicates those references in another Excel file.

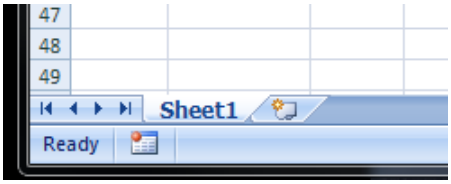
We'll use the `For Each` loop to go through each name on a sheet:

```
Sub copy_names()
Dim target_wb as String
target_wb = "newfile.xlsm"
For Each n In ActiveWorkbook.Names
    Workbooks(target_wb).Names.Add Name:=n.Name, RefersTo:=n.Value
Next n
End Sub
```

That's it! A quick and easy macro that can save a lot of manual effort. The key line here is the indented one, which adds a named reference to the target file.

Use the Status Bar

The status bar in Excel (and other Windows programs) is at the bottom left of your windows, and usually just says "Ready".



Did you know that you can insert custom text to replace "Ready", using VBA? To change what the status bar says, try this code:

```
Application.StatusBar = "Hello!"
```

This feature is useful with macros that run for a long time. You might have a macro that loops 10,000 times and runs for several minutes (or hours). While the macro is running, you have no idea how soon it will finish.[57] Let's write some macro code that displays which loop we are on and the percentage to completion:

```
Sub StatusCounter()
Dim loops As Integer
Dim c As Integer
Dim progress As Double
loops = 10000
For c = 1 To loops
```

[57] Later in this chapter we'll talk about how to speed up long-running macros. One trick is to turn off screen updating.

```
        progress = Application.Round(c / loops, 2) * 100
        Application.StatusBar = "Progress: " & c & " of " _
           & loops & " loops (" & progress & "%)"
    Next c
    End Sub
```

Analyzing this line by line:

- You've seen `Dim` before for declaring values, and the `For` loop for looping a specific number of times.
- The line with `Application.Round` just takes `c` divided by the total number of `loops`, and rounds it to two significant digits (for a nicely formatted percentage number). Then we multiply by 100 to display values like 50 instead of 0.5.
- The status bar text didn't fit on one line for me, so I used the underscore character to continue the same command on the next line. You could skip the underscore and put that command all on one line.

Send Emails: Outlook Module

Macros can send automated emails using special commands that interface with Outlook (for these to work, you need to have your email account set up within Microsoft Outlook). In this example, we'll email the current Excel file to a list of people.

The below code is a template for sending emails with VBA. It creates an object called an Outlook Application, which lets Excel generate emails for Outlook to send:

```
Sub email()
Dim olapp As Object, olmail As Object
Set olapp = CreateObject("Outlook.Application")
Set olmail = olapp.createitem(olmailitem)
With olmail
    .To = "to@to.com"
    .cc = ""
    .bcc = ""
    .Subject = "Subject of Email"
    .Body = "Body of email"
End With
'olmail.Attachments.Add ActiveWorkbook.FullName 'attach the active workbook
'olmail.Attachments.Add ("C:\Folder\file.pdf") 'attach a file
olmail.Send
End Sub
```

There are two commented lines (starting with the apostrophe) which let you add attachments in two different ways; uncomment those lines (remove the apostrophe) if you want to use them.

The code segment `With [. . .] End With` is a convenient way to specify multiple properties of an object, in this case the object `olmail`. You could replace that entire section with the following (it's just more repetitive):

```
olmail.To = "to@to.com"
olmail.cc = ""
olmail.bcc = ""
olmail.Subject = "Subject of Email"
olmail.Body = "Body of email"
```

Next we'll customize this template code to send the current Excel file to a list of email addresses, which we get from the range named "emails".

Start by filling out the basics: the subject line and the body text of the email (`olmail.Subject` and `olmail.Body`).

The “To” part is trickier because we want to build a list of email addresses. Fortunately, there is a `For Each` loop for that! We’ll build a long string called email_to that contains each email address separated by semicolons:

```
Dim c as Range
email_to = ""
For Each c In Range("emails")
    email_to = email_to & c.Value & "; "
Next c
```

Here’s how this code segment works:

- First we declare `c` as a `Range`. A `Range` is another type of VBA variable (we’ve seen numbers, strings, and objects). A `Range` variable stores cell references from the Excel sheet.
- Initialize a string variable called `email_to` and give it blank value (using double quotes). Like the `Dim` statement, this line is not strictly required, and your macro will work without it. But it’s good coding practice to set things up nicely.
- The `For Each` loop takes the range called “emails” (don’t forget the quotes, and make sure it’s spelled right!) and loops through each cell, storing the cell in variable `c`.
- Inside the loop, we want to tack on the new `c.Value` to the end of `email_to` and also add a semicolon separator.

When the loop finishes, you’ll end up with `email_to` containing a value something like: person1@company1.com; person2@company2.com; person3@company3.com;. It’s OK to have the extra semicolon at the end.

Finally, modify the command `olmail.Attachments.Add` to attach the current workbook. We need a file path and file name; both are obtained with `ActiveWorkbook.FullName`. The final email macro looks like this:

```
Sub email()
Dim c As Range
email_to = ""
For Each c In Range("emails")
    email_to = email_to & c.Value & "; "
Next c

Dim olapp, olmail As Object
Set olapp = CreateObject("Outlook.Application")
Set olmail = olapp.createitem(olmailitem)
With olmail
    .To = email_to
    .Subject = "Report on many important things"
    .Body = "Good morning, please open this report on many important things."
End With
olmail.Attachments.Add ThisWorkbook.FullName
olmail.Send
End Sub
```

You may run into some errors when running this email module. If one of the email addresses is not formatted properly, you’ll get this error:

“Outlook does not recognize one or more names.”

This should be easy to fix by looking at your list of email addresses. If necessary, copy the list of emails into a new email within Outlook and try sending it manually. Outlook should tell you what’s wrong.

You might also get Excel complaining about “missing references”. In this case, go to the VBA editor, and click Tools, References. You’ll see a list of Available References with check boxes next to them. Uncheck anything that says “Missing” and click OK. That should fix the missing references error.

Manipulate Pivot Tables with VBA

In this section, we’ll automate these common pivot table tasks using VBA:

- Refresh all pivot tables in your workbook
- Manipulate filters
- Change the data source
- Keep pivot table filters in sync

Refreshing pivot tables is easy. As you know, pivot tables don’t update when you change something in the underlying data. Fortunately, there’s a single line of VBA that refreshes all pivot tables and all data sources:

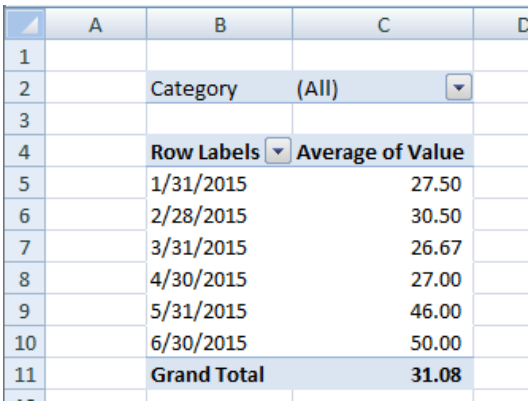
```
ThisWorkbook.RefreshAll
```

If you have just pivot tables and no external data connections (database connections that pull external data into Excel), this line of code is sufficient. If you’re having difficulty getting this to work properly (either because of external data, or because your Excel version is older than 2007), you might need this alternative, which specifically refreshes each pivot table with a loop:

```
Sub refresh_pivots()
Dim ws As Worksheet
Dim pt As PivotTable
For Each ws In ThisWorkbook.Worksheets
    For Each pt In ws.PivotTables
        pt.RefreshTable
    Next
Next
End Sub
```

We’re using `For Each` loops. The outer loop goes through each worksheet, and the inner loop goes through each pivot table on a given sheet. `pt.RefreshTable` takes a specific pivot table on a specific sheet and tells it to refresh.

Next, **pivot table filters**. To start you’ll need a pivot table, and for that you’ll need some data. Hopefully you have some data of your own to work with, but you can always download my example data set from www.AdvancedExcelBook.com. You’ll need a pivot table that has some kind of filter; the one from the website looks like this screenshot, with Category being filtered:



	A	B	C	D
1				
2		Category	(All)	
3				
4		Row Labels	Average of Value	
5		1/31/2015	27.50	
6		2/28/2015	30.50	
7		3/31/2015	26.67	
8		4/30/2015	27.00	
9		5/31/2015	46.00	
10		6/30/2015	50.00	
11		Grand Total	31.08	

You can use the following commands to change the Category filter. Notice that each line requires you to specify a pivot table; in this case it's the first (and only) one on the sheet.

To filter for just Category A, use this:

```
PivotTables(1).PivotFields("Category").CurrentPage = "A"
```

Use either of these commands to reset the Category filter to All:

```
PivotTables(1).PivotFields("Category").CurrentPage = "(All)"
PivotTables(1).PivotFields("Category").ClearAllFilters
```

To enable or disable the ability to select multiple items, use one of these commands:

```
PivotTables(1).PivotFields("Category").EnableMultiplePageItems = True
PivotTables(1).PivotFields("Category").EnableMultiplePageItems = False
```

With `EnableMultiplePageItems` set to True, you can toggle a single value such as Category B to be visible (selected) or not visible (unselected):

```
PivotTables(1).PivotFields("Category").PivotItems("A").Visible = False
PivotTables(1).PivotFields("Category").PivotItems("A").Visible = True
```

The general hierarchy for pivot table items is the following:

Sheet → PivotTable → PivotField → PivotItem

In the above examples:

- the Sheet is called Sheet1 (although it isn't referenced), which contains
- the PivotTable called PivotTable1, which has
- PivotFields that include the Category field, which has
- PivotItems, which are possible values of the Category field.

Next, let's **change the data source** of a pivot table. Remember from Chapter 6 that pivot tables require a data source, which you can manually change under the Options ribbon. To do the same thing with VBA, use this complicated line of code:

```
ActiveSheet.PivotTables(pt).ChangePivotCache ThisWorkbook.PivotCaches.Create _
(SourceType:=xlDatabase, SourceData:=rng)
```

Notice the underscore, which means this command continues on the second line (it's too long for me to fit on one line). The highlighted `pt` and `rng` are VBA variables that you need to set up before setting the data source.

- `pt` is either the name of the pivot table (a string variable) or a pivot table index number (the number 1 if it's the first pivot on the sheet)
- `rng` is a range variable that defines the pivot source range

To define `rng`, one of the following examples will work:

```
Set rng = Range("A2:C7")
Set rng = Range("my_named_range")
Set rng = Range(Selection, Selection.End(xlDown))
```

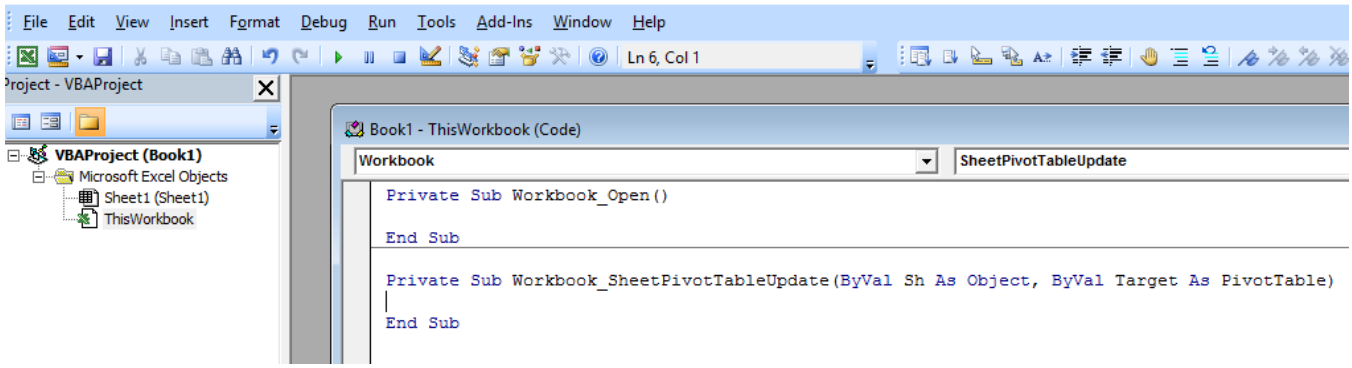
Notice that when you assign a value to a range, you need to begin the command with `Set`*. It's a special requirement for ranges, and it doesn't apply to other types of variables.*

That's it! You could use this code in a loop to quickly update the data sources behind all pivot tables in the entire file.

Finally, I'll show you how to **keep pivot filters synced** across multiple pivot tables. The idea is this: you have two pivot tables referencing the same data, and they both have filters. You change the filter for the first pivot to Category A, and you want the other pivot to switch to the same filter. Don't do this by hand; there's VBA for that! In fact, we'll make it automatic, so the pivot tables are always in sync without having to run the macro manually.

We'll start a little differently: instead of creating a new module to store this macro, we will assign it to a special event in the workbook. Double-click ThisWorkbook in the project tree in VBA, which opens up a code window for ThisWorkbook.

Notice there are two dropdowns at the top of the code window. Click the left dropdown that says (General) and switch it to Workbook. The other dropdown now says Open, and some code was automatically added below. Switch the second dropdown to SheetPivotTableUpdate. You'll end up with something like this:



You're looking at special sections of code that execute when certain events happen. `Private Sub Workbook_Open` will run immediately after you open the workbook. `Private Sub Workbook_SheetPivotTableUpdate` (with all of the things inside the parentheses after the name) will run whenever you update a pivot table – for instance, by changing its filters. This is what we want.

Add the code below to `Workbook_SheetPivotTableUpdate`. It's a lot to type, so head to www.AdvancedExcelBook.com and download the Macros and Pivot Tables file to copy-paste this code:

```
Private Sub Workbook_SheetPivotTableUpdate(ByVal Sh As Object, ByVal Target As
PivotTable)
Dim ws As Worksheet
Dim pt As PivotTable
Dim pfTarget As PivotField
```

```
Dim pf As PivotField
Dim pi As PivotItem
Dim MultItems As Boolean
On Error Resume Next
Application.EnableEvents = False
Application.ScreenUpdating = False
For Each pfTarget In Target.PageFields
    MultItems = pfTarget.EnableMultiplePageItems
    For Each ws In ThisWorkbook.Worksheets
        For Each pt In ws.PivotTables
            If ws.Name & "_" & pt <> ActiveSheet.Name & "_" & Target Then
                pt.ManualUpdate = True
                Set pf = pt.PivotFields(pfTarget.Name)
                MultItems = pfTarget.EnableMultiplePageItems
                pf.ClearAllFilters
                If MultItems = False Then
                    pf.CurrentPage = pfTarget.CurrentPage.Value
                Else
                    pf.CurrentPage = "(All)"
                    For Each pi In pfTarget.PivotItems
                        pf.PivotItems(pi.Name).Visible = pi.Visible
                    Next pi
                    pf.EnableMultiplePageItems = MultItems
                End If
                pt.ManualUpdate = False
            End If
        Next pt
    Next ws
Next pfTarget
Application.EnableEvents = True
Application.ScreenUpdating = True
End Sub
```

Note: the first two lines here (before the first Dim) should be all on one line in VBA, the way it was auto-generated. It just doesn't fit in the width of a page.

I won't go into detail on how this code works, because it's a bit tedious. You'll recognize commands from before such as `For Each` loops, and setting the Visible property of various fields. You can test it out by changing the filters on one of the pivots; the other one should update as well.

Run a Macro when you Change a Cell

In the previous section, you saw how special Excel events can trigger a macro. Next we'll look at running a macro automatically when you change things on a sheet.

This code will go into Sheet1 in the project tree, so double-click the sheet in VBA to open up its code window. Once again, we have two dropdowns. Choose Worksheet on the left, and Change on the right. This generates a macro called `Worksheet_Change`, which will automatically run every time you change a cell on this sheet (but not if you change other sheets).

Let's try a very simple example. Add this code to the Worksheet_Change macro:

```
Private Sub Worksheet_Change(ByVal Target As Range)
Target.Offset(1, 0).Value = Target.Value * 2
End Sub
```

Can you guess what this one line of code does?

Let's read the middle line to make sense of it: take the `Target` cell, then offset it by 1 row down and 0 columns, and make the value of that cell equal to the value of the target cell, times 2. `Target` is the cell that was actually changed, which is what triggered this macro in the first place.

In other words: whatever cell you changed, put double its value in the cell below.

Try this out by entering 1 into any cell on this sheet. Something strange happens! Instead of just adding a 2 in the cell below, the macro keeps going and adds 4, 8, 16, and so on down for a long time.

Note: if you run into errors with this code, especially a Type Mismatch error, click End and try again by entering a number in a cell. Chances are you entered text or changed multiple cells at once. If you clicked Debug and can't get the macro to run again, make sure you press the stop button in VBA.

Why is the macro entering so many values? Because we told it to change another cell (below the one you changed), and that action itself triggers another `Worksheet_Change` event. The macro is triggering itself (eventually it does stop)! Fortunately, you can stop this behavior by turning off "events" while this macro is running. Just add two new lines of code:

```
Private Sub Worksheet_Change(ByVal Target As Range)
Application.EnableEvents = False
Target.Offset(1, 0).Value = Target.Value * 2
Application.EnableEvents = True
End Sub
```

Turn off events so the macro cannot trigger itself, then turn events back on at the end of the macro.

Note: if you don't turn events back on, the auto-triggering events will completely stop working! To fix it, you have to create a brand new macro that turns events back on. Run that macro and everything will be back to normal. That macro can be very simple, like this:

```
Sub reset()
Application.EnableEvents = True
End Sub
```

Let's make one more quick change: if the Target range is larger than one cell, the code above gives an error and breaks. Allowing this macro to break leaves events disabled (we never got a chance to turn events back on at the end), which is a pain to reset, as you've just seen.[58]

The solution: resize the `Target` range to be a single cell, assigned to a new range variable called `Target2`. When assigning ranges, remember that you must write `Set` at the beginning of the line:

```
Set Target2 = Target.Resize(1, 1)
```

Here's the updated code:

```
Private Sub Worksheet_Change(ByVal Target As Range)
Application.EnableEvents = False
Set Target2 = Target.Resize(1, 1)
Target2.Offset(1, 0).Value = Target2.Value * 2
Application.EnableEvents = True
End Sub
```

Finally, let's look at how to execute code only when you change a specific cell or a range of cells. We're still using the `Worksheet_Change` macro, but it should only run if column A was changed. If you change anything

[58] If you're curious, you can break the current macro by selecting a multi-cell range (that is not blank) and pressing [DEL]. Probably not worth the effort, though.

else on the sheet, then nothing should happen. To do this, we'll add this line of code to the beginning of the macro to check whether `Target` is within the desired range:

```
If Intersect(Target, Range("A:A")) Is Nothing Then Exit Sub
```

You've seen `If` statements before, but this one is a little different. The expression `Intersect() Is Nothing` is a special way to look for overlap (intersection) between ranges. If the intersection between the `Target` range and column A is nothing (which happens if we didn't change anything in column A), then the If statement is True and we execute `Exit Sub`. That terminates the macro immediately and the rest of the lines don't run. Otherwise, if there is overlap between these ranges, we continue on to the next line of the macro.

Note: this `If` statement does not have a corresponding `End If`. You can write short If statements like this on a single line of code, as long as you're only doing one thing after `Then`. It's a little more compact this way.

Add Hyperlinks

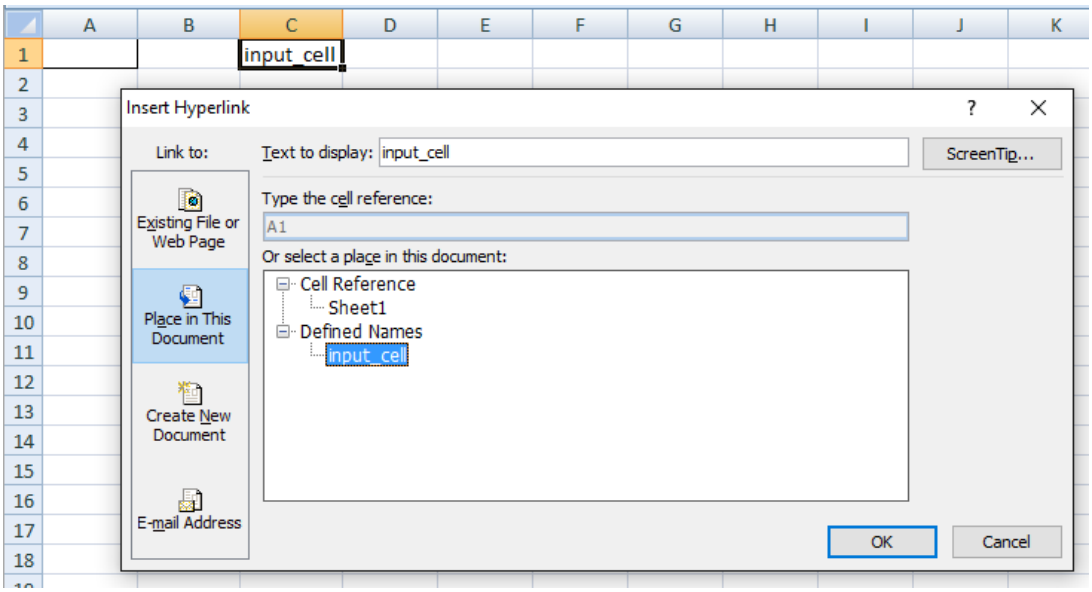
In this example, we'll use a macro to create links to named references within an Excel file. You can add hyperlinks in Excel by right-clicking a cell with some text in it and choosing Hyperlink. If you need to add more than a few links, the macro can be a nice time-saver.

This example shows how to use Record Macro to your advantage. There's no reason to memorize the VBA code for adding a hyperlink, since it's fairly obscure. Just record yourself adding a hyperlink, then modify the code in VBA to suit your needs. **Everything you do in Excel has a VBA equivalent – record a macro to see the underlying VBA code.**

Start by giving cell A1 the named reference input_cell. Then let's type the text "input_cell" into C1. We want C1 to contain a hyperlink to its corresponding named reference, which is in A1.

Start recording a macro and give it a shortcut (I like [CTRL]+[SHIFT]+[H] for hyperlink). While the macro is recording:

1. Right-click on the text in C1 and click Hyperlink
2. In the popup window, switch over to Place in This Document on the left sidebar and look for the list of Defined Names in the middle
3. Choose input_cell from the list and click OK or press [ENTER]
4. The hyperlink is generated and you can stop the recording now



Switch over to VBA with [ALT] + [F11] to see your newly created macro:

```
Sub hyperlink()
'
' Hyperlink Macro
'
' Keyboard Shortcut: Ctrl+Shift+H
'
   ActiveSheet.Hyperlinks.Add Anchor:=Selection, Address:="", SubAddress:= _
        "input_cell", TextToDisplay:="input_cell"
End Sub
```

The first few lines are comments; you can delete them or just leave them there. The actual code is just one command, separated into two lines with an underscore. It gives us the template code for adding a hyperlink! The `SubAddress` is the named reference we want to link to, and the `TextToDisplay` is the text of the link that is displayed on the sheet.

Let's modify this macro to read the cell's value and point to the corresponding named reference:

```
Sub Hyperlink()
    link = Selection.Value
    ActiveSheet.Hyperlinks.Add Anchor:=Selection, Address:="", SubAddress:= _
        link, TextToDisplay:=link
End Sub
```

Notice the use of the variable `link`, which we assign to two things: the address and the text to display. This code macro is flexible enough to create hyperlinks based on the value of any cell. Test it out by defining a new named range and typing the name of the range into another cell. Call the macro from the second cell using [CTRL] + [SHIFT] + [H].

Message Boxes

You can use a simple command to pop up messages while your code is executing. You could use message boxes to very clearly announce that something has happened, or perhaps as a debugging tool. Here's an example message box:

```
MsgBox "Row deleted. Press OK to continue."
```

You can use message boxes to ask Yes/No questions. For example, you could add a message box to the beginning of a macro to ask "Are you sure?". Since there is no undo after running a macro, this popup helps you to avoid running unintended macros. This code asks a question and then terminates the macro (`Exit Sub`) if the answer is no:

```
If MsgBox("Are you sure?", vbYesNo) = vbNo Then Exit Sub
```

Practice: Delete Empty Rows Macro

Here's an exercise to put your new macro skills to the test: take a dataset, and write a macro that deletes all empty rows from it. The file for this is available at www.AdvancedExcelBook.com; you can also just set up something that looks like this screenshot:

	A	B	C	D	E	F
1		**Build a macro that deletes all blank rows in the data**				
2		Note: use only VBA code to make changes on the sheet				
3						
4		Row ID	VALUE	TEXT		
5		1	1000	Some data for row 1		
6						
7		2	2000	Some data for row 2		
8		3	3000	Some data for row 3		
9		4	4000	Some data for row 4		
10						
11		6	6000	Some data for row 6		

So in the above screenshot, the macro would delete rows 6 and 10. There are multiple correct solutions to this exercise; use the techniques from this chapter to plan out your approach, and then write your own code.

Hint: you could use a loop that steps through values of column B, and stops if it sees an empty cell. At that point, you will still want to check if the entire row is empty; if so, then delete it.

Custom Functions within Excel

Functions are similar to macros: they use the VBA language, and you write their code in modules with the VBA editor. There's an important difference, though: you can use your custom functions[59] inside Excel formulas!

As a simple example, type this function (notice it's not a Sub) into a new VBA module:

```
Function triple(v As Double)
triple = v * 3
End Function
```

Before we interpret this code, let's try it out. Running this function like a macro, with the play button or [F5], doesn't do anything. Instead, you will be calling this function from the Excel sheet. Try putting a 1 in cell A, and then write this function in cell A2:

```
=TRIPLE(A1)
```

Excel recognizes your custom function and returns 3 in A2.

Here's how the function works:

- The first line declares a Function and gives it a name. In the same line, we need to define the inputs for the function. In this case, we want one input; we'll call it variable `v` and give it data type `Double`. Remember that the Double data type means we can accept decimals; it has nothing to do with doubling things.
- The second line takes `v`, multiplies it by 3, and assigns the result to the variable `triple`. This is how you specify what value the function will return: set the name of the function (`triple`) equal to whatever it should give as the output.

Functions can take one or more inputs, and always return a single output. With this light introduction out of the way, let's build some functions that are actually useful.

[59] Custom functions are also known as User Defined Functions, or UDFs, in case you want to search for them online.

Note: functions must go inside modules. With macros, we could add code to the worksheet with the Worksheet_Change event; that does not work with functions.

Compounded Annual Growth Rate (CAGR) Function

You'll run into the concept of Compounded Annual Growth Rate frequently in financial modeling. It's also known as a geometric mean. The idea is this: you have a quantity that changes from time period to time period. A company's revenues might grow by 2% during one year, then by 5%, then 4%, then 5% again. What is the average annual rate of growth for revenues? In other words, if revenues grew at the same rate over these four years, what rate would result in the same ending value?

If year 0 revenues are $100, then year 4 revenues are: 100 * (1+2%) * (1+5%) * (1+4%) * (1+5%) = $116.95. To get the compounded annual growth rate, we need to solve the following equation for R:

$$100 \times (1 + R)^4 = 116.95$$

Rearrange to get the following:

$$R = \left(\frac{116.95}{100}\right)^{1/4} - 1$$

Solve to get R = 0.039927, which is 3.99%. It's a pretty simple math formula, but why worry about remembering it? Let's just build a CAGR function in Excel that reads the inputs and returns CAGR:

```
Function CAGR(begin_val As Double, end_val As Double, periods As Double)
   CAGR = (end_val / begin_val) ^ (1 / periods) - 1
End Function
```

- The function takes three inputs, which must be in the correct order: beginning value (100), ending value (116.95), and number of periods (4).
- The next line is the math calculation, which is the same as the equation above. The result is assigned to the variable `CAGR` to make it the function output.

To test this out in Excel, type =CAGR(100, 116.95, 4) into a new cell. You should get the same result, 0.039927.

We can improve this function by adding some error handling. If the first input is 0, you get a #VALUE! error. It would be nice if we got #DIV/0! instead, to correctly indicate a division by zero error. Here's how to do that:

```
Function CAGR(begin_val As Double, end_val As Double, periods As Double)
If begin_val = 0 Or periods = 0 Then
    CAGR = CVErr(xlErrDiv0)
Else
    CAGR = (end_val / begin_val) ^ (1 / periods) - 1
End If
End Function
```

All that's changed is that we first check whether the beginning value or the number of periods is 0. If so, we use a special function `CVErr` to return an error code – in this case, the division by zero error. Other possible `CVErr` codes are:

```
xlErrDiv0
xlErrNum
xlErrValue
xlErrNA
xlErrName
xlErrRef
```

Notice that we have two different lines that assign a value to `CAGR`. That's allowed. The first line is called if the inputs are zero, and it returns an error code. Otherwise, the second line is called, and it returns a calculated result.

Concatenate Range Function

Remember the CONCATENATE function? It lets you string together text values one after another. To combine the text in cells A1, A2, A3, A4, A5, you'll use: `=CONCATENATE(A1,A2,A3,A4,A5)`

Unfortunately, `=CONCATENATE(A1:A5)`, doesn't work; you can't pass ranges to that function. Instead, you can build your own Concatenate function that does accept ranges:

```
Function CONCATRANGE(rng As Range)
Dim output As String
For Each c In rng
    output = output & c.Value
Next c
CONCATRANGE = output
End Function
```

- This function takes one input, which is the range in question. It goes into the variable `rng`.
- We declare a string called `output` that is initially empty. It will store all the individual strings as they are being concatenated.
- The `For Each` loop goes through each cell within the range, and assigns the cell to variable `c`.
- Add on the value of cell `c` to the value of the `output` string. This is done over and over while the loop is running.
- Finally, assign the value of `output` to the name of the function to make that the function output.

Call the function from Excel like this: `=CONCATRANGE(A1:A5)`

There's one more way we could enhance this function: what if we wanted a custom separator between each value, such as a comma? The function should take an optional second input and use it as a separator. Here's the modified code:

```
Function CONCATRANGE(rng As Range, Optional separator As String)
Dim output As String
For Each c In rng
    If output = "" Then
        output = output & c.Value
    Else
        output = output & separator & c.Value
    End If
Next c
CONCATRANGE = output
End Function
```

Take a look at the first line: we have the word `Optional` before the second input variable. Without that, CONCATRANGE would cause an error any time the second input was missing.

Why do we need the IF inside the loop? If the result string is empty, there's no reason to add a separator; you don't want to start your output with an extra comma or something. Only add on the separator if the output already has some value.

Try out your new function like this: `=CONCATRANGE(A1:A5,",")`

Middle to End of Text Function

Excel has built-in functions for grabbing the beginning, end, or middle of some text: LEFT, RIGHT, and MID. The problem with the RIGHT function is that you need to specify the number of characters from the end. There isn't a function to take everything that comes after a specific character number (like the opposite of the LEFT function). I don't have a function to grab everything after character number 4 in the text "Hi there", which should be "here".

We could accomplish that with this Excel formula, where A1 contains the text "Hi there":

```
=RIGHT(A1, LEN(A1)-4)
```

But it's fun to do this with a custom function. Here's the code we'll use:

```
Function mid_to_right(txt As String, num_skip As Integer)
num_keep = Len(txt) - num_skip
mid_to_right = Right(txt, num_keep)
End Function
```

- The inputs are the string itself and the number of characters to skip from the left
- The `num_keep` calculation takes the length of the text and subtracts the number of characters to skip
- Then we output a `num_keep` number of characters from the right; `Right` is a valid function within VBA

To call the function in Excel, use: `=mid_to_right(A1, 4)`

Sign Functions and Filtering Positive Values

Excel already has the built-in function SIGN, which returns either 1, -1, or 0 based on the sign of the input number. So `SIGN(-10)` returns -1, SIGN(1) returns 1, and sign (0) returns 0.

Let's build a similar custom function, one that returns the number itself for positive numbers and 0 for everything else. Call it a POSITIVE_FILTER function. Here's what it would look like:

```
Function positive_filter(n As Integer)
If n > 0 Then
   positive_filter = n
Else
   positive_filter=0
End if
End Function
```

Change PivotTable Filters with a Function

This last example is a function you don't call from the spreadsheet, but from inside a macro. It simplifies the task of changing pivot table filters with VBA, by selecting multiple filters.

Normally, you would use code like this to set pivot filters to category A and B:

```
pt = "PivotTable1"
cat = "Category"
Activeheet.PivotTables(pt).PivotFields(cat).ClearAllFilters
Activeheet.PivotTables(pt).PivotFields(cat).PivotItems("A").Visible = True
Activeheet.PivotTables(pt).PivotFields(cat).PivotItems("B").Visible = True
Activeheet.PivotTables(pt).PivotFields(cat).PivotItems("C").Visible = False
```

It's a bit clunky to turn each category on and off like this. It's also pretty slow to execute if you have a large data set, because the pivot table refreshes each time you make a change. Instead, you can call another function from inside a macro. So one of the lines within your macro would look like this:

```
Call ChgPivotFilter("PivotTable1", "Category", "A/B")
```

This line calls a function, and that function looks like this:

```
Function ChgPivotFilter(Pivot As String, Category As String, FieldNames As String)
    Dim itms As PivotItem
    Dim intIndex As Integer
    Dim eachfield As Variant
    Dim selected As Boolean
    On Error GoTo EH

    eachfield = Split(FieldNames, "/")
    ActiveSheet.PivotTables(Pivot).ManualUpdate = True
    ActiveSheet.PivotTables(Pivot).PivotFields(Category).ClearAllFilters
    If FieldNames <> "(All)" And FieldNames <> "" Then
        For Each itms In _
        ActiveSheet.PivotTables(Pivot).PivotFields(Category).PivotItems
            selected = False
            For intIndex = LBound(eachfield) To UBound(eachfield)
                If eachfield(intIndex) = itms.Name Then
                    selected = True
                End If
            Next
            itms.Visible = selected
        Next itms
    End If
    ActiveSheet.PivotTables(Pivot).ManualUpdate = False
    ActiveSheet.PivotTables(Pivot).Update
EH:
    Resume Next
End Function
```

This code is pretty long, but you've seen most of the building blocks before. There's a familiar `For Each` loop, as well as some logic to change the `selected` property of each pivot item. Go slow and type it in line by line; it's good VBA practice!

The following are all valid ways to use this function inside another macro:

```
Call ChgPivotFilter("PivotTable1", "Category", "A")
Call ChgPivotFilter("PivotTable1", "Category", "C")
Call ChgPivotFilter("PivotTable1", "Category", "B/C")
```

When your macro is running and the computer gets to one of these lines, it jumps over to the function `ChgPivotFilter` and runs through it from start to finish. Then it continues the existing macro.

To select all categories (un-filter the pivot), you would call the function like this:

```
Call ChgPivotFilter("PivotTable1", "Category", "(All)")
```

Using functions such as the above makes your overall macro code look cleaner and easier to understand! You're able to take a lot of repetitive code, package it into a function, and call that function over and over. It's much better than repeating all those lines of code multiple times!

The function above has one special feature we haven't discussed before, which is an **error handler**. The reason for it is this: if you call `ChgPivotFilter` with a field name that is not in the list (maybe Category "F"), VBA loops through every other value and unselects it. This will cause an error, because you're not allowed to have every Category unselected in a pivot filter. In case of an error, we don't want the function to crash, so there's an error handler that tells the macro what to do next. Near the beginning of the macro, we have a line like this:

```
On Error GoTo EH
```

Then at the end of the macro, we have a corresponding line starting with `EH:` (it doesn't have to be "EH", by the way; they just have to match in both places).

Anytime there's an error, the computer jumps down to `EH:` and does whatever the next command is. In this case, we use `Resume Next` to keep going and ignore the command that caused the error (this leaves the last pivot category selected).

In your own macros or functions, you could make an error handler that gives a popup error message, ends the macro, or does something else.

Other VBA and Macro Topics

The above examples should give you a running start with macros to improve your Excel productivity. Keep practicing and writing your own macros as you run into new tasks you want to speed up and automate.

To conclude this chapter, we'll address a few more issues about debugging functions, the dangers of macros, how to speed up your macros, and how to address common VBA errors.

Custom Function Debugging and Tooltips

What if your new custom-built function is misbehaving? Perhaps it returns a #VALUE! error. The good news is, you can debug functions the same way as macros: use break points, hover over variables to see their values, and step through the function. We covered these techniques earlier in the chapter.

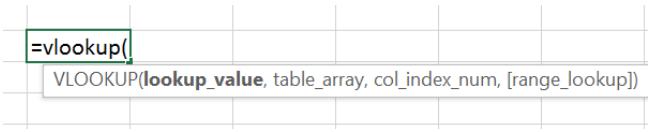
Remember that to actually "run" a function, you need to call it from the Excel sheet (although some functions are called from inside macros or other functions). If the function is just sitting in a cell, it doesn't automatically re-rerun itself. You need to specifically press [F9] to re-calculate the cell, which reruns the function. If you placed break points in the code with [F9], then it will pause at those points.

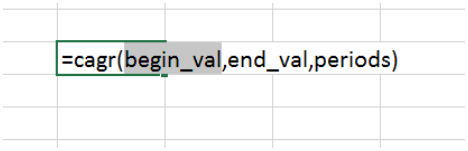
A drawback of custom functions, compared to built-in functions, is that they don't come with tooltip popups to explain the input variables. Remember that Excel functions show a very helpful tooltip to show what inputs they need, as soon as you type the open parenthesis. With user defined functions, you don't get the tooltip. There is one thing you can do, though: [CTRL] + [SHIFT] + [A].

Specifically, start typing your function name with the open parenthesis, like this:

```
=cagr(
```

for the custom CAGR function. Pressing [CTRL] + [SHIFT] + [A] **auto-fills the input names inside the function**. That's not as pretty as tooltips, and it doesn't indicate which inputs are optional – but it's something!





Dangerous Macros

As you get more comfortable working with VBA and macros, you'll probably start to ignore the warning Excel gives when opening a macro-enabled file. Don't drop your guard! Macro-enabled files can be very dangerous, and you should use caution. Here's a quick, scary list of things VBA can do:

- Automatically run when a workbook opens, without your knowledge
- Delete specific files, or even delete all files within a particular folder

- Execute unwanted Windows commands
- Log all of your keystrokes, and potentially steal website logins and passwords

You can find specific macro code snippets that do all of the above, pre-made and ready to go (yes, you could build a malicious keylogger macro with some Google and what you've already read in this book).

Don't let bad things happen to you. When in doubt, disable macros and look at the code for anything suspicious. Search for "Kill" (delete files), "Shell" (execute Windows commands), and "GetAsyncKeyState" (log keystrokes) commands as prime suspects for suspicious behavior. Also look inside the workbook and individual worksheets in the VBA editor's project tree for potential hidden code.

Speed Up Your Macros

Sometimes macros take a long time to run. And sometimes the screen flickers badly and goes crazy while your macro is running. These are not desirable things! Here are a few ways to speed up your macros.

First, **write good code**! If you've seen the code behind recorded macros, you know what bad code looks like: excessive selecting of cells, switching between sheets, scrolling, copy-pasting values, and so on. While recorded macro code is a decent foundation, you'll want to clean it up! For example, take the following piece of recorded macro code:

```
Sheets("Sheet1").Select
Range("D4").Select
Selection.Copy
Sheets("Sheet3").Select
ActiveWindow.SmallScroll Down:=-30
Range("D2").Select
Range("F3").Select
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
    :=False, Transpose:=False
Application.CutCopyMode = False
Range("H7").Select
```

Most of this is totally unnecessary! It's also quite difficult to understand what the macro is actually trying to do; in this case, it's just setting the value of one cell equal to another cell. That whole nasty thing can be replaced with one line:

```
Sheet3.Range("F3").Value = Sheet1.Range("D4").Value
```

The second way to speed up macros is to **turn off screen updating**. Just put this line of code at the beginning of your macro:

```
Application.ScreenUpdating = False
```

Then at the end, turn it back on:

```
Application.ScreenUpdating = True
```

These two lines let the macro do its thing in the background, without refreshing what the screen shows. That saves a lot of computing power! You won't see anything happening until the macro finishes, and then the results will show up all at once.

The third technique is to **turn off automatic calculations**. Normally, Excel re-calculates everything in the workbook whenever you update a cell. If you have a lot of data, and especially if you have VLOOKUP or MATCH functions, running unnecessary calculations slows your file down. You can choose to change multiple things on the sheet, then run one calculation at the end. We saw this trick in Chapter 5 when setting

calculation options (Excel Options, Formulas, Calculation Options section). Manual calculation mode means cells don't recalculate unless you press [F9].

You can replicate the same behavior with VBA. To speed up your macro, switch to manual calculations near the beginning of the macro:

```
Application.Calculation = xlCalculationManual
```

Then, call a calculation (same as pressing [F9]) before you try to read any values from the sheet:

```
Application.Calculate
```

You may end up running Calculate multiple times in the macro; basically, run it any time you want to know the result of a calculation on the sheet. At the end of the macro, it's best to switch back to Automatic calculations:

```
Application.Calculation = xlCalculationAutomatic
```

A special case of too many calculations involves PivotTables. If you're planning to make multiple changes to a pivot table (such as filtering out several categories), each change will cause the pivot to re-calculate. That can take a long time for large data sets! The trick is to turn off this automatic updating feature until you're done with all the pivot changes. You might have noticed this in the `ChgPivotFilter` function from earlier!

You can turn off automatic pivot updates like this:

```
ActiveSheet.PivotTables("PivotTable1").ManualUpdate = True
```

Then turn manual update off (automatic update on) at the end:

```
ActiveSheet.PivotTables("PivotTable1").ManualUpdate = False
```

If necessary, force an update of the pivot at any time with:

```
ActiveSheet.PivotTables("PivotTable1").Update
```

VBA Error Messages

Finally, we'll cover some common VBA error messages. If you've been working through the examples in this chapter, chances are you've encountered many of these! If you see any other errors, I recommend Googling "Excel VBA <my error>" – someone will have a good explanation for how to fix it.

"The macros in this project are disabled." You'll need to close and re-open the Excel file. As a security feature, Excel disables macros by default until you choose to enable them. When you open the file, you'll either get a popup or a bar near the top of the window that asks to enable macros. Click Enable before you start doing anything else, or the question goes away.

If you open a file with macros and don't see the option to enable macros, your security settings may be preventing them. Go to Excel Options, then Trust Center, then Macro Settings. The default setting is "Disable all macros with notification".

Note: be careful not to enable macros in suspicious files!

"Type mismatch." This error is often caused by a formula error on the sheet, not by any problems in your VBA code. There might be a `Range` function in your VBA that references a cell with an error; look for #N/A or #REF! errors on the sheet and fix them.

Type mismatch can also happen if you assign a text value to a numeric VBA variable (Integer or Double), or vice versa. You'll need to declare that variable with a different type, or change what value is assigned to it.

"Method 'Range' of object '_Global' failed." This message is a confusing way of saying: the `Range` name you are referencing doesn't exist in this Excel file. Check your named references and make sure there are no typos. VBA will highlight the line of code with the error, so it should be easy to fix.

"Application-defined or object-defined error." This error can be caused by a variety of issues. You'll see it most often when trying to open a file that doesn't exist, using `Workbooks.Open`. When debugging, check for the full file path and see if that file exists.

"Subscript out of range." This error is commonly caused by referencing a worksheet or workbook that doesn't exist. For example, these lines of code will generate a Subscript out of range error if there isn't a fourth sheet, or if there isn't a workbook open named "newfile.xlsx".

```
Sheets(4).Activate
Workbooks("newfile.xlsx").Activate
```

"Variable not defined." Remember when I mentioned that VBA doesn't require you to declare variables, and you can just start using them? That's true, unless somewhere in the beginning of the module you typed `Option Explicit`. That's a special command, on one line by itself, outside of any macros. It means you will be required to declare all of your variables with `Dim`.

To fix the error, declare your variables or remove the line `Option Explicit`.

"ByRef argument type mismatch." This is a very strange error that can come up when calling a VBA function with inputs. Consider this example, where `a` and `d` are both supposed to be integers. First we have these lines of code in a macro:

```
Dim a, b, c as Integer
a = 3
Call myfunction(a)
```

And then outside of this macro, we define a function:

```
Function myfunction(d as Integer)
```

When you run the macro, you will get a ByRef error as soon as it calls the function `myfunction`. The issue is this: when you declare multiple variables in a single line of `Dim`, the code above is incorrect (although Excel won't tell you this)! The correct way is this:

```
Dim a as Integer, b as Integer, c as Integer
```

In the shorter code above, `c` is properly assigned as an Integer, but `a` and `b` are not (they take a placeholder data type called a Variant).[60] When we pass variable `a` with the data type Variant into the function, we are assigning it to variable `d`, which is an Integer. That's what causes the type mismatch – Variant and Integer types don't match.

[60] "A Common Mistake When Declaring Variables in VBA", https://colinlegg.wordpress.com/2013/06/08/a-common-mistake-when-declaring-variables-in-vba/

If you don’t like the extra typing to declare variables, there’s an alternative solution. We can avoid passing the wrong data type to a function if we tell the function to use just the value of `a`. All you need to do is add ByVal inside the function inputs: `(ByVal d as Integer)`; then the code will work as expected.

ByVal passes variables by value rather than by reference to a function. The distinction is very technical and not worth getting into; the practical point is that `ByVal` allows more flexibility on data types with function inputs.

9: Cool Excel Tricks

This final chapter will show you various cool tricks in Excel that I've come across. It includes solutions to some common and tricky problems such as working with big and slow files, recovering data from a corrupted workbook, and the one I like to show off the most: breaking password protection in Excel.

There's no need to read this chapter in order; just read the parts you're interested in.

Speed Up a Big Slow File

We've all been there: Excel is working very slowly, pausing for several seconds every time you change a cell. Here's the process for speeding up a slow file.

Before we start, do you know which file is slow? If you have multiple large files open, even working in a small file or a brand new workbook becomes slower. By default, Excel recalculates every cell in every open workbook when you change something! That can be a lot to calculate. Close those large files if you don't need them.

Try these four steps to speed up large files, ideally in this order:

1. **Make the file smaller.** Sometimes Excel files take up much more space than they need to. On each of your sheets, do this: press [CTRL] + [END] to get to the last cell that Excel is using on this sheet. Usually nothing special happens, but sometimes you end up going much farther right or down than you expected, perhaps all the way to row 1,048,576 or column XFD. That's because at some point you (accidentally) made changes to that column or row. These extra rows are blowing up your file size. To make the sheet smaller: select and delete all unnecessary rows and columns out to the [CTRL] + [END] mark, then save and close the file immediately. When you reopen it, try [CTRL] + [END] again. It should cover a much smaller area this time, and your file should be much smaller now. I've reduced a 138MB file to 2MB this way!
2. **Replace computationally intensive calculations.** If you're using VLOOKUP on a large number of rows and multiple columns, replace them with INDEX and MATCH (see Chapter 4, INDEX-MATCH). If you have complex SUMIF or COUNTIF types of logic on over 500 rows of data, I recommend using PivotTables instead (see Chapter 6).
3. **Turn off automatic calculations.** On really large and complex sheets, maybe it's better to only recalculate cells when you want to. Go to Excel Options, Formulas, and under the Workbook Calculation heading choose Manual (see Chapter 5, Calculation Options).
4. **Too much data? Use a database.** If you're asking Excel to do serious calculations on multiple tables of data, each with thousands of rows and many columns, you're probably using the wrong tool

for the task. Excel is not meant to handle such a large amount of data. Especially at a file size of 50MB or more, you're pushing the program's limits. Your best alternative is to use a database to store your data, and query that database from Excel. You can embed automated queries into Excel, but that's beyond the scope of this book. Look into Access databases, SQL, or Google BigQuery for some alternatives when working with large amounts of data.

Repair a Corrupted File

Sometimes Excel crashes – that's why you should save your work often, and make multiple versions of your files! Now let's talk about what to do if it's too late: Excel crashed, you lost some work, and you're trying to reopen the file.

Sometimes Excel can just recover everything automatically. Sometimes you lose one or two pivot tables (they convert to just the values), but otherwise everything is still there. That's the best-case scenario; just pick up from where you left off, and hopefully it doesn't crash again.

If you recover the file but it keeps crashing, you might have to recreate everything in another file. In this case: move your data, formulas, and everything besides PivotTables and Charts to a new file. You can copy the sheets themselves, or just the contents on these sheets. Then recreate pivots and any other complex objects in this new file. Don't forget any macro code behind the scenes!

Now for the worst-case scenario – what if Excel cannot automatically recover your file? Here's how to **manually repair corrupted files**, to the extent possible:[61]

1. Go to File, Open and browse for the corrupted file you want. Don't open it yet!
2. There's a dropdown arrow next to the Open button. Click that and choose Open and Repair
3. Choose Extract Data instead of Repair.

Excel will do its best to at least recover the values of cells. Any formulas, pivot tables, and other nice things will likely be gone. But it should still be better than losing everything and starting over.

Fix Annoying Errors

"A formula in this worksheet contains one or more invalid references." This error might pop up from time to time, even though nothing appears to be wrong in your file! It's just very annoying and seemingly random. Your formulas are probably fine; 99% of the time, the problem is with a chart reference that is now broken.

We talked about how to fix this error at the end of Chapter 6; take a look for the explanation. You'll basically need to fix the source data for one or more of your charts.

"Too many different cell formats." This is a really weird error, and in my experience doesn't really mean what it says. When you see this error, Excel is about to crash! Save immediately; use Save As with a different file name in case the file becomes corrupted (not likely but possible). Then close Excel, not just this file, but the entire program. When you reopen it, the problem should go away.

[61] "Repairing a corrupted workbook", Microsoft Support. https://support.office.com/en-us/article/Repairing-a-corrupted-workbook-e5b49891-dde7-4796-b60a-49b0d2478a62

Avoid Array Formulas

In my 15+ years of working with Excel, I've never found a good reason to use array formulas. You're not missing anything if you don't learn about them; they'll probably confuse much more than help. Feel free to Google some tutorials to see why they are so terrible.

The good news is: everything you might want to do with array formulas, you can also accomplish with regular formulas! Keep in mind two Advanced Excel tips:

- Set up interim calculations. For example, you might have two columns of numbers, columns B and C. Add in new columns with calculations such as the difference or sum of these values; then, you can run calculations on the new columns. There is no shortage of columns available for these interim calculations: Excel gives you 16,000 columns to work with!
- The OFFSET function works very well when dealing with arrays. We covered it in Chapter 3: Functions.

Approximate Matching for Text

While a computer can easily find exact matches between two text entries (and search through millions of entries within seconds), approximate matching is very difficult. Do you want to match up a list of names, and consider "John Smith" to match "Johnny Smith", "John D. Smith", and "Jon Smith"? That's called fuzzy matching, and it's a very difficult task for a computer program, including Excel.

There are three viable ways to execute approximate matching (also known as "fuzzy matching") in Excel, although sadly, none of them are perfect.

The first way involves the **wildcard characters**, * and ?. The asterisk (*) represents any characters of text (including multiple characters or nothing); the question mark (?) represents exactly one character. You might use these characters in VLOOKUP, MATCH, COUNTIF, or SUMIF functions. You can also use them with Find and Replace ([CTRL] + [F]). Here are a few examples:

- `MATCH("C*", A1:A10, 0)` look for anything starting with a C in range A1:A10
- `COUNTIF(A1:A10, "*Smith*")` count number of cells containing "Smith" in range A1:A10

Wildcards do not work when checking for simple equality between two cells. If cell A1 has the value "Hello", then you would expect this formula to return TRUE:

```
=A1="Hel?o"
```

It actually returns FALSE. To trick Excel into behaving properly with wildcards, use the MATCH function instead of checking for equality:

```
=MATCH("Hel?o", A1, 0)
```

That returns a 1 if matching and #N/A if not matching. You know from Chapter 5 how to handle this error gracefully:

```
=IFERROR(MATCH("Hel?o", A1, 0) ,0)
```

To match the actual question mark or asterisk characters (not as wildcards but as literal characters in text) use the ~ character before the * or ? as an escape character. To look for all cells starting with "Hello?" in a COUNTIF function, use "Hello~?*". It lets Excel know you are looking for an actual question mark, followed by the asterisk wildcard.

Wildcards are useful when you know specifically what positions in your text should be wildcards. But to recognize minor typos and other small variations between text values (which can occur anywhere), you need a different method. One solution is to use **matching algorithms**, which are mathematical formulas built by smart people to measure the similarity between two strings of text.

One such algorithm is called Levenshtein distance, which counts the number of edits you would have to make to the first text to make it equal to the second one.[62] You can build a custom Excel function (see Chapter 8 for custom functions) that measures the Levenshtein distance between two inputs. Just Google "Levenshtein distance Excel VBA" to find a suitable piece of code to copy-paste into your file. I usually prefer code examples from Stackoverflow.com.

As a third option, Microsoft designed a fuzzy lookup add-in for (you guessed it) fuzzy lookup of text. Just search for it by name on the Microsoft website.[63] I won't go into too much detail on this add-in besides giving my initial impressions of it: it isn't particularly polished, and it takes time to learn. I'm not a fan, but perhaps you'll find it useful.

Two-Dimensional Lookup Table

This section is for the fans of VLOOKUP and INDEX-MATCH. Have you ever needed a two-dimensional lookup table – a table that returns one result based on two inputs? You can make such a table with MATCH and OFFSET.

The screenshot below shows the full setup. You need a lookup table (range A6:I11) with headings along the top and the left. In my example, the lookup table outputs either "GOOD" or "BAD".

We need two inputs – in this case, A and B in cells B2 and B3. The values of these cells could come from somewhere else, maybe as outputs of a complex model. They could also just be hard-coded inputs. Next to each input we calculate a "match" value to determine which row and column of the lookup table they represent. The formulas for C2 and C3 are:

```
=MATCH(B2, B6:I6, 1)
=MATCH(B3, A7:A11, 1)
```

In each formula we take the input value, match across the row headings (B6:I6) or column headings (A7:A11), and use type "1" for ascending range lookup.

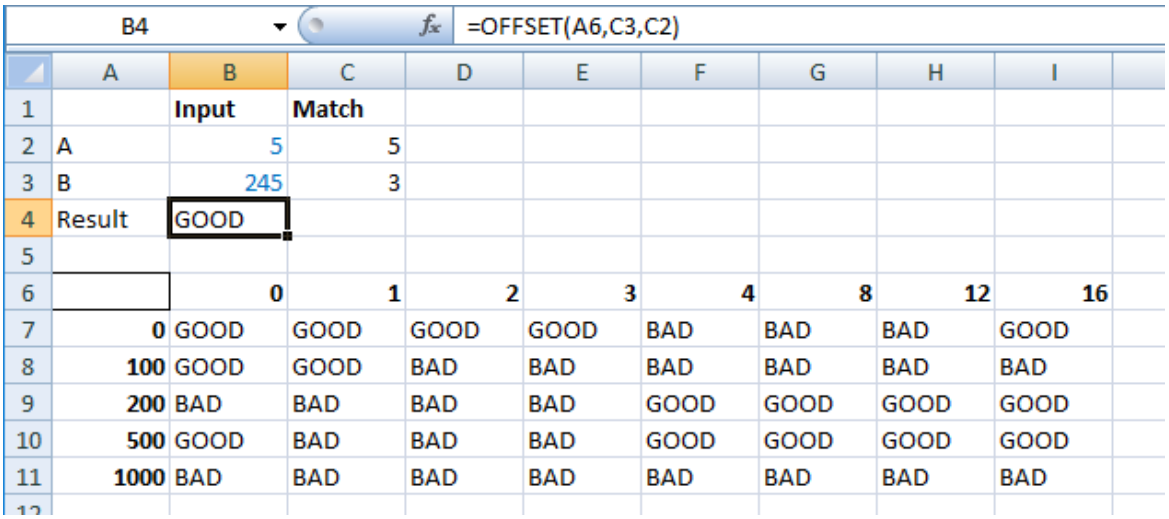
Input A = 5 matches column 5 in the table, because it's greater than or equal to 4 but less than 8. Input B = 245 matches row 3, greater than or equal to 200 but less than 500.

The final result goes in cell B4 and uses an OFFSET function:

```
=OFFSET(A6, C3, C2)
```

This tells Excel to take the top-left corner of the table, and offset by C3 number of rows (down 3 rows) and C2 number of columns (right 5 columns) to get the result. Use different inputs to test that the formula works!

[62] "Levenshtein distance", Wikipedia. https://en.wikipedia.org/wiki/Levenshtein_distance
[63] "Fuzzy Lookup Add-In for Excel", Microsoft. http://www.microsoft.com/en-us/download/details.aspx?id=15011



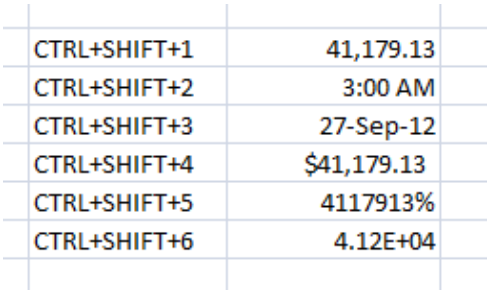
B4 =OFFSET(A6,C3,C2)

	A	B	C	D	E	F	G	H	I
1		**Input**	**Match**						
2	A	5	5						
3	B	245	3						
4	Result	GOOD							
5									
6		**0**	**1**	**2**	**3**	**4**	**8**	**12**	**16**
7	**0**	GOOD	GOOD	GOOD	GOOD	BAD	BAD	BAD	GOOD
8	**100**	GOOD	GOOD	BAD	BAD	BAD	BAD	BAD	BAD
9	**200**	BAD	BAD	BAD	BAD	GOOD	GOOD	GOOD	GOOD
10	**500**	GOOD	BAD	BAD	BAD	GOOD	GOOD	GOOD	GOOD
11	**1000**	BAD	BAD	BAD	BAD	BAD	BAD	BAD	BAD

Custom Number and Date Formats

You can display your numbers or dates in just about any format in Excel. Keep in mind that number formatting doesn't change the value of a cell, it just changes the way it is displayed. If you reference a cell inside formulas elsewhere, changing the original cell's number formatting has no impact on those formulas.

The fastest way to apply basic number formatting is the keyboard shortcuts [CTRL] + [SHIFT] + [1 through 6] from Chapter 3. Here's the same input value displayed with six different number formats:



CTRL+SHIFT+1	41,179.13
CTRL+SHIFT+2	3:00 AM
CTRL+SHIFT+3	27-Sep-12
CTRL+SHIFT+4	$41,179.13
CTRL+SHIFT+5	4117913%
CTRL+SHIFT+6	4.12E+04

To increase or decrease decimals, I like to use [ALT] [H] [0] and [ALT] [H] [9], which navigates the Home ribbon for the appropriate shortcuts.[64] The Number area of the Home ribbon includes commonly used "accounting" ($ sign), "percent style", and "comma style" buttons. These are good alternatives to the keyboard shortcuts above. You'll also find Currency and Number formatting choices in the Format Cells popup with [CTRL] + [1].

We've just reviewed three different ways to apply comma and decimal formatting to numbers, with or without currency symbols. Here's what the various formats look like in Excel; the ones whose names are in quotes are from the Format Cells dialog box:

[64] Here's how I remember which is which: [0] adds an extra zero, while [9] removes a zero.

CTRL+SHIFT+1	1,000.00	0.00	-1,000.00	
Comma Style	1,000.00	-	(1,000.00)	
"Number"	1,000.00	0.00	-1,000.00	
CTRL+SHIFT+4	$1,000.00	$0.00	($1,000.00)	<-- this is red
Accounting format	$ 1,000.00	$ -	$ (1,000.00)	
"Currency"	$1,000.00	$0.00	-$1,000.00	
"Accounting"	$ 1,000.00	$ -	$ (1,000.00)	

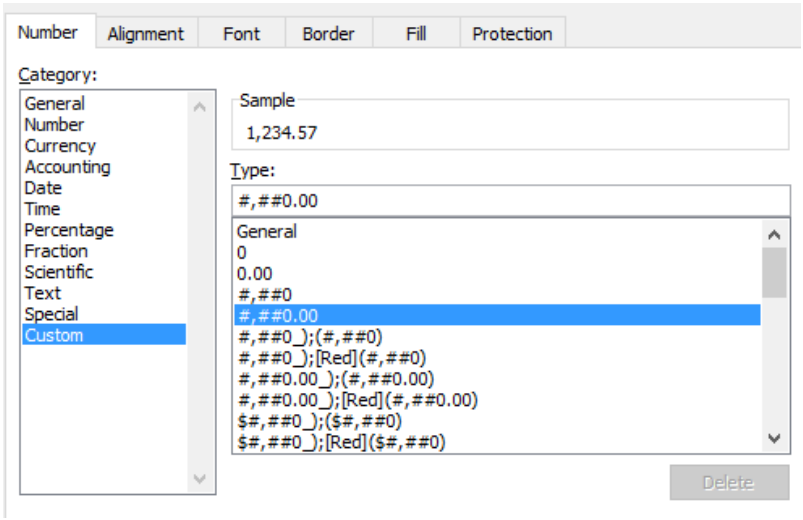
Notice the many small differences and inconsistencies! Some of the numbers align flush to the right, and some have a slight indent. Some have the dollar sign right next to the first digit; others align it all the way left. Some use a dash for zero and others write out 0.00; some use minus signs and others use parentheses for negatives. There's even red text for one of the negative values.

Do these small differences bother you? If so, be careful not to mix how you apply number formatting (keyboard, Home ribbon buttons, or Format Cells). In my opinion, your sheet looks a lot less professional if your number formats are inconsistent.

Back to Format Cells. Excel gives you a lot of built-in formats, which are sufficient most of the time. Here are some interesting or obscure number format choices you may have missed:

- Use parentheses and/or red font for negative numbers (under the Number and Currency categories)
- Use fractions such as 23/25 under the Fraction category
- Display zip codes, including leading zeros, under the Special category

Finally, the Custom category lets you define any number format you like. The trick is to learn how Excel's special number format notation works. Fortunately, Excel gives you a lot of built-in examples, and provides a sample output as you type out a custom format.



I'll give you a quick orientation to these symbols:

- "0" stands for a required digit, so "0.00" format forces 2 decimals and a single digit before the period

- "#" stands for optional digits; it's typically used as a filler to place currency symbols and thousands commas
- Use text within quotes to output that text, like this: 0 "cars"
- If you just type General, nothing else, you get unformatted numbers
- You can use a semicolon (;) to separate your format string into multiple formats for different values (positive values; negative values; zero value; text values)

Here are some examples:

Cell Value	Format String	Output
1234.567	0.00	1234.57
1234.567	0	1235
0.501	0.00	0.50
0.501	.00	.50
0.501	#.00	.50
1234.567	#,##0	1,235
1234.567	#,###	1,235
0	#,##0	0
0	#,###	(blank)
0	#,#00	00
1234.567	General	1234.567
1234.567	0 "feet"	1235 feet
2100	+#,##0;-#,##0;0	+2,100
-2100	+#,##0;-#,##0;0	-2,100
0	+#,##0;-#,##0;0	0

Using text as part of your number formats, such as "1235 feet" above, can be very handy! The cell is labeled with proper units (feet), while the cell's value is still just a number (1235) that you can use in formulas.

The last three examples show how you can format positive, negative, and zero values differently in the same format string, separated by semicolons. So we use this long format string that really has three parts: +#,##0;-#,##0;0. It applies a plus sign for positive numbers (+#,##0), a negative sign for negative numbers (-#,##0), and a regular 0 for zeroes (0).

Custom number formats are especially useful for dates. I personally don't like any of Excel's built-in date options. To build dates with Custom formats, use the letters y, m, and d:

- Use y or yy to show a 2-digit year (16) and yyy or yyyy for a 4-digit year (2016)
- Use m for the month's number (4), mm for 2-digit month number (04), mmm for abbreviated month name (Apr), and mmmm for full month name (April)

- Use d for the day's number (8), dd for 2-digit day (08), ddd for abbreviated day name (Fri), and dddd for full day name (Friday)

Examples:

- mmm yyyy Jan 2016
- mm.dd.yyyy 01.08.2016
- dddd, mmmm d Friday, January 8

As a bonus, if you really want to mess with somebody who doesn't know about number formats, apply one of these formats to the whole worksheet:

- General"0"
- -0;0;0

Printing Tips

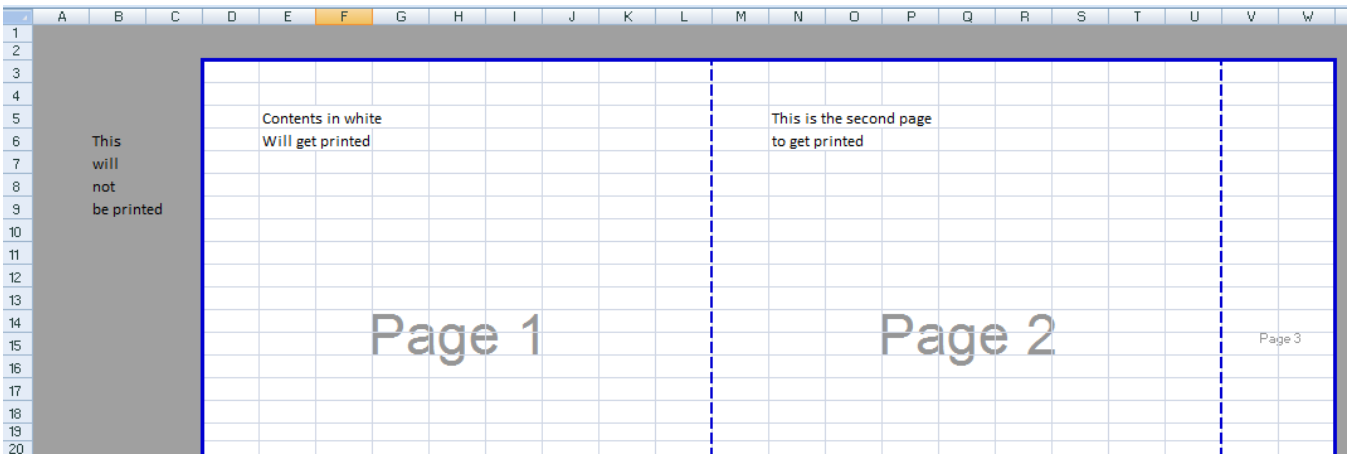
Printing is simple, right? Click print (or [CTRL] + [P]), and that should be it. Nevertheless, the tips in this section should improve your printing experience – whether you print to physical paper or to a PDF file.

Use the **Page Break Preview** under the View ribbon. It lets you set the exact range you want to print, skipping any extraneous data or formulas. Click and drag the thick blue borders in Page Break Preview mode to set the print area. Everything in white will be printed, and the area in gray will not.

The dotted lines represent page breaks – drag them all the way to the edges to force everything onto one page.

You can clear your print areas and start over: select a range of cells, right-click, and choose Set Print Area. That range becomes your new print area.

You can also create multiple blocks of separate print areas: select a new range of cells, right-click, and choose Add to Print Area.



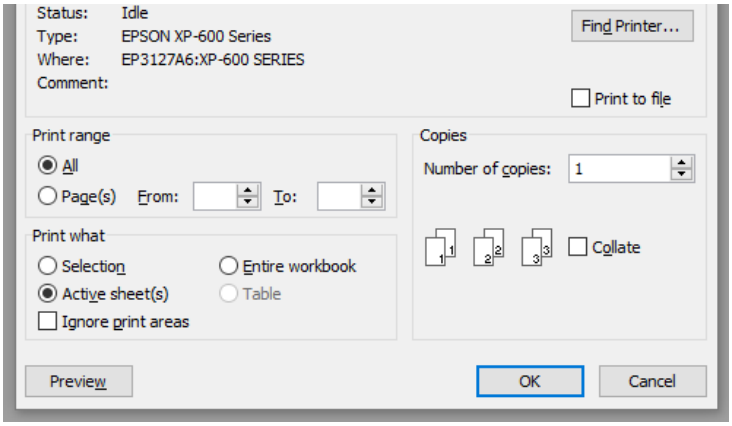
Print Preview is a nice way to test out how things will actually look on paper.

If you want to fit everything on one page, use **Page Setup**, then choose Scaling Fit to 1 page wide by 1 page tall. I don't recommend using the "X% of normal size" – it's hard to accurately calculate what percent will fit on your paper. Be careful, though: fitting to one page can make your printouts unacceptably small. If

you're squinting to read the smallest font on the page, it's not readable. Try two pages, or reduce the print area.

You can also reduce your margins to fit a little more on one page. The defaults are 0.7 on the left and right, and 0.75 on the top and bottom. You could safely reduce these by one click each, to 0.45 and 0.5. Choosing Center on Page on the Margins tab makes printouts look better.

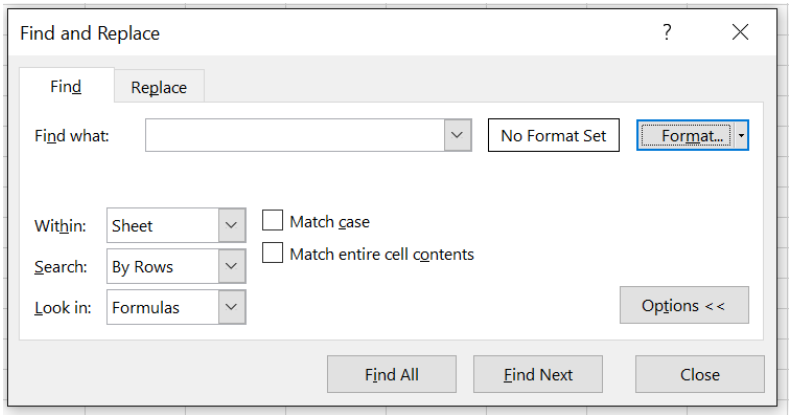
To **print multiple sheets** (but not all sheets in the workbook), hold down [CTRL] or [SHIFT] while clicking on multiple worksheet tabs. Then in the Print window, choose Active Sheet(s), which should already be selected. To help you keep track, it's also a good idea to right-click and color the tabs you intend to print; maybe make them all green.



Find/Replace Tips

Searching is about as simple as printing, but perhaps I can show you some new tricks here. For starters, you should be using the keyboard shortcuts [CTRL] + [F] for Find, and [CTRL] + [H] for Replace.

Let's look at some of the options under the Find popup; open these with the Options button.



These settings are pretty self-explanatory, but I do want to call attention to Look in Formulas or Values. If your sheet pulls in data with formulas such as VLOOKUP, searching for data in Formulas (the default setting) is no help. Switch to searching in Values.

You can use Find to look for **references to other files or other sheets**. Search in Formulas for ".xls" to get references to outside files, or "!" to get references to other sheets. You probably want to search within the entire Workbook, not just the Sheet.

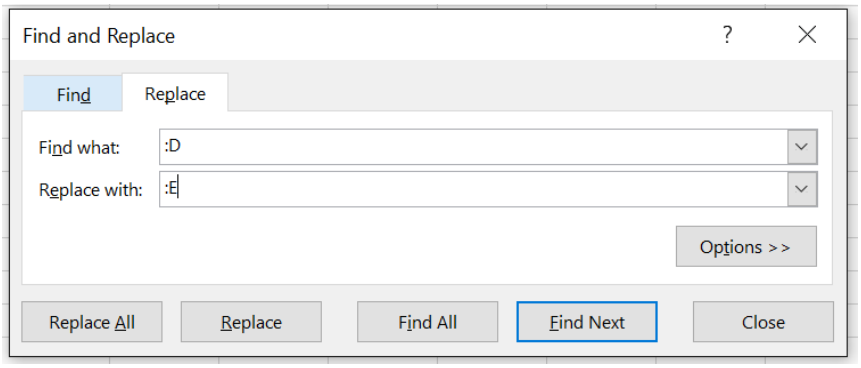
Here are two specific ways I like to use the Replace feature:

Replace to modify a formula in a column, instead of dragging down the formula. Picture this situation: a bunch of cells in column B have a formula such as `SUM(C10:D10)` in cell B10. But some cells in this column have different formulas, or are blank. You want to replace the sum formulas to go through column E instead of D. How do you make that change so that it only affects the appropriate formulas?

One option is to be very careful, and copy-paste the new formula over the correct rows. But if you have a lot of exceptions and skipped rows, this method takes some time and effort. Try this alternative:

1. Select the entire column
2. Open up Replace with [CTRL] + [H]
3. Type in Find ":D" and Replace with ":E"
4. Replace All

You'll end up with SUM(C10:E10) and the equivalent formulas for each row! Notice that we used the colon inside the Find term, to really narrow down the results. We don't want to accidentally overwrite every occurrence of the letter D!



Replace to remove external links. If you copy a sheet from one file to another, that sheet can come with references to other sheets in the old file. Remember, these external references look something like:

```
=[OldFile.xlsx]Sheet2!A1
```

Use Replace to get rid of the [OldFile.xlsx] part of this formula: search for [OldFile.xlsx] and replace it with nothing. You'll remove the external file part of this formula (and all others on the sheet), and instead you'll get:

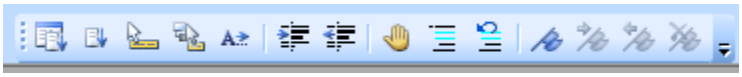
```
=Sheet2!A1
```

This approach works very well if you intended to reference Sheet2 in the new file. Replace is the fastest way to get rid of unwanted external links.

Excel Settings Tips

You'll rarely need to mess with Excel settings and defaults, but the following are good to know when setting up your work environment.

1. **Hide Gridlines.** This is not technically a setting, but you can find a checkbox for Gridlines under the View ribbon in Excel 2007.[65] (I've also seen this done by applying white background color to the entire sheet.)
2. **Show Developer Tab.** You'll use the Developer Tab to build and run macros. To turn it on, go to Excel Options. For Excel 2010 and newer, go to Customize Ribbon and check the Developer box. For Excel 2007, go to Popular options and check the box for Show Developer tab in the Ribbon.
3. **Calculation mode: manual.** We've mentioned a few times in this book that Excel's automatic calculations can be really slow when working with large files. Use Excel Options, Formulas, and switch Workbook Calculation to Manual. It's a good idea to keep Recalculate Workbook Before Saving checked. In manual calculation mode, press [F9] anytime you want to actually recalculate formulas. This will be really weird at first, and it will look like Excel is broken! Just remember to use [F9] to calculate.
4. **Turn off GetPivotData**. In Chapter 6, we talked about how Excel automatically uses the GETPIVOTDATA function when you reference a cell in a pivot table. These types of formulas are more difficult to work with; you can't drag down the formula in the way you expect. To turn off GETPIVOTDATA forever and allow regular references (like "B3") in pivots: go to Excel Options, then Formulas, and uncheck the box for Use GetPivotData Functions for PivotTable References.
5. **VBA comment block buttons.** If you're writing macros and functions in VBA, you'll want to easily be able to comment or un-comment many rows of code at a time. With the VBA window option on (use [ALT] + [F11]), right-click the toolbar on top and make sure Edit is checked. The Edit toolbar in VBA gives you two very useful buttons, which let you add or remove comment quotes for selected lines of code. It looks like this, and the comment buttons are just to the right of the hand icon:

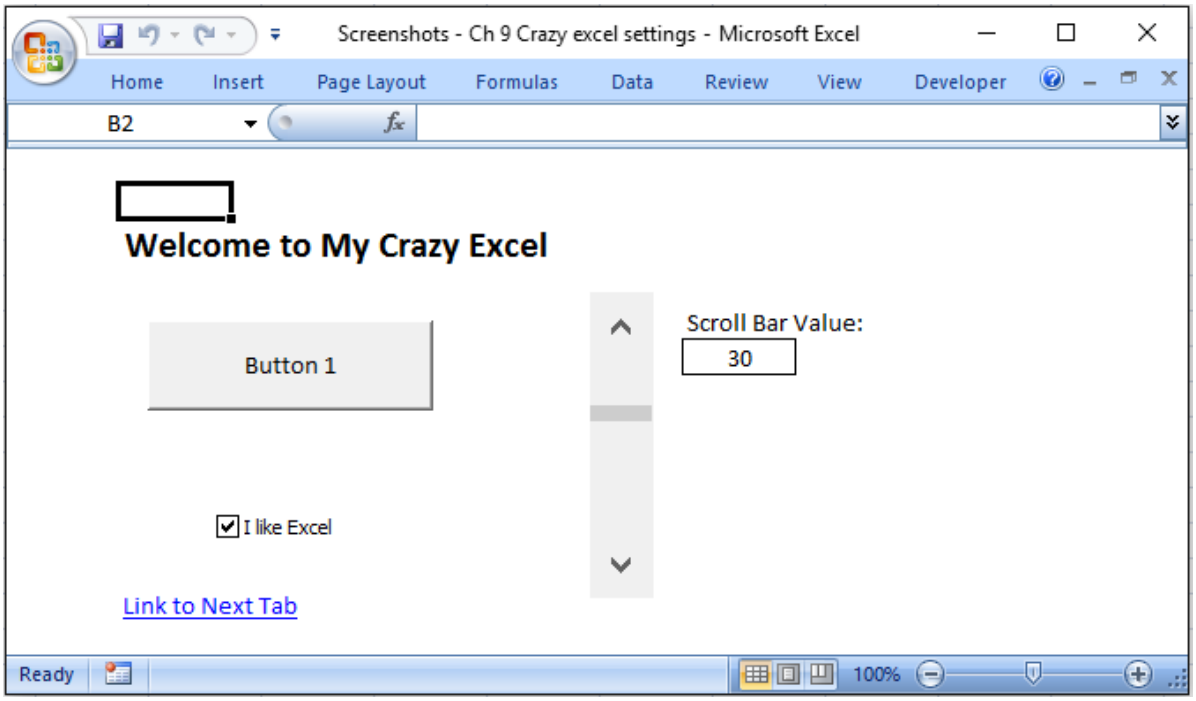


Besides these common-sense settings changes, you can choose to make Excel look nothing like Excel. Try the following:

1. **Hide Gridlines, the Formula Bar, and Column/Row Headings**. These three settings are found under the View ribbon. Look for the checkboxes.
2. **Hide the Ribbon**. Right-click the ribbon itself or the menu bar above it, and choose Minimize the Ribbon.
3. **Hide scrollbars and sheet tabs**. These settings are under Excel Options, then Advanced, then scroll down to the "Display Options for this Workbook" section.

You can also add various form control elements and buttons (under the Developer ribbon, Insert), as well as hyperlinks (right-click a cell and click Hyperlink). The end result might look something like this:

[65] In Excel 2003, go to Tools, Options, View, then uncheck Gridlines.



Excel Security and Cracking Passwords

There are three common places you'll likely encounter password protection in Excel:

1. Locked cells or workbooks
2. Protected VBA code
3. Password to open the Excel file

Password protection is worth using for a basic amount of security, but don't rely on any of the above options for safeguarding truly sensitive and important documents!

Disclaimer: I'm not claiming responsibility for any adverse outcomes from breaking Excel passwords. Usually the ethical route is to ask for permission and obtain the original password from the file's creator. We'll proceed with the assumption that you are, in fact, the rightful user of a protected file, but you misplaced or forgot the password. That happens all the time.

The first kinds of protection, locked cells or workbooks, are the simplest to break. In Chapter 7, we looked at password-protecting sheets so they cannot be edited. To break this password: Google "Excel unlock password VBA" and you will quickly find the code for a macro that does the job. I posted a downloadable macro that has worked for me on this book's website, www.AdvancedExcelBook.com.

Another disclaimer: I cannot take the credit for this macro; it was originally written by a guy named Kevin Jones.

That's it – just copy the password-cracking macro code, and paste it into your VBA editor (see Chapter 8 for more information on VBA). I'm not revealing some big secret; the answer really is that easy. I recommend keeping this code saved in a text file somewhere in case you need it.

The second type of password protection is to protect your VBA code itself. Here's how to set up VBA protection in your own files:

1. Open the VBA editor ([ALT] + [F11])
2. In the project tree, right-click the name of the file and choose VBAProject Properties
3. In the dialog box, switch to the Protection tab
4. Check the box Lock Project for Viewing and choose a password
5. Save this file as a macro-enabled workbook and close it
6. When you re-open the file, you'll be unable to look at its VBA code without providing the password.

To crack the VBA password, there are several solutions at a website called StackOverflow.com; you can search for "Is there a way to crack the password on an Excel VBA Project?"[66] and read the top solutions. Some involve using a Hex editor to change just one little piece of code in the Excel file. My favorite is posted by a Vietnamese developer named Duc Thanh Nguyen; it involves running a piece of VBA code in another Excel file while the original file is open. It's as simple to use as the worksheet unlocking code; head to www.AdvancedExcelBook.com for a downloadable version.

Finally, to crack passwords that prevent you from opening an Excel file, you'll need more sophisticated software. There are several available for purchase online, generally for under $80 (still feeling pretty good about that valuable password-protected Excel file of yours?). Google for something like "Excel Password Recovery" to find a wide array of choices. Make sure the one you buy includes the ability to open password-protected files. You might consider www.lostpassword.com, although I cannot vouch for their product.

Great Add-Ins

Excel's built-in functionality is great, and with VBA you can be even more productive. You might also be interested in Excel add-ins, which give you additional features and menu items that are not part of regular Excel. Here are two of the best-known ones:

Analysis ToolPak. This one comes pre-installed with Excel, but you might need to enable it using these steps:

1. Go to Excel options
2. Click Add-Ins on the left
3. You'll see Analysis ToolPak and Analysis ToolPak VBA under the heading for Inactive Application Add-Ins; enable these and click the Go button at the bottom
4. In the popup, check the boxes next to the Analysis ToolPak items (and anything else that interests you) and click OK.

To actually use the Analysis tools, go to the Data ribbon and click on the new Data Analysis button, all the way to the right. You can use this add-in for regressions and other statistical tools.

ASAP Utilities is a downloadable free add-in that speeds up common tasks in Excel. It contains simple tools for a huge number of tasks, including:

- Delete excess space characters in a range of text, with one click
- Add row numbering with advanced features such as categories
- Select cells based on their formatting
- Apply print settings from one sheet to other sheets

[66] "Is there a way to crack the password on an Excel VBA Project?" StackOverflow. http://stackoverflow.com/questions/1026483/is-there-a-way-to-crack-the-password-on-an-excel-vba-project/1038783#1038783

A lot of these features can be replicated with some simple functions and techniques that we've covered in this book. But an Advanced Excel user knows the fastest and easiest way to get something done, which is often the ASAP Utilities way. There's a tutorial available on their website.[67]

[67] "Examples of how ASAP Utilities will save you time and make YOU rock in Excel." ASAP Utilities. http://www.asap-utilities.com/asap-excel-tutorial-how-to.php

10: Conclusion

Advanced Excel for Productivity is not just a book title – it is a skill that should be honed and practiced. You will not remember everything in this book just by reading it once, but I've tried to organize the book so you can find what you need when you need it.

Now that you're done reading this book, what's next?

1. Re-read the parts that were new to you, and follow along with a similar real-world example. Repetition helps you retain the material.
2. Print out the reference sheets (keyboard shortcuts, functions, and VBA) and keep them close to your desk. They are available to download from www.AdvancedExcelBook.com.
3. Use your new skills as much as possible!

Index