

# CISA315: Introduction to Electronic Spreadsheet

Tak Auyeung, Ph.D.

November 10, 2005

- 20050214 TA: Added chapter 5.
- 20050206 TA: Added chapter 4.
- 20050130 TA: Added formula related stuff at 3.
- 20050120 TA: Added more interesting stuff for autofill at 1.4
- 20050117 TA: Added first chapter (week one) at 1

# Contents

<b>1</b>	<b>Week One, Supplemental</b>	<b>7</b>
1.1	What is a spreadsheet, anyway? . . . . .	7
1.2	Do I need Excel for this class? . . . . .	7
1.2.1	Getting OpenOffice . . . . .	8
1.2.2	How does Calc compare to Excel? . . . . .	8
1.2.3	Other versions of Excel . . . . .	8
1.3	Final Project . . . . .	8
1.4	More on AutoFill . . . . .	9
1.4.1	Autofilling a Large Range of Cells . . . . .	9
1.4.2	Custom Increments . . . . .	9
1.4.3	In General... . . . .	10
1.4.4	Autofilling from One Cell . . . . .	10
<b>2</b>	<b>Week Two: Formatting and Formulae</b>	<b>11</b>
2.1	Formatting . . . . .	11
2.1.1	It's a Table! . . . . .	11
2.1.2	Printing . . . . .	11
2.1.3	Online Viewing . . . . .	11
2.2	Formulae . . . . .	11
2.2.1	Cell Referencing . . . . .	11
2.2.2	The Simplest Formula . . . . .	12
2.2.3	Moving a Range of Cells . . . . .	12
2.2.4	But Wait! . . . . .	12
2.2.5	Absolute Referencing . . . . .	13
2.2.6	Absolute Row and/or Column . . . . .	13
<b>3</b>	<b>Week Three</b>	<b>15</b>
3.1	Revisiting Autofill . . . . .	15
3.2	Calc and Excel . . . . .	15
3.3	Conditionals and Logic . . . . .	15
<b>4</b>	<b>Week Four</b>	<b>17</b>
4.1	Formatting and Printing . . . . .	17
4.1.1	OpenOffice Calc . . . . .	17
4.1.2	Printing to PDF . . . . .	17
4.2	Spreadsheet vs. Database . . . . .	18
4.2.1	Rows and Columns . . . . .	18
4.2.2	Sorting . . . . .	18
4.2.3	List . . . . .	18
4.2.4	Applications of Lists . . . . .	19
4.2.5	When to switch to a database? . . . . .	19
4.3	Publishing HTML from Excel . . . . .	19
4.3.1	Worksheets with Static Data . . . . .	19
4.3.2	Worksheets with Dynamic Data . . . . .	19

4.3.3	How do we make an HTML document interactive? . . . . .	20
4.4	Using “Live” External Data . . . . .	20
<b>5</b>	<b>Week Five</b>	<b>21</b>
5.1	Cell referencing functions . . . . .	21
5.1.1	The Problem . . . . .	21
5.1.2	Computing the Row Numbers . . . . .	21
5.1.3	The “Address” Function . . . . .	21
5.1.4	The “Indirect” Function . . . . .	22
5.1.5	General Use . . . . .	22
<b>6</b>	<b>Week 8</b>	<b>23</b>
6.1	Database and Spreadsheet . . . . .	23
6.1.1	Spreadsheet versus Database Table . . . . .	23
6.1.2	When to use a database table? . . . . .	24
6.1.3	Combining a spreadsheet and a database table . . . . .	24
6.1.4	vlookup and hlookup . . . . .	24
6.2	Goal Seek and Solver . . . . .	27
6.2.1	Goal Seek . . . . .	27
6.2.2	Solve . . . . .	27

**Copyright Notice**

All materials in this document are copyrighted. The author reserves all rights. Infringements will be prosecuted at the maximum extent allowed by law.

You are permitted to do the following:

1. add a link to the source of this document at [www.drtak.org](http://www.drtak.org)
2. view the materials online at [www.drtak.org](http://www.drtak.org)
3. make copies (electronic or paper) for *personal* use only, given that:
  - (a) copies are not distributed by *any* means, you can always refer someone else to the source
  - (b) copyright notice and author information be preserved, you cannot cut and paste portions of this document without also copying the copyright notice



# Chapter 1

## Week One, Supplemental

Chapter 1 of the textbook talks a lot about basic operations and navigation in Excel. There is no way I can do this with all the screen shots and detailed explanations. Nonetheless, I find certain elements to be missing from the first chapter of a book that talks about Excel.

### 1.1 What is a spreadsheet, anyway?

Excel is a Microsoft product that allows users work with spreadsheets. Before Excel became dominant, there was Lotus 1-2-3. And, before that, there was VisiCalc. Those who are interested in the history of spreadsheet programs should visit <http://dsresources.com/history/sshistory.html>.

So, what exactly is a spreadsheet?

The following elements characterize a spreadsheet:

- Data is organized as a table.
- A spreadsheet table can be very large. Conceptually, there should be no limit to the size of a spreadsheet table.
- Each cell in a spreadsheet table can hold text, numeric or formula data.
- *Most importantly*, The formula of one cell can refer to other cells.

Great. So, what is a spreadsheet? Come on!

Essentially, a spreadsheet is table. This means it can be used to store shopping lists, phone number lists, schedules and etc. However, the additional ability to define a cell by other cells makes a spreadsheet much more useful. It can be used to store invoices, analyze data from scientific experiments, compute values for tax returns and much, much more.

### 1.2 Do I need Excel for this class?

The answer is no. You can use OpenOffice Calc instead.

OpenOffice is an office suite like Microsoft Office. However, OpenOffice is free software. You can download it from the web, or ask someone else to download it, then burn it on a CD for installation. OpenOffice includes the following major components:

- Calc: this is the spreadsheet program. It can read/write Excel formats in addition to its own format.
- Write: this is the word processor program. It can read/write Word formats in addition to its own.
- Impress: this is the presentation program. It can read/write PowerPOint formats in addition to this own.

In other words, in addition to being useful for this class, OpenOffice can be quite useful for most office productivity applications.

### 1.2.1 Getting OpenOffice

- Go to <http://www.openoffice.org>.
- Click the “Download” tab.
- Click “Latest Stable Release...”
- Select language, operating system and download site.
- Read how you can contribute to a free product! You don’t have to send money. Reporting bugs and making suggestions are both good contributions. Spreading the word is another way you can make a difference.
- Click “Continue to Download”.
- When prompted, select a folder to store the file. Be sure to remember this folder!
- The download size is about 65MB. This means it will take a while to download, even with broadband connection.
- Once downloaded, run the downloaded installation file, and follow the instructions. Accepting defaults should work for most people.

### 1.2.2 How does Calc compare to Excel?

This is a natural and very real question.

For the most part, Calc will do exactly what Excel does, including keyboard shortcuts and some of the toolbar icons. This is especially the case for this class. When you get *much* more advanced, you may find features that are missing from Calc. One such feature is the use of Visual Basic in Excel. Calc uses Java as its internal scripting language.

Some features are placed differently in Calc. For example, to set the layout of a page (for printing purposes), you access the File menu, then “Page Setup” in Excel. In Calc, on the other hand, you access the Format menu, then select “Page ...”. Personally, I find the Calc organization more sensible. To veteran Excel users, however, this difference can be a little bit annoying.

It is a little early, but there is another important difference. When you enter a formula, or whenever you need specify a list of cells, Excel uses the comma , symbol. On the other hand, Calc uses the semi-colon ; symbol for this purpose. This is a minor difference, but enough to send experienced users scratching their heads.

There is, however, one feature that is definitely on the plus side of Calc. All OpenOffice applications can export to PDF natively. This means you don’t have to purchase Acrobat to generate PDF documents in Calc.

If you run into any trouble using Calc instead of Excel, please send me a message.

### 1.2.3 Other versions of Excel

For the most part, other versions of Excel will work just fine. This is especially the case for Excel 2002, which is nearly identical to Excel 2003 for the purpose of our class.

## 1.3 Final Project

Your final project will carry a large percentage of the final grade (25%). So, it is time to start thinking about it.

The final project needs to demonstrate that you have learned everything in the scope of this class, and that you are able to apply the knowledge. Traditionally, you will be given the final project topic. I think it is better if you think of a topic that interests you, so that you have more incentives to complete the final project.

For the purpose of evaluation, your final project cannot be too simple. It needs to involve the use of practically all the features of Excel (or Calc) that we will discuss in this class. I will give you some suggestions here.

- Household budget analysis.
- Business cashflow analysis
- Energy audit.
- Retirement investment projection.

You can see that all of these topics involve time. This is because this makes it easier to create charts, and use various kinds of formulae in the spreadsheets. If you can think of better or more interesting topics, please do not hesitate to send me a message for feedback.

## 1.4 More on AutoFill

AutoFill is an extremely useful tool, more so than people may realize. It is particularly helpful for larger tables, but it is also helpful for small tables. What makes AutoFill so useful is its flexibility in terms of what can be autofilled.

### 1.4.1 Autofilling a Large Range of Cells

Using the mouse method (dragging the fill handle) is an intuitive method to autofill. However, this method is poorly suited when you need to autofill a column or row of hundreds of items.

For example, if you want to use a spreadsheet to track daily activities (expenses, purchases, etc.), you may want to autofill the dates of an entire year. Doing this by mouse operations is simply not feasible (though possible).

The most powerful way to autofill is to use the autofill dialog box. Let's say you want to create a column of workdays starting at cell C6, and the dates range from January 3, 2005, all the way to December 31, 2005. This is one way to do it:

- Go to cell C6, make it the active cell
- Type in 1/3/2005
- Click the **Edit** menu, then select **Fill**, then select **Series...**
- In the "Series" dialog box:
  - "Series in": select **Columns**
  - "Type": select **Date**
  - "Date Unit": select **Weekday**
  - "Step value": keep the default 1
  - "Stop value": type in 12/31/2005
- Click the "OK" button

You should observe several outcome. First, the spreadsheet is now much longer than what it used to be. Second, the filled dates do not include weekends. Third, the last date is December 30, 2005.

### 1.4.2 Custom Increments

The default increment is 1 unit (day, hour, month, etc.). For example, using the mouse to autofill based on a cell with a value of **January** gives you consecutive months.

This is fine, but it is too restrictive. For example, when you need to create a time table, the time increment can be 30 minutes, it can also be 45 minutes.

Autofill actually lets you specify the difference between autofilled cells. Let's say we are trying to create a timesheet for some lab experiment, and a reading is made every 17 minutes. We want to start the log at 8:15 in the morning, and stop at 5:30 in the evening. The first cell should be D5.

- Go to cell D5, make it active.
- Type in 8:15
- Go to cell D6, make it active.
- Type in 8:32
- Select a range of cells from D5 to AZ5. You can do this using the mouse, or type in D5:AZ5 in the name box.

- Go to the **Edit** menu, select **fill**, then select **series...**
- This time, most of the default settings are fine:
  - “Series in”: select **Rows**
  - “Type”: select **Linear**
  - “Stop value”: type **17:30**
- Click “OK”

You should be able to observe the result. The last cell that has a value should be AJ5, and the time should be 17:19 (5:19 in the evening). This is interesting, because the spreadsheet is extended only as necessary when you specify a stop value. If you specify a range that is too small for the stop value, then autofill only fills up the range, and the stop value may not be achieved.

### 1.4.3 In General...

Excel (as well as Calc) can autofill by example. If you specify two cells, and then use autofill, Excel (and Calc) uses the difference between the two provided cells to fill other cells. This works with numbers, dates and even months. Yes, you can use autofill to fill “every other month”.

Also, note that it is much more handy to use the name box to select a large range of cells, then use **Edit -> Fill -> Series** to fill in the values.

### 1.4.4 Autofilling from One Cell

The result of autofilling from one cell is somewhat interesting. In Excel, if you try to autofill from a cell that has a numeric value, the filled cells have the same value. However, if you autofill from a cell that has a month, a time, or anything that is autofillable that is not a number, it autofills by a default increment.

In Calc, if you autofill from a number, it uses a default increment of 1. This is one of the few differences between Calc and Excel.

## Chapter 2

# Week Two: Formatting and Formulae

### 2.1 Formatting

Chapter two of the textbook is mostly about formatting. While it is a rather long chapter with lots of diagrams and illustrates operation sequences, there is little about the *design* of a spreadsheet.

So, here I am, attempting to write a little about how to design a spreadsheet.

#### 2.1.1 It's a Table!

Of course a spreadsheet is a table! This means a spreadsheet is arranged by rows and columns. A row is a set of cells arranged horizontally, whereas a column is a set of cells arranged vertically.

Logically speaking, in the design of a spreadsheet, you can interchange the use of rows and columns. There is no inherent difference between the two. In practice, however, you need to factor in practical considerations.

#### 2.1.2 Printing

A practical consideration is how the spreadsheet looks when you print it out. Note that in Excel (and in Calc), you can use either the landscape orientation (wider than tall) or the portrait orientation (taller than wide). It goes without saying that the landscape orientation allows more columns and fewer rows.

Generally speaking, because western languages are mostly written horizontally, a cell tends to be much wider than it is tall. This means we can fit very few columns compared to rows on a single page.

What does this mean? This means that in a table, whatever is larger in number should become rows, whereas whatever is fewer in number should become columns.

For example, in a schedule, it is “traditional” that we use columns for day-in-a-week, and use rows for time periods. This is because we only have 5 to 7 days in a schedule, where as we can have many more time periods in a day.

#### 2.1.3 Online Viewing

When a spreadsheet is viewed online, we can apply the same reasoning as when it is printed. Most computer screens are arranged in a landscape mode already, which means you naturally can display more columns. Even so, a spreadsheet often displays many more rows than columns because a cell tends to be *much* wider than it is tall.

### 2.2 Formulae

While the textbook provides a rather hands-on treatment of this subject, I'll try a dryer but more general introduction to formulae.

#### 2.2.1 Cell Referencing

The power of a spreadsheet comes from the ability to compute the value of a cell based on values of other cells. This means that there must be a means by which we can refer to other cells in a spreadsheet.

As already discussed in chapter one of the textbook, each cell has a name. The name is the concatenation of the column name (A, B, C, ...) and the row name (1, 2, 3, ...). This is how we can use the value of a cell in the computation of the value of another cell.

### 2.2.2 The Simplest Formula

Do this experiment.

Open a new spreadsheet. In cell A1, type in the formula as follows:

=A2

Now, read the formula bar when A1 is the active cell. The formula is clearly =A2, but the cell displays 0. This is because by default, cells in a spreadsheet display the *value* of cells, while the formula bar displays the *definition* of the active cell.

Why does A1 have a value of 0? Well, it is because there is nothing in A2! An empty cell has a default numerical value of 0.

Next, make A2 the active cell, and type in a literal numerical value:

23

As soon as you press ENTER or use the cursor keys, you can see that the value of A1 is automatically updated to 23.

This should not be surprising because, after all, we did specify that the value of A1 should be the same as the value of A2.

Now, if you change the value of A2 to 45, the value of A1 should be updated to 45 as well.

### 2.2.3 Moving a Range of Cells

To understand the two types of cell referencing, we need to move the cells defined in the previous section.

To move a range of cells, first select the cells to be moved. In this case, make A1 and A2 both be active. There are many ways to do this. Once these two cells are selected, “cut” the cells. This can be done using the keyboard shortcut **control-X**. It can also be done using the pair of scissors on the tool bar.

Once the cells are “cut”, we want to paste them to column C. Move the mouse pointer to cell C1, then “paste” the cut cells. You can paste using the keyboard shortcut **control-V**, or the toolbar icon that looks like a clipboard.

What you should see now is hardly surprising. Both C1 and C2 have values of 45, while A1 and A2 become blank.

### 2.2.4 But Wait!

Hold on a second here, something does not make any sense!

Recall that cell A1 has a definition of =A2, which means it refers to the value of cell A2. So, after we move the cells, cell C1 should also refer to cell A2, which is now blank. In other words, C1 should have a value of 0 because A2 is blank after the move operation.

Being suspicious, let’s check the definition of C1. Click cell C1 and read the formula bar. Hey, it is now =C2 instead of =A2. But why? Is this a bug?

As it turns out, this is a *feature* called “relative referencing”. When we refer to cell A2 in cell A1, although the formula reads =A2, the spreadsheet understands it as “the value of the cell that is one cell down from the referring cell”.

Now it makes more sense. When we move the cells from A1:A2 to C1:C2, the definition of C1 is still the same “the value of the cell that is one cell down from the referring cell”. What is one cell down from C1? C2, of course!

Relative referencing is useful in most cases. It allows a block of cells that contains referencing among them to be safely moved from one part of a spreadsheet to another part. This ability is also helpful when we use autofill on cells that contain formulae. See Exercise 4 of chapter two in the textbook.

### 2.2.5 Absolute Referencing

Although relative referencing handle the *majority* of referencing, sometimes it is not desirable. Observer the following example:

	A	B	C
1	Student	Raw score	20
2	White, Snow	5	
3	Skywalker, Leia	10	
4	Miss Piggy	12	
5			

We want to “normalize” the score to a scale of 0 to 1 in column C. In other words, the raw score of 5 should be normalized to 0.25, 10 should be normalized to 0.5, and 12 should be normalized to 0.6. The number to normalize to is 20, which is the value of cell C1.

A quick way to do this is simply to use the formula `=B2/20` for cell C2, then use autofill to fill C3 to C4. This will work, but it is not very flexible. If we decide to change the normalizing base number (20), we’ll need to change all three formulae. It’s much more flexible to use C1 to hold the normalizing base value.

Okay, we can try to use the formula `=B2/C1` for cell C2. It works! Unfortunately, it does not work when you autofill to C3 and C4. Why? (But go ahead and try it this way first so you can see the result.)

Because “C1” is a relative reference from C2, it gets changed when we autofill C3 and C4. In other words, when we autofill C3 from C2, the relative reference to C1 becomes a reference to C2 in C3. Likewise, in C4, the divisor becomes C3.

How are we going to solve the problem? Referring to the title of this section, we already know the answer: “absolute referencing”.

Instead of referring to C1 as C1 in C2, we’ll use `C$1` instead. The dollar sign \$ means absolute referring. This means that the row number is now absolute, and it does not change when we move and/or copy the formula.

In summary, the solution is as follows. In C2, we use the formula `=B2/C$1`. Now, we can autofill to C3 and C4. The result should be correct now.

### 2.2.6 Absolute Row and/or Column

Note that a cell reference can be one of the following four combinations:

- relative column, relative row: e.g. C1. This is the most common method.
- relative column, absolute row: e.g. C\$1.
- absolute column, relative row: e.g. \$C1.
- absolute column, absolute row: e.g. \$C\$1.

In general, you should use absolute referencing *only when it is necessary*. If you use absolute referencing too much, you will find that your formulae cannot be moved/copied as flexibly as it should be.

In our example, if we used `$C$1` instead of `C$1`, you will see that we cannot the cells from A1:C4 to other parts of the spreadsheet. Using `C$1`, we can move A1:C4 to D1:F4 without changing the values.



# Chapter 3

## Week Three

We focus on formulae this week. The textbook already explains most of the common functions used in a spreadsheet. This chapter discusses some of the more interesting ways to utilize functions in spreadsheets.

### 3.1 Revisiting Autofill

Autofill is a very useful feature. However, it has its pitfalls. Because autofilled values are just values. If you change the first in an autofilled series, the rest do not get automatically updated. This can be quite annoying when you need to use the same spreadsheet for various scenarios. In fact, our week three homework is a good example. If we want to change the start date of a loan, we have to autofill the dates again in the main table.

This problem can be solved by the use of formula. In other words, we specify the next month using a formula that refers to the prior month. Excel has no built-in function to “increment this date by a month”. However, it does have functions to separate and combine components of a date.

In our example, if we want to increment the date of cell A10 by a month, we can use `=date(year(a10),month(a10)+1,day(a10))`. The functions `year`, `month` and `day` extract the year, month and day (of month) from a given date. The function `date` combines the given year, month and day (of month) to a date. This method works on December, because the `date` function automatically realizes that month 13 is January of the following year.

Using this method and autofill (change of cell references), we can now generate a series of dates that updates automatically when the first date is changed.

Of course, most of the time, it is much easier to compute the value of the following cell. For example, if cell A10 has a value, and we want subsequent cells to leap by 3 from the previous one, then A11 can be specified as `=A10+3`. Next, we can apply autofill to all subsequent cells.

### 3.2 Calc and Excel

If you do not use OpenOffice Calc, you can ignore this section.

In Excel, we use commas to separate parameters in a formula. In other words, we use `=sum(a1,a2,a3)` to find the sum of cells A1 to A3. In OpenOffice Calc, we use semicolons for the same purpose. In other words, we use `=sum(a1;a2;a3)`.

This difference in syntax is fairly minor, and it is converted automatically. In other words, if you open an Excel `.xls` file in OpenOffice Calc, the commas are automatically converted to semi-colons. The reverse happens when you save a Calc `.sxc` file as Excel `.xls`.

Note that this automatic conversion does *not* happen when you read in a comma-delimited text file (extensions `.txt` or `.csv`). If Excel saves a file as a text file, you will need to manually convert all commas to semicolons before the file is opened in OpenOffice Calc.

### 3.3 Conditionals and Logic

The `if` function is a very powerful function because it gives a spreadsheet the ability to make decisions. Nonetheless, this ability to specify logic is still fairly limited.

For those who want to implement more complex logic integrated to a spreadsheet, Visual Basic should be considered. All Microsoft Office products, including Excel, has an interface to utilize Visual Basic to specify programming logic. Visual Basic, for all practical reasons, is a general purpose programming language. You can write programs to handle extremely complex logic.

It is out of the scope of this class to discuss the use of Visual Basic in Excel. Those who are interested in pursuing this should take CISP370 to learn more about Visual Basic.

For OpenOffice Calc, the equivalent programming language is OpenOffice Basic. From what I can review, OpenOffice Basic is fairly similar to Visual Basic.

# Chapter 4

## Week Four

This week, we mainly deal with some formatting options and the internet.

### 4.1 Formatting and Printing

There is not much that I want to add to the textbook.

#### 4.1.1 OpenOffice Calc

For the most part, everything described by the textbook applies to OpenOffice Calc directly. However, for page formatting, this item is under the “Format” menu in Calc, rather than the “File” menu in Excel. I think Calc makes more sense in this case.

#### 4.1.2 Printing to PDF

PDF is the format of choice for document distribution. This is because the viewer only needs to have Acrobat Reader installed. In our case, the viewer does not need to have Excel installed to view a spreadsheet that is exported as a PDF.

To do this in OpenOffice Calc is easy. All the built-in applications of OpenOffice have the ability to generate PDF files directly. This is done by **File -> Export PDF ...** A few options are presented in the dialog box. The default often works fine, but you can tweak the parameters to optimize the result.

Doing this in Excel 2003 is a little harder. If you have Acrobat Distiller installed, then the process is almost the same as printing, except you select the Distiller device instead of a “real printer”.

As it turns out you can also generate PDF without paying a dime. This requires a bit of time and expertise. I’ll outline what you need:

- Ghostscript. This is a free program that interprets the printer language PostScript. It is often used by Unix-like machines to print to non-Postscript printers. This tool can also convert Postscript files to PDF. However, Ghostscript is a command-line program, which means you need to learn how to use the CLI (command-line interface) of Windows.
- GSView. This is also free, but it does not use the GNU General Public License. This is a full GUI program that provides the frontend of Ghostscript. This means you can preview Postscript files, and convert them to PDF without learning the CLI.

Both of these programs can be downloaded from <http://www.cs.wisc.edu/ghost>. Be sure to get the Windows versions. Installing these programs should be fairly simple, just like most other Windows programs. Note that you need to install Ghostscript first, then GSView.

Once you have both programs installed, you still need to install a Postscript printer driver. All versions of Windows come with printer drivers for a variety of Postscript printers. I’ll choose one that is generic, like the “HP LaserJet 4/4M PostScript”. Be sure to choose **FILE:** as the port of this printer. This tells Windows to generate a file instead of sending the output to the printer port. With the exception of very recent versions of Windows, you will most likely need the installation CD of Windows to install this driver.

When you want to generate a PDF file (from any source, not just Excel), this is how you do it:

- When you print, choose the Postscript printer. In our example, it is an “HP LaserJet 4/4M PostScript”.
- Be sure to select “to file” if the port is not FILE:.
- Print. Windows will ask you to enter a file name for the output file. For simplicity, let’s use `C:\output.ps`. It is a convention to use the extension `.ps` for PostScript files.
- After the printing is done, use GSView to open `C:\output.ps`. Obviously, if you choose another file location and/or name, you need to change this accordingly.
- In GSView, select the “File” menu, then “Convert”, in the dialog box, select “pdfwrite” as an output device. The default resolution usually works fine. Click OK to convert. Specify the name of the output PDF file, and you are done!

For those who wish to streamline the process, you can look into using the program “RedMon” to automatically capture Postscript from a fake printer port, and convert to PDF automatically. Personally, this is too much trouble to generate PDFs occasionally.

## 4.2 Spreadsheet vs. Database

(This section is added on 2005/11/10.)

At the end of chapter 4 of the textbook, there are discussions of sorting tables, as well as turning cells into lists, and to use “filters” to limit displayed data. Let’s revisit these concepts, and compare these features with a database program.

### 4.2.1 Rows and Columns

In a spreadsheet, there is no rule to specify how data should be organized in rows and columns. In other words, it is up to the designer of a spreadsheet to determine how data is organized.

However, if you plan to use a spreadsheet to store data in a way that data can be reordered and filtered, then you need to conform to some rules. A column represents a field, while a row represents a record.

Think of each individual fields of a physical form as a cell, and think of each individual form as a row. This means that the fields of all the filled forms become a column. It is, therefore, common to have columns labeled “Name”, “Phone number”, “Employee ID”, “Gender” and etc. Each row, then, specifies information for a particular employee in this example.

### 4.2.2 Sorting

Sorting a table in which each row is a record requires special care. When such a table is sorted, care must be taken so that all cells in a row are moved at the same time! A spreadsheet does not automatically understand that cells of a row are linked. This is one reason why sorting should not be performed by a spreadsheet.

Nonetheless, because sorting a table in a spreadsheet is a common task, it is a feature of Microsoft Excel and OpenOffice Calc. You can, as the textbook describes, sort a table using different columns.

### 4.2.3 List

The name “list” is really arbitrary. In Excel terms, a “list” is a construct that is based on a rectangular section of cells, in which cells across a row compose a record. The significance is that when a list of sorted or filtered, all cells in a row must move, hide or unhide together.

The significance of the concept of “list” is that once a rectangular region is marked as a list, a user needs not worry about “did I select the entire block (for sorting)”. Care still needs to be taken when a list is constructed. After the construction, however, there is no need to worry about the boundary of the region.

Another significance of “list” is the concept of filters. Although a table can be sorted (using any column for sorting), a list offers the unique capabilities of filtering. Using the drop-down box of each column heading, a user can very easily view all the unique values for that column, and choose to display rows that have a specific value for the column.

Filters can also be used for sorting, in increasing order or decreasing order. Once a table becomes a list, sorting it based on a particular column becomes a trivial operation.

The most flexible feature of a list is the use of custom filters. This feature allows a user to tailor make conditions for filtering. The conditions can specify a range, individual values and etc. *However*, a filter in a spreadsheet cannot refer other cells. This may not seem like an important restriction, but it differentiates a spreadsheet from a database.

#### 4.2.4 Applications of Lists

Lists are useful whenever you need to quickly control how rows are displayed in a table. For example, you can have a table that includes all the items sold in a grocery store on a particular day. Each row may include the SKU, item description, location ID, time of sales and register number. Once this table is turned into a list, a user can quickly single out all items sold through a particular register, and sort the entries by time. Or, a manager can determine the sales of a particular item by filtering the SKU.

The examples mentioned are all traditionally database queries. However, they are fairly simple database queries. If the spreadsheet has another table to associate registers and time with employees, a list cannot make the association so that we can filter by employee.

#### 4.2.5 When to switch to a database?

There are two important factors determining when a list in Excel should be implemented by a database.

The first factor is efficiency. Although Excel is fairly efficient with smaller tables, it becomes inefficient with larger tables. The reason is simple: a spreadsheet is not intended to handle the sorting and filtering of large tables. Furthermore, a spreadsheet loads everything into memory. This means as a table grows larger, the *required* amount of memory increases proportionally.

The second factor is flexibility. While Excel lists offer some flexibilities, it still lacks some fundamental features that are standard in a database. For example, the criteria of list filters cannot be computed values. This means you cannot say “compute the total of sales and display the item only if the amount of more than 50”. You can *get around* the problem by adding a new column for “total of sales”, but it is merely a patch that does not work well in general. A database query, on the other hand, can be quite general, and criteria can include computed values based on fields of a row.

Yet another flexibility that is lacking in a List filter is the ability to relate one table to another table. In our example, we may have another table storing “register”, “start time”, “end time” and “employee ID”. This table tells us whom worked at which register at what time. If we want to combine the two tables to tell us “who sold the most amount of tomatos”, we need to “relate” the two tables by register and time.

Excel (or any spreadsheet program) are not designed to do this. There is a way to do this in Excel, but is quite involved (using “vlookup”, “address” and “indirect”). One has to substantially alter a table to prepare for this kind of filter. However, a relational database can handle this with a single query.

### 4.3 Publishing HTML from Excel

Neither Excel nor Calc are the best applications to generate HTML documents. However, both programs do a reasonable job generating HTML documents from worksheets.

What *is* the best way to generate HTML documents? It depends.

#### 4.3.1 Worksheets with Static Data

If you want to publish a worksheet with static data to the web, Excel and Calc are reasonable choices. Although the generated files are not optimal in terms of size, they can be viewed by a variety of web browsers.

It should be noted that the HTML document generated from a worksheet cannot retain formulae. All cells computed by formulae simply retain their value.

#### 4.3.2 Worksheets with Dynamic Data

“Dynamic Data” means data that changes over time. For example, the current federal interest rate is dynamic. Other dynamic data items include stock indices, weather data and expenses.

For example, let us consider a worksheet that computes mortgage payment based on the current interest rate. As an Excel or Calc worksheet, a user can interactively change the loan amount, length of loan and play out different scenario with different interest rates. However, once the spreadsheet is converted to HTML, the HTML document is static.

In short, HTML documents generated by Excel and Calc are not interactive.

### 4.3.3 How do we make an HTML document interactive?

This is out of the scope of this class. However, since we talk about HTML documents, it is only reasonable to explore this topic.

Getting back to our example, if you want to make a web page that is interactive to compute values based on user-specified parameters, you need to use server side and/or client side scripting. Client side scripting is often done in Java or JavaScript, while server side scripting is often done in PHP, Perl, Visual Basic or Java.

It is not an easy step to transition from Excel or Calc to scripting. To learn more about scripting, you should take CISW300, CISW310, CISW410 or CISW420.

## 4.4 Using “Live” External Data

Both Excel and Calc can refer to “live” data from the web. This feature is useful only for computers that are constantly connected to the internet.

A better method to get “live” data is to connect a spreadsheet to a database. However, this is out of the scope of this class. You may want to take an Access class or two to learn how to pull data from a database into Excel.

# Chapter 5

## Week Five

### 5.1 Cell referencing functions

At this point, we have used cell referencing in many contexts. We have also used autofill for various purposes. Unfortunately, there are still certain tasks we cannot do with what we have learned so far.

#### 5.1.1 The Problem

Autofill and copy-and-paste both handle relative cell references. For example, if cell A1 has a formula of =G10, cell A2 will have a formula of =G11, A3 will have a formula of =G12 and etc. from autofill or copy-and-paste.

This works in most cases when we need to refer to cells in a consecutive way. However, occasionally, we need to “hop” cells.

In our week 5 homework, we need to such a thing. Because the “Monthly” worksheet works on monthly computations, annual data is available every 12 rows. That is, A2 in “Annual” refers to A21 in “Monthly”, but A3 in “Annual” refers to A33 in “Monthly”. We *can* just type in these cell references, but it is tedious and prone to error.

To summarize, our problem is that there is easy way to refer to cells that are not consecutive.

#### 5.1.2 Computing the Row Numbers

The first cell in “Monthly” to be referred is on row 21. The next row is 33, then 45 and etc. In other words, the row numbers that we need to use have a general formula of  $21 + 12 \times n$ , where  $n$  is an integer from 0 to 29.

This means we can *compute* these row numbers. In “Annual”, we use column D to computer the row numbers that we need in “Monthly”. This is rather easy. Cell D2 in “Annual” has an initial value of 21, then D3 is =D2+12, and D4 is D3+12, and etc.

Note that we can use autofill or copy-and-paste to create these 30 row numbers.

#### 5.1.3 The “Address” Function

The address function is hardly exciting. It requires a number of parameters to generate the *text* of a cell address.

- Parameter 1: this is the row number. For cell G5, we should specify 5 because G5 is on row 5.
- Parameter 2: this is the column number. Although in a cell address the column is alphabetical, we can easily translate them to numbers. “A” is column 1, “B” is column 2, and etc. As a result, cell G5 has a column number of 7.
- Parameter 3: A number to indicate relative/absolute modes. The default value of 1 is okay for our assignment.
  - 1 for absolute for both row and column
  - 2 for absolute for row only
  - 3 for absolute for column only
  - 4 for relative for both row and column

- Parameter 4: A number to indicate the address form. The default of 1 is okay for our assignment
  - 1 for the usual cell address format (alphabetical column, numeric row)
  - 0 for an alternate address formate using the R1C1 notation for cell A1.
- Parameter 5: this is the name of the worksheet. This is only necessary when we refer to cells that are on a different worksheet (from the cell containing the formula).

Note that the “address” function only returns the *text* of a cell address. It is great for viewing, but it does not actually *get the value* of the cell.

As an example, we can specify the address of cell A21 of sheet “Monthly” by `=address(21, 1,, "Monthly")`. However, recall that we already how column D of worksheet “Annual” listing the useful row numbers of “Monthly”. This means we can use `=address(D2, 1,, "Monthly")` as the formula of cell A2 of “Annual”.

#### 5.1.4 The “Indirect” Function

The “address” function synthesizes the address of a cell, but it does not get the value of the cell. In order to get the value of a cell, we can use the “indirect” function.

The first parameter of the “indirect” function is the text of the address of a cell. But this is exactly the output of the “address” function!

The second parameter of “indirect” is like parameter 4 of “address”. 0 means we use the R1C1 notation, 1 means we use the A1 notation. We need this to be consistent with how we use “address”, the default value of 1 is okay.

Combining this function with “address”, the formula of cell A2 in worksheet “Annual” is now `=indirect(address(D2, 1,, "M`

Once we use “indirect” and “address”, we can now use autofill (or copy-and-paste) to replicate A2 to the range A2:A31.

#### 5.1.5 General Use

The functions “indirect” and “address” are very handy when we need to *compute* cell addresses and get their values. As demonstrated in our assignment, one simple use of “indirect” and “address” is to hop through a worksheet with a fixed stride.

However, these functions have other interesting uses. Such uses (such as multiple indices for sorting and searching) are out of the scope of this class.

# Chapter 6

## Week 8

### 6.1 Database and Spreadsheet

I was asked how to handle situations when data is maintained in spreadsheet tables. Instead of just providing a single answer, let me discuss this matter in more depth.

#### 6.1.1 Spreadsheet versus Database Table

This section is only intended for those who want to know more about the differences between a spreadsheet table and a database table, as well as when to use a spreadsheet instead of a database table.

A spreadsheet is inherently a fancy table. However, a database table is also a table. Besides the similar visualization, however, a spreadsheet is very different from a database table. The following is a simple comparison between the two.

Pros of a spreadsheet:

- A set of powerful functions to perform all kinds of calculations.
- Flexible “Random access” cell referencing, a cell can refer to the value of any other cell.
- Lots of formatting options.

Cons of a spreadsheet:

- A worksheet must be loaded all-or-none. This means large worksheets can exhaust memory quickly. One can say that a spreadsheet does not scale well.
- No indexing. This means it takes a long time to locate a particular cell in a search.
- Difficult to link a table to another. Although `vlookup` and `hlookup` can be used (see later sections), a spreadsheet has very poor support to link tables.

Pros of a database table:

- File-based implementation means size limitation has to do with mass storage (hard disk) space. Scales well to a huge number of entries.
- Indices can be specified and maintained. This means it can be very efficient to locate a particular record based on values of fields.
- Linking tables is easy. Most databases are relational, which means one can define “relationships” among tables. This way, duplicate and redundant data is minimized or eliminated altogether.

Cons of a database table:

- Very limited number of functions.
- Rigid referencing rules. A function can either refer to the value of a field of the same “row”, or be a summary of field values of an entire table (such as maximum, minimal, total, and etc.)
- Formatting is done by reports or forms.

### 6.1.2 When to use a database table?

Based on the pros and cons of database tables, they are most suitable for situations that satisfy the following criteria:

- The number of entries can be very large.
- Tables are linked.
- Search operations must be efficient.

Generally speaking, any time you want to store records that are formatted similarly, you should consider the use of a database table.

### 6.1.3 Combining a spreadsheet and a database table

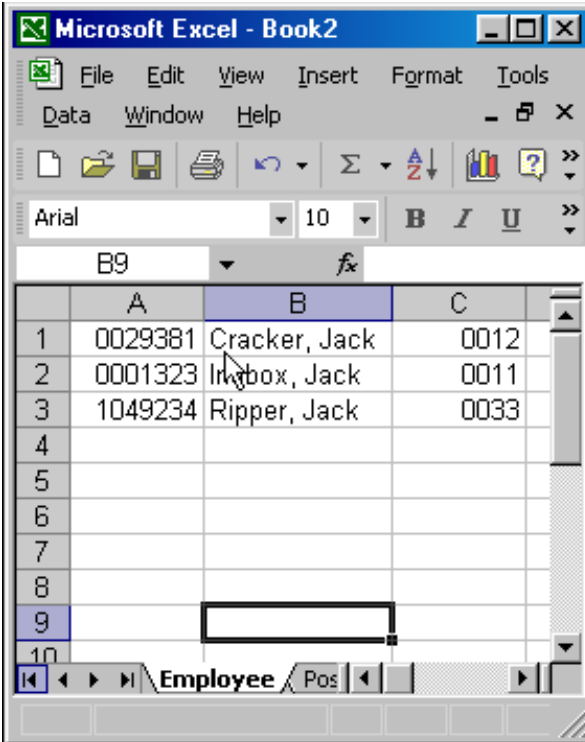
Sometimes, you need to store data in a database table, but you may want to perform complex computation or plot graphs with the data. This is when you need to refer to a database from a spreadsheet.

Both Excel and Calc have extensive support to import data from database tables and queries. This topic is well beyond the scope of CISA315, but those who are interested can read the appropriate topics in the online help of Excel and Calc, or come visit me during my office hours.

### 6.1.4 vlookup and hlookup

You can perform a limited amount of table linking in Excel and Calc using the function `vlookup` and `hlookup`. To understand how these functions work, let us use an example.

First we assume there is a table that lists employees and their personal data. Column A is the employee ID, and Column C is the job description ID.



	A	B	C
1	0029381	Cracker, Jack	0012
2	0001323	Irishbox, Jack	0011
3	1049234	Ripper, Jack	0033
4			
5			
6			
7			
8			
9			
10			

Next, we have a table that maps job description ID to job description text. Note that this table is in another sheet.

The screenshot shows a Microsoft Excel window titled 'Microsoft Excel - Book2'. The menu bar includes File, Edit, View, Insert, Format, Tools, Data, Window, and Help. The toolbar contains various icons for file operations and calculations. The active cell is B8, which contains the text '0033 Drone'. The worksheet has columns A, B, C, and D, and rows 1 through 10. The data in the worksheet is as follows:

	A	B	C	D
1	0001	Supervisor		
2	0010	Vice President		
3	0011	Chief Executive Officer		
4	0012	Chief Financial Officer		
5	0030	Peon		
6	0031	Farmer		
7	0032	Worker		
8	0033	Drone		
9				
10				

Last, we want to generate a report to list employee name with job description.

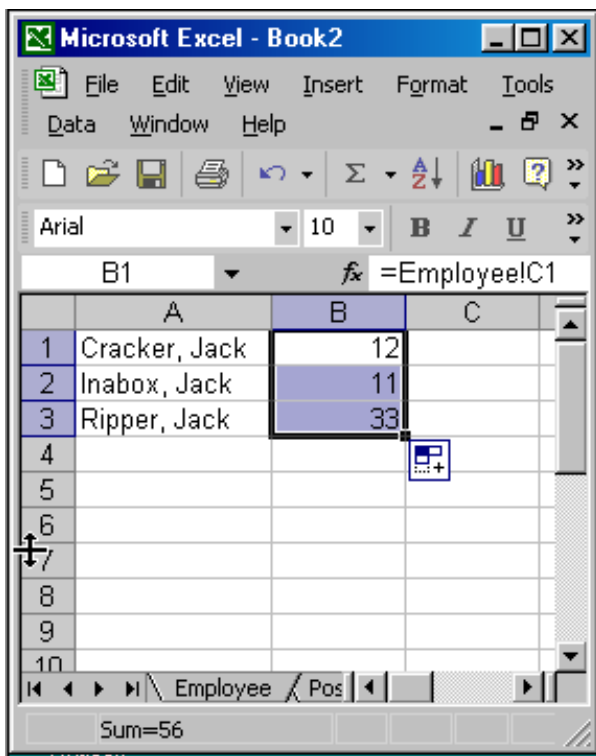
The screenshot shows a Microsoft Excel window titled 'Microsoft Excel - Book2'. The menu bar includes File, Edit, View, Insert, Format, Tools, Data, Window, and Help. The toolbar contains various icons for file operations and calculations. The active cell is A1, which contains the formula '=Employee!B1'. The cell displays the text 'Cracker, Jack'. The worksheet has columns A, B, C, and D, and rows 1 through 10. The data in the worksheet is as follows:

	A	B	C	D
1	Cracker, Jack			
2				
3				
4				
5				
6				
7				
8				
9				
10				

Our first step is to get the employee names from worksheet "Employee". After we get the first one, we just use autofill to get the other three.

Next, we change the column width to accommodate the entire width of names. And, here comes the most challenging step. We need to somehow use the job description ID of worksheet "Employee" to lookup the rows in worksheet "Post".

We'll do this in steps. First, we just want to refer to the job ID of Employee.



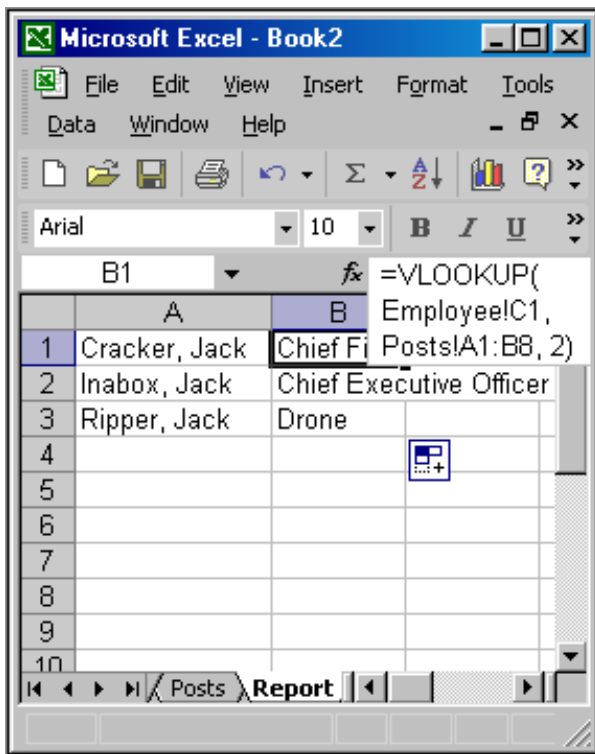
Next, we need to use this value to look up the table in worksheet “Post”. According to worksheet “Posts”, column A lists the job IDs, and column B is the description. What we want to do is to search A1:B8 in worksheet “Posts” using values from column C of worksheet “Employee”.

One function that can do this is `vlookup`. In our case, in worksheet “Report”, cell B1 has a formula as follows:

```
=vlookup(Employee!C1, Posts!A1:B8, 2)
```

Let’s explain the parameters:

- First parameter. This specifies the value that we are looking for. In our example, it is `Employee!C1` because this cell reference refers to the employee job ID.
- Second parameter. This specifies where to lookup and find the value. This parameter specifies two or more columns. The first column (hopefully) contains the value specified by the first parameter. In our example, the parameter is `Posts!A1:B8` because that subtable contains the job ID as well as the job description.
- Third parameter. This number specifies the column of the subtable specified by the second parameter that returns the value of the function. In our example, this parameter is 2 because column B of the subtable specified by the second parameter contains the job description.
- Fourth parameter. This parameter is optional. It specifies whether the lookup is exact or not. The default value is 0. If the value is non-zero, then the table specified by the second parameter must be sorted, and the function attempts to return the best match. We leave this parameter unspecified in our example to use the default value of 0.



A companion function, `hlookup` does a similar job, except it works in a “horizontal” format instead of “vertical” format.

## 6.2 Goal Seek and Solver

Goal Seek is available in Excel and Calc, whereas Solver is only available in Excel. You can look at Solver as a more generalized version of Goal Seek.

The textbook demonstrates and explains both features very well, so there is no need to repeat any material here. It is, however, worthwhile to point out more about what these two functions can be utilized in our “daily activities”.

### 6.2.1 Goal Seek

Goal seek is useful when a value depends on another value. The relationship can be simple, as in Celcius to Farenheit conversion, or it can be somewhat complex, as in interest rate versus monthly payment.

For those of us who can handle algebra, goal seek is not always necessary. One can solve an equation symbolically to express one variable using another variable. For the rest of us, goal seek works like a charm without any need of algebra!

Note that sometimes it is not easy to solve an equation symbolically. This means goal seek actually has value even for those of us who can handle algebra.

For most of us, goal seek can be used to convert units. If you now how to convert from unit *A* to unit *B*, goal seek helps you convert from unit *B* back to unit *A*.

### 6.2.2 Solve

Solve is powerful, and it is not available in Calc.

Solve can do everything that goal seek can. Solve is *much* more powerful than goal seek. As a textbook already describes, solve can handle many variables, and it can be used to maximize, minimize or solve to a particular value.

How is Solve useful in our daily activities? I can think of many scenarios. Let’s just focus on one right now.