APPENDIX

A OO Calc 3 Features and Functions

INTRODUCTION

The main features and functions of Apache OpenOffice (OO; formerly, OpenOffice.org or OOo) Calc 3 are introduced here. Go through this appendix to start learning the simple technical material. To effectively and instinctively apply them, you have to practice modeling. The notes and exercises in the main text of this book are specifically provided for that purpose. As you do the modeling, remember to periodically review the list below and identify the items you should know but are still unclear. Be sure to check out **Help** in Calc and **Help** in OpenOffice.org **Basic Editor**.

<u>Elementary</u>

Files

- Default directory and some file management, Properties
- New, Open/Close
- Save, Save as, *.ods, *.ots, *.sxc, *.stc, *.xls, *.xlt, *.csv, *.html, *.pdf

Rows, Columns, and Sheets

- Insert/Move/Hide row/Column/Sheet, Change column width, Change row height
- Rename Sheets (Tab name)
- Cells and Ranges, Notes
- Formulas (+-*/^, calculation order of A1 + 2*B1/C1^2)
- Editing formulas, Detective
- Number formats (general, fixed, scientific (E), $, and %), Date/Time formats
- Font (size, type, color), Alignment, Border, Patterns, Protection (locked, hidden)
- Buttons (bold, italic, underline, 0.00, $, and %)
- Character strings and string arithmetic ("=A"&B6&"B")

Commands

- Cut/Copy/Paste, Absolute/Relative/Mixed referencing (A1, $A$1, $A1 and A$1)
- PasteSpecial (Formats, Text, Numbers, Transpose, Operations, Shift), Fill, Undo

Tools/Options/General File (password) protection, Macro Security

Tools/Options/View: Gridlines, Row and Column Headers, Formulas, Page Break

Drawing: Draw, Group, Order, Rotate/Flip, Auto-shapes, Arrows, Alignment, Anchor

Intermediate
Functions: Function wizard, SUM, COUNT, AVERAGE, MAX, MIN
Functions: IF, SUMIF, COUNTIF, SUMPRODUCT
Data: Data/Multiple Operations; Tools/Goal Seek, Scenarios, Solver
Charts: Types (Bar, Line, XY), Formats, Series (add, delete), Legends, Axis, Titles
View: Windows, Menus, Toolbars, New (Multiple) windows, Freeze pane, Split
Customization: Tools/Options, View/Toolbar
Page layout: Set/Clear Print Area, Print Preview, Header/Footer, Fit to Page
Options: Formula calculation (Automatic, Manual, Iteration)

Advanced
Data: Sort, Validity, Form, Filter, DataPilot
Functions: LOOKUP, VLOOKUP, HLOOKUP, MATCH, INDEX, INDIRECT, CHOOSE, OFFSET, ADDRESS
Functions: NORMDIST, NORMSDIST, NORMINV, NORMSINV, EXP, LN, BINOMDIST, CRITBINOM, LINEST
Functions: Array function and formula, FREQUENCY, RAND, RANDBETWEEN
Functions: DSUM, DCOUNT, DAVERAGE, DMAX, DMIN

Add-ins: DataForm
Form: Button, Group, Option, Check, Spinner, Slider, Control
Macros: Sub and Function procedures, User-defined functions (CellFormula, ShowFormula)
Protection: Sheet, Document
Developer: Basic Editor (Alt+F11), Basic Help, Object Catalog

FURTHER REFERENCES

- Leete, Finkelstein and Leete, 2004. *OpenOffice.org for Dummies*, Wiley.
- Leong, 2014. LibreOffice and OpenOffice Web Resources
  - http://isotope.unisim.edu.sg/users/tyleong/SpreadsheetModeling.htm#Calc
  - http://dl.dropboxusercontent.com/u/19228704/SpreadsheetModeling.htm#Calc

OO 3 CALC PRIMER

Knowing the basic features and functions of the spreadsheet well is paramount before you can ever be competent in spreadsheet modeling. A short review is done in this appendix to help first-time users learn it faster, and experienced users to learn it better. There is hardly any book written on OpenOffice and Calc. However, online tutorials and help are very available. Moreover, Calc is largely similar to Excel 2003 in layout and features, and there are many excellent books in the market that deal directly with Excel 2003, which you may want to refer to as additional help and draw parallel inferences. In this appendix, *Note 1* through *Note 4* are essentially for first-time users. If you have used spreadsheets for some time now, you may skip these notes and proceed straight to *Note 5*.

NOTES

Note 1: SPREADSHEET WINDOW The Calc spreadsheet file is called a **SpreadsheetDocument** and the worksheets it contains are called **Sheets**. These correspond to **Workbook** and **Worksheets** in Excel. Both naming conventions are used interchangeably in this book. The main text would adopt the Excel convention, and where possible, we will use the OOo terms in their context. By default, a new spreadsheet document contains three sheets (aptly named by default as *Sheet1*, *Sheet2*, and *Sheet3*). You can easily insert or delete sheets, which is similar to adding pages to or deleting pages from an exercise book. These sheets can also be renamed and their tabs can be colored by you. Right-click on the worksheet tab. You will see, in the side menu, the options to do this and more.

There are rectangular spaces on each sheet, each referred to as a **Cell**, into which you can enter values or formulas. These values can be text, numbers, or logical values (i.e., TRUE or FALSE). The mathematical formulas that you write into the cells are used to compute any desired results. This is the most formidable part of spreadsheets since their formulas can be made to link data and values from all other cells.

Cells, arranged in a grid, are identified by their row and column positions. There is only one way to identify a cell: the default **A1**. (The **R1C1** reference alternative style in Excel is not yet available in Calc. For example, D2 refers to the cell in row 2 of column D and equivalently R2C4 refers to the same since column D is the 4th column.) Alphabet column labels go from A to Z, then AA to AZ, BA to BZ, and so on to IV. These correspond to columns 1 through 256.

Trivia Fact

Q: How many rows and columns are there in a Calc sheet?

A: Calc has 65,536 rows and 1,024 columns. The numbers look rather odd because they are numbers expressed in powers of 2. There are actually few situations where these limits are ever reached and the spreadsheet document would usually work rather slowly when the sheets get to be so large.

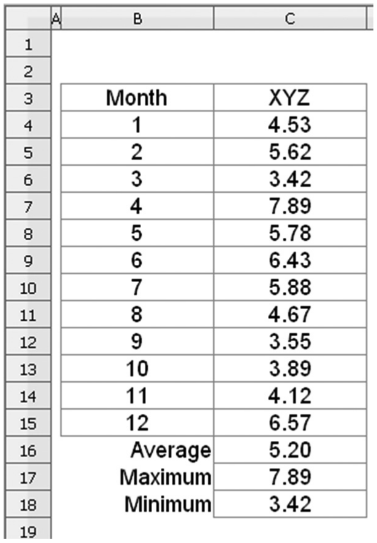
To enter a formula into a cell, begin with the **=** sign. For example, to compute 1 + 2, just type =1+2. Once you hit the return key, the result will immediately appear in the cell as the value 3. Note that Calc does not by default show the formula expression in the cell itself, but rather the result of the formula. You can view in the **Formula** bar the formula for the cell where the cursor resides. Alternatively, you can change the sheet option to view formulas (instead of values) in all the cells of the sheet. This can be done by keying **Ctrl +** ` (typically located just left of the **1** key at the upper-left corner of your keyboard). To switch back to the default option, key **Ctrl +** ` again. If this does not work, change your keyboard settings and restart Calc, Windows, (or MacOS) or both.

Quick Tip

The usual mathematical operators, namely multiply (*), divide (/), add (+), subtract (–) and power of (^), are applicable in Calc. Do note that symbols used in Calc for multiplication and division are not × and ÷. The order of calculation still follows the usual mathematical *BODMAS* (bracket, of, divide, multiply, add, subtract) convention. Only round brackets ( ) can be used to group and alter the calculation order. Other types of brackets are not permitted in Calc. Also, brackets do not automatically indicate multiplication and so the multiply (*) operator has to be inserted between brackets where applicable.

Of course, Calc's computing ability is more than just adding two numbers. It has a large set of functions, and together with its (absolute, relative, and mixed) cell referencing system, you can construct complex mathematical and other kinds of formulations in a small collection of cells and massively replicate them with little effort to other cells to complete your model. In fact, you should only be working with **Cell References**, and not data values, in formulas. This will be further illustrated and discussed in *Note 2* and *Note 3*.

Note 2: COMPUTATION Let us try to compute the average, maximum, and minimum of stock prices for company XYZ for all the months in a year, as given in *Figure A-1*.



	A	B	C
1			
2			
3		Month	XYZ
4		1	4.53
5		2	5.62
6		3	3.42
7		4	7.89
8		5	5.78
9		6	6.43
10		7	5.88
11		8	4.67
12		9	3.55
13		10	3.89
14		11	4.12
15		12	6.57
16		Average	5.20
17		Maximum	7.89
18		Minimum	3.42
19			

Figure A-1

Stock Prices for Company XYZ

We will compute the average stock price in cell C16. You would have learned in high school that the average stock price is equal to the sum of all the 12 months' stock prices divided by 12. If you still remember this, your math teacher will certainly be proud of you. To make it easier for you, Calc provides the **AVERAGE** function to do the same arithmetic calculations. Now, enter into cell C16 the formula =AVERAGE(C4:C15) and key **Enter**. The result is exactly what we want!

Quick Tip 1

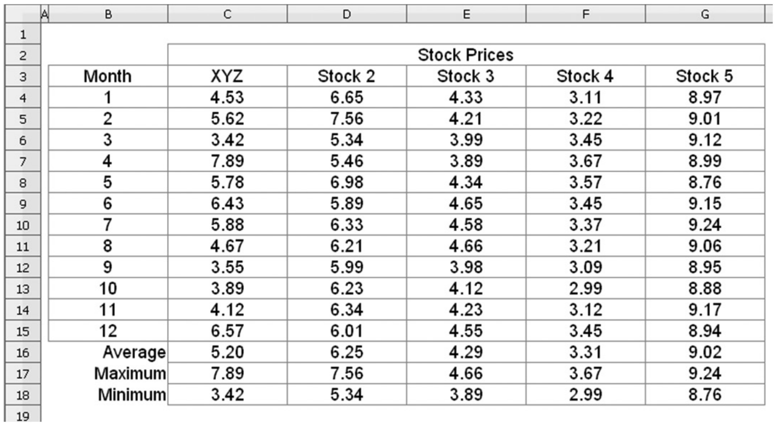
While entering the formula, you can select cell C4, hold down the mouse's left button and drag it to cell C15 instead of typing the cell reference C4:C15 in the formula. This is quicker and less prone to error. Indeed! Calc has such a useful function! Yes, and it has about 370 or more. To view the other functions, you can either go to Calc **Help**, or simply click on the *fx* button in the **Formula** bar. In Calc, the arguments in any function are separated by semicolons (;) and not commas (,) as in Excel.

Now, continue to compute the maximum and minimum stock prices in cells C17 and C18, respectively. What Calc functions should you use? You have guessed it, the **MAX** and **MIN** functions. Enter into cells C17 and C18, the formulas =MAX(C4:C15) and =MIN(C4:C15) to obtain the results.

Let us imagine that we want to analyze the average, maximum, and minimum prices for five different stocks, as shown in *Figure A-2*. We do not need to type the same functions five times. Instead, we can perform a **Fill** operation. To begin, we can just select the three cells with formulas, namely C16 to C18, as a group, bring the cursor to the lower-right corner of the group, and drag it to the next four columns (i.e., to column G). Magically, the average, maximum, and minimum stock prices for the next four stocks will be computed.

Figure A-2

Stock Prices for Five Stock Counters



	A	B	C	D	E	F	G
1							
2					Stock Prices		
3		Month	XYZ	Stock 2	Stock 3	Stock 4	Stock 5
4		1	4.53	6.65	4.33	3.11	8.97
5		2	5.62	7.56	4.21	3.22	9.01
6		3	3.42	5.34	3.99	3.45	9.12
7		4	7.89	5.46	3.89	3.67	8.99
8		5	5.78	6.98	4.34	3.57	8.76
9		6	6.43	5.89	4.65	3.45	9.15
10		7	5.88	6.33	4.58	3.37	9.24
11		8	4.67	6.21	4.66	3.21	9.06
12		9	3.55	5.99	3.98	3.09	8.95
13		10	3.89	6.23	4.12	2.99	8.88
14		11	4.12	6.34	4.23	3.12	9.17
15		12	6.57	6.01	4.55	3.45	8.94
16		Average	5.20	6.25	4.29	3.31	9.02
17		Maximum	7.89	7.56	4.66	3.67	9.24
18		Minimum	3.42	5.34	3.89	2.99	8.76
19							

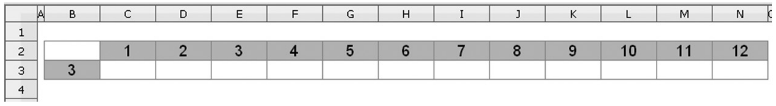
Fill, perform **Copy** and **Paste** operations, will automatically duplicate the formulas. It will copy the first set of selected cells and paste them over cells in other locations. In our example above, the formulas in the destination cells take their inputs not from the first set of stock prices, but the stock prices of cells directly above them. This phenomenon is known as **Relative Referencing**. The Fill operation can also be performed on cells that contain data values instead of formulas like what we have demonstrated so far. This will be discussed later.

Quick Tip 2

Dragging the selection one column to the right will cause an increment in the column letters in the formulas by one letter. Similarly, dragging the selection one row down will increment the row numbers in the formulas by one.

What if you do not wish the column letters or row numbers to be changed when performing a Fill operation or Copy and Paste operations? Well, you can switch off the relative referencing to use **Absolute Referencing** or **Mixed Referencing** instead (see next section). **Cell Referencing** is one of the most, if not the most, useful features in electronic spreadsheets. It is also one of the most difficult to understand. Most people find it confusing when they are first introduced to it. However, you will benefit tremendously from it if you make an effort to understand it. Once you have learned it, you will never go back to the old way. Let us move on to *Note 3*.

Note 3: CELL REFERENCING Cell referencing, that does not permit column or row, or both to change when the cell formulas are copied elsewhere, is done using the $ sign. To disallow column change, just prefix $ to the column letter in the cell reference; to disallow row change, do the same to the row number in the cell reference. You can also do both concurrently. That is, =$A$1 will stay as =$A$1 when the cell formula is copied to any other cell in the sheet. A cell with =$A1 will only have its row number changed in the destination cell formula when copied to another cell in a different row; a cell with =A$1 will have its column changed in the destination cell formula when copied to another cell in a different column. Let us try to understand this concept further by working through the following example. Prepare a multiplication table as shown in *Figure A-3*. How can we apply the **Fill** operation to compute the results effectively?



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2			1	2	3	4	5	6	7	8	9	10	11	12
3		3												
4														

Figure A-3

Multiplication Table for the Number 3

In cell C3, type the formula =B3*C2. If you simply drag this cell selection to the cell to the right of C3, the results will be incorrect. This is because B3 will be changed to C3 in the formula in cell D3, to D3 in cell E3, and so on. To keep B3 as it is, modify the formula to =$B3*C2. Now, drag this cell selection to the right again. You will see that $B3 remains as $B3 in all cells. The $ before B indicates that column letter B must not be changed in a Fill operation.

Quick Tip 1

$A$1 refers to **absolute referencing** where the same value in cell A1 will be applied after a Fill operation, regardless of dragging across rows or across columns, because the $ sign preceding column letter A restricts increments in the column letter, while the $ sign preceding row number 1 restricts increments in the row number.

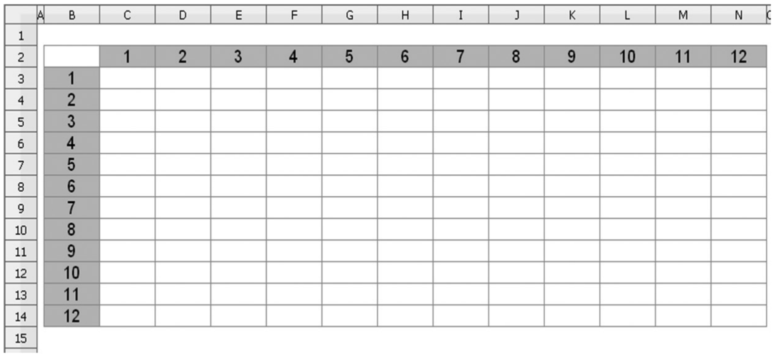
$A1 refers to mixed referencing, where only the column letter is restricted from increments. A$1 is the other mixed referencing where only the row number is restricted from increments.

The **Shift**+**F4** (or **Shift**+**fn**+**F4** in MacOS) key in the top row on your computer keyboard is a toggle key to facilitate changing the cell reference to the different forms (A1, $A$1, A$1, $A1). Just hit the key after you have typed a cell reference to cycle through the set of cell reference forms.

As a further challenge, let us prepare a bigger multiplication table as shown in *Figure A-4*. Can you figure out where the $ signs should be added in the formula?

Figure A-4

Multiplication Table for 1 to 12



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2			1	2	3	4	5	6	7	8	9	10	11	12
3		1												
4		2												
5		3												
6		4												
7		5												
8		6												
9		7												
10		8												
11		9												
12		10												
13		11												
14		12												
15														

In this multiplication table, enter into cell C3 the formula =$B3*C$2. The $ sign preceding column letter B ensures that the reference to column B remains static, while the $ sign preceding the row number 2 ensures that the reference to row 2 remains static. Try it out. If it works, expand the multiplication table to 50 by 50 or even 200 by 200 to see if it is still okay.

Quick Tip 2

All cell formulas should only contain cell references, operators and functions. There should not be any numerical, text, or logical value in them. Formulas therefore show the relationships between cells, which in turn can contain values or formulas. This makes updating the input data values explicit and clearer to the user.

To see a visual representation of the cell relationships, you can click on the cell of interest and then click the cell formula in the **Formula** bar. The cell references in it will then become color coded, and boxes whose colors match those used in the spreadsheet will frame the corresponding cells referenced in the formula. You can even modify the cell formula by dragging or changing the shape of these boxes.

An alternative approach is to use Calc's **Detective**. This is found in the **Tools/Detective**. Click **Trace Precedents** or **Trace Dependents** to visually show arrows linking the cells.

Hopefully, the examples used in this note have helped to illustrate the usefulness of cell referencing and clarified its correct use. We are now ready to learn more advanced features of Calc. As you model and analyze more complex problems, you will need to learn these additional spreadsheet features to be proficient and effective. In the following few notes, we will cover some of the more useful advanced features.

Note 4: FORMATTING The appearance of a cell can be altered by changing its row height or column width. Rows or columns can be moved, deleted, or inserted to alter the arrangement of cells in any spreadsheet model. The cell references in the affected formula will automatically change without altering the underlying logic in the cell formulas. Most of the formatting actions can be done intuitively using the options available in the **Format** menu or toolbar as shown in *Figure A-5*. The options available include changing the font type, style, size, alignment, cell shading, and borders.

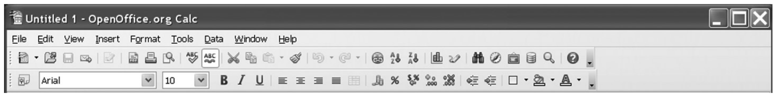

Untitled 1 - OpenOffice.org Calc
File Edit View Insert Format Tools Data Window Help
Arial
10
B I U

Figure A-5

Options in the Formatting Toolbar

The value in a cell is automatically formatted according to the way you entered it. If you enter 0.3501, then it would appear as such with the same number of decimal places. If the intended entry is 35.01%, it may be better to enter as such. The decimal value entered can of course be formatted to be a percentage by clicking the **%** icon in the menu. Other icons allow you to change the number of decimal places displayed, add commas, or add currency prefix. You can also right-click on a cell and then select **Format Cells** to open the dialog.

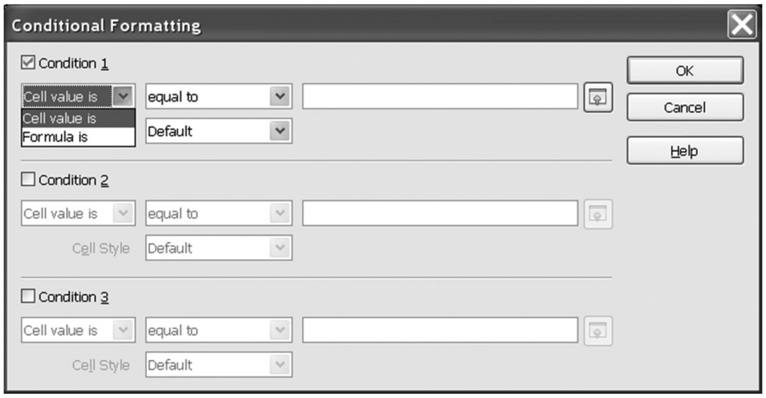
No matter how a value is reformatted, the value stored in the cell remains unchanged. In particular, a cell with the value 35.01% displayed is actually stored as 0.3501 and will be computed as 0.3501 and not 35.01 in formulas. There is therefore no need to divide by

100 to get the proportion value. You, however, should exercise caution when reducing the number of decimal places displayed as it may lead to calculations being misinterpreted. As a convention, it would be good to keep the number of decimal places the same for all values of the same variable and possibly across similar variables in the whole spreadsheet.

The format of a cell may be changed conditionally. The condition may be the state of value in the cell to be formatted or based on a logical formula. The logical formulas topic is discussed in *Note 10*. You can select **Conditional Formatting** in the **Format** menu and then select the options in the dialog shown in *Figure A-6*. The rules based on either values or logical formulas can be specified there. Calc allows only up to three rules and they must work in succession, stopping at the first rule that is satisfied. Select **Cell Value Is** to change the format based on the cell values or select **Formula Is** to specify the desired logical formula. **Cell Style** option enables you to specify the displayed format when the values or formula conditions are met. You can define your own style. This can be done, but earlier on, in **Format/Styles and Formatting**.

Figure A-6

Conditional Formatting

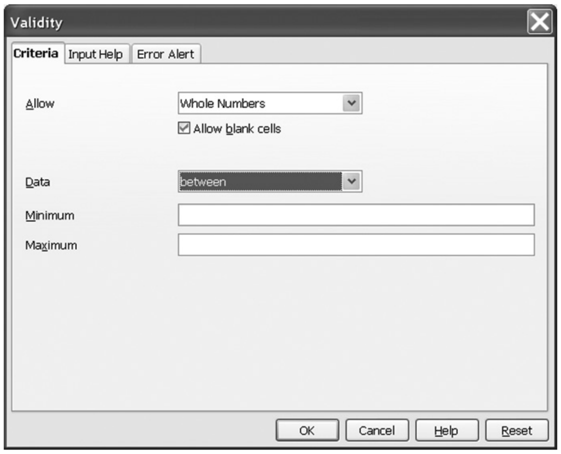


Note 5: DATA VALIDATION AND LOOKUP

Validity

Whereas cells with formulas may be protected, input cells need to be available for users to enter the required values. These cells can therefore be subject to entry mistakes. To offer some assurance that the input cells are correctly used, you can apply **Validity** to check the inputs as they are entered. Data validation checks the variable type and value range for applied cells.

To set the validation for a cell, select the cell and then click **Data/Validity**. In the dialog as shown in *Figure A-7*, select the variable type in **Allow**, and then the conditions in **Data**. Enter also the lower and upper limits, where applicable, of the values.



Allow: Whole Numbers
☑ Allow blank cells
Data: between
Minimum:
Maximum:

Figure A-7

Validity

Ever think of allowing users to enter inputs by letting them choose from a list of permitted values? Calc's Validity also allows you to prepare a list of given values to be assigned to a cell. Once assigned, a drop-down list will appear to allow the user to make the selection. This helps to minimize mistakes in data entry.

Let us construct a drop-down list to allow a user to select *gender* (*female* or *male*).

1. Select the cell where the drop-down list is to appear.
2. From the **Data** menu, select **Validity**.
3. In the **Settings** panel, in the **Allow** drop-down list, select **List**.
4. In the **Settings** panel, in the **Entries** input area, enter each value or text in separate lines: *Female* and *Male*.

Quick Tip 1

For more flexibility, you may want to put the source input values as a group in a separate vertical or horizontal cell range, and in step 4 above, set the cell range reference or make the selection instead of entering the values. Unlike Excel, this feature is not yet available in Calc.

For more complex data validation in Excel, you can use the **Custom** option in **Allow** of its **Data Validation** dialog to set logical formulas to govern the cell's data validation. This option is however not available in Calc, at least not at the time of publication of this book.

Table Lookup

One of the most useful features of Calc is its ability to find the corresponding result for a lookup value in a table. For example, at a game stall, if you have shot three or fewer targets, you win yourself a bronze medal, and if you have shot four to seven targets, you win a silver medal, and finally, if you have shot eight to ten targets, you win a gold medal. The stall owner needs to look up the number of targets shot in a table to determine what kind of medal to award. This example seems trivial because there are only three categories of results. Imagine having to manage a large number of categories, with dynamically changing lookup values. The values in the table themselves may even be dynamically changing. The table lookup feature will prove to be extremely helpful then. Here are some lookup functions:

LOOKUP(*lookupValue; lookupRange; valueRange*)

- Returns the value in *valueRange* that has the same relative position that *lookupValue* has in *lookupRange*. The values in *lookupRange* must be in ascending order. If LOOKUP cannot find *lookupValue* in *lookupRange*, it matches the largest value in *lookupRange* that is less than *lookupValue.*
- The square brackets around *rangeLookup* indicate that it is an optional input. This square bracket convention applies to all other functions as well.
- For example, LOOKUP(6;A1:A6;B5:B10) with (3, 5, 7, 9, 11, 12) in A1:A6 and (2, 4, 6, 8, 10, 12) in B5:B10 would return the value of 4, as the value 5 in A1:A6, the largest value less than or equal to 6, is in the second position, and the second value in B5:B10 is 4.
- LOOKUP(RAND(); *lookupRange*; *mid-binValueRange*) returns a random value, according to the distribution of the values listed in the table defined by the mid-bin value and lookup ranges. This is one of the ways you can generate random values from a given distribution frequency table. Refer to *Note 6* for comments on the **RAND** function and other ways to generate random values.
- LOOKUP is best applied when the *lookupRange* and *valueRange* are not residing in the same table or are not of the same (row or column) orientation.

VLOOKUP(*lookupValue; tableArray; columnIndexNum;* [*rangeLookup*])

- Searches for *lookupValue* in the left-most column of *tableArray*, and returns the value in the same relative row position in the column identified by *columnIndexNum*. The *columnIndexNum* is a positive integer and *rangeLookup* is a logical value. The square brackets around *rangeLookup* indicate that it is an optional input. This square bracket convention applies to all other functions as well.
- When *rangeLookup* is TRUE (or omitted), the function will look for an approximate match for *lookupValue* in the left-most column, similar to the way the LOOKUP function works. For approximate match lookup, the values in the left-most column of *tableArray* must be in ascending order.
- However, when the *rangeLookup* is FALSE, an exact match for *lookupValue* in the left-most column is searched, and returns #N/A if none is found. The first matched position is returned if there are multiple possible matches.
- VLOOKUP is best applied when the *lookupValue* is compared with values in the left-most (i.e., first) column of *tableArray*.

MATCH(*lookupValue*; *lookupRange*; [*matchType*])

- Returns the relative position that the *lookupValue* is in *lookupRange.*
- If *matchType* is 0, MATCH looks for an exact value of *lookupValue* in *lookupRange* and returns #N/A if none is found. The first matched position is returned if there are multiple possible matches.
- If *matchType* is 1 or omitted, MATCH looks for the largest value less than or equal to the *lookupValue*. The *lookupArray* must first be in ascending order: …–1, 0, 1, …, A–Z, FALSE, TRUE.
- If *matchType* is –1, MATCH looks for the smallest value greater than or equal to the lookupValue. The *lookupArray* must first be in descending order. Only MATCH can deal with the approximate matches of a descending sorted *lookupRange*.

INDEX(*tableArray; rowIndex; [columnIndex]*)

- Returns the value found in the relative *rowIndex* and *columnIndex* positions in *tableArray*.
- When *tableArray* is a single column of values, then *columnIndex* is not needed.
- However, when *tableArray* is a multiple-row and multiple-column matrix, then both *rowIndex* and *columnIndex* are needed to identify the value.

Quick Tip 2

The clever use of MATCH within INDEX allows us to identify values using dynamically changing *rowIndex* and *columnIndex*, where the *rowIndex* and *columnIndex* are results obtained from using MATCH twice.

Note 6: RANDOM NUMBERS AND RESAMPLING

Random Number

Random numbers are useful, especially in Monte-Carlo simulations, where variations of input values are needed. Two functions are available in Calc to generate random numbers, one for continuous values and the other for discrete values.

RAND( ) returns with equal probability a random value in [0,1), i.e., between 0 and 1, inclusive of 0 but not of 1. The brackets, with nothing inside, are needed when you enter the function into a cell. This function can be used in formulas to generate random values of other probability distributions and interval values. For example, to generate a random real value uniformly distributed between 5 and 9, we can use =5+RAND()*(9–5). So when RAND is 0, it returns 5; when RAND is almost 1, it returns a number that is almost 9 and when RAND is a value between 0 and 1, it generates the interpolated value between 5 and 9 with uniform probability density.

RANDBETWEEN(*loNum, hiNum*) returns with equal probability a random integer in {*loNum, loNum* + 1, …, *hiNum*}, i.e., integers between *loNum* and *hiNum*, *loNum* < *hiNum* being integers. For example, RANDBETWEEN(1, 6) can be used to simulate the toss of a die. This function is part of Calc; in Excel, it is only available when the **Analysis ToolPak** (a standard **Add-in** in Excel) has been activated. Alternatively, formula =INT(RAND()*((*hiNum* – *loNum*)+1)) + *loNum* would also give the same result. That is, you can also simulate the toss of a die using INT(RAND()*6) + 1.

Quick Tip

Calculation Options is found in **Tools/Cell Contents**. When the **AutoCalculate** option is checked (by default), any recomputation or data entry into the sheet will cause the random function to generate a new random number. To restrict this, especially when it is rather disruptive during model construction, you can uncheck this option. Key **F9** whenever you wish to regenerate a new random value.

Resampling Data

Data resampling is important when we want to generate random data points from a handful of sample data points. Calc has some useful functions that can be used to perform data resampling. (Refer to the *Data Simulation* exercise and the *Resampling* tool in Chapter 6 for more details.)

SMALL(*valueRange*; *k*)

- Returns the *k*th smallest value, with *k* being an integer, in the data set specified by *valueRange*.
- For example, SMALL(A1:A20;1) returns the smallest value in A1:A20 (same result as MIN(A1:A20)), SMALL(A1:A20,2) returns the second smallest value, SMALL(A1:A20,3) returns the third smallest value, and so on. When used against a column of running serial numbers 1, 2, 3, …, it can be used to sort a set of numbers in ascending order.
- SMALL(*valueRange*, RANDBETWEEN(1, *n*)) resamples with equal probability the *n* sample data values stored in *valueRange*.

LARGE(*valueRange*; *k*)

- Returns the *k*th largest value, with *k* being an integer, in the data set specified by *valueRange*.

- For example, LARGE(A1:A20;1) returns the largest value in A1:A20 (same result as MAX(A1:A20)), LARGE(A1:A20;2) returns the second largest value, LARGE (A1:A20;3) returns the third largest value, and so on. When used against a column of running serial numbers 1, 2, 3, …, it can be used to sort a set of numbers in descending order.
- LARGE(*valueRange*; RANDBETWEEN(1; *n*)) also resamples with equal probability the n sample data values stored in *valueRange*.

PERCENTILE(*valueRange*; *k*)

- Returns the *k*th fractile, with *k* being a real number between 0 and 1, from among the data in *valueRange*.
- For example, PERCENTILE(A1:A5;0.5) with (91, 33, 52, 45, 67) in cell range A1:A5 returns the 50th percentile value of 52. PERCENTILE(*valueRange*; RAND()) works like SMALL(*valueRange*; RANDBETWEEN(1; *n*)), except now PERCENTILE also interpolates when the fractile required does not coincide with one of the given sample data points and it returns a continuous value.
- By using PERCENTILE to resample data points, you may end up having a resulting data point which is not one of the given sample data points, due to interpolation. When used with the **INT** function, it is even applicable for resampling from discrete data.

Note 7: AUTOMATIC TABULATION

Multiple Operations is a Calc feature that automatically generates a table of results of a spreadsheet model by replacing up to two of its variables with many given sets of input values for them. This can be done for a single input variable (in 1-dimensional Multiple Operations) with one or more output variables, or for two input variables (in 2-dimensional Multiple Operations) with only one output variable. This operation is particularly useful when the model comprises more than just a formula in a single cell. Its equivalent in Excel is **DataTable**.

We will illustrate, as shown in *Figure A-8*, the construction of a 1-dimensional Multiple Operations table to compute the values of two output variables Y and Z using different values of input variable X. We arbitrarily define Y = 2X and Z = 3X + 2.

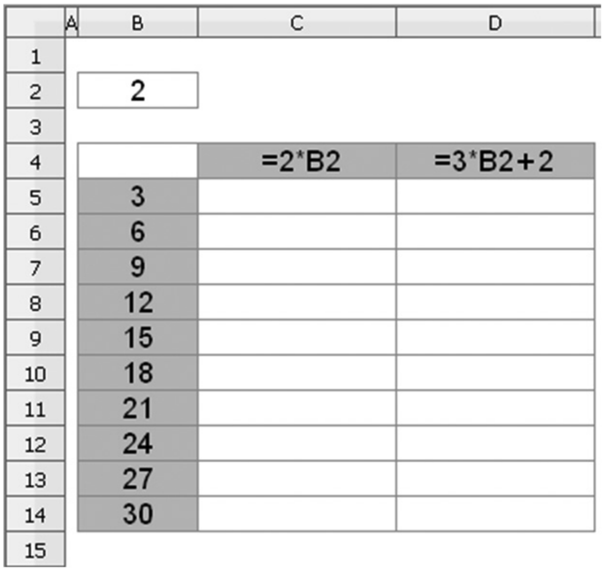
1. Prepare a column containing the candidate values 3, 6, …, 30 for input variable X in cells B5:B14.
2. Enter output formula Y = 2X into the cell C4 as =2*B2. Note that the formula for output variables in Calc can reside anywhere, whereas in Excel they must be in the row just above the first candidate input value for X, with the first output variable taking the first column to the right of the input value column, the second output variable in the second column, and so on. For compatibility reasons, we have followed

the Excel convention. The output formulas must of course refer as input variable X a cell outside of the table. In this example, this cell is B2.

3. Enter the second formula Z = 3X + 2 into the cell D4 as =3*B2+2. This second formula in Calc must reside adjacent to the first formula. Repeat the step for additional formulas (if any) to form an array of formulas.
4. Select the whole table in B5:D14 (inclusive of the column input values, but not the headers or the output formulas) and then activate the computation by selecting **Data/ Multiple Operations** from the main menu in Calc.
5. A dialog will prompt you to enter the **Formulas, Row Input Cell,** and **Column Input Cell**. Since this is a 1-dimensional Multiple Operations table constructed with input X values in a column, you can ignore the Row Input Cell and enter B2 as the Column Input Cell. You can enter C4:D4 as the input for **Formulas**.
6. Results for the output variables will be computed automatically and placed into the table.

Figure A-8

One-dimensional Multiple Operations Table

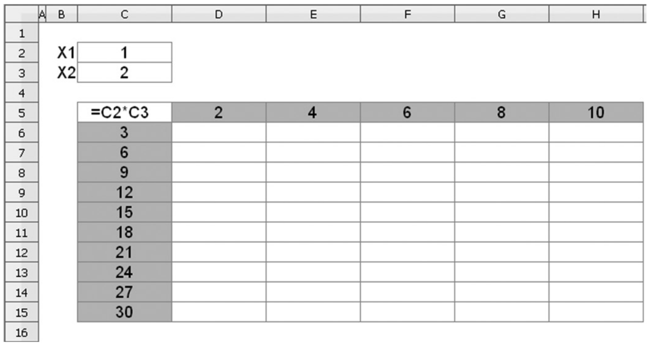


	A	B	C	D
1				
2		2		
3				
4			=2*B2	=3*B2+2
5		3		
6		6		
7		9		
8		12		
9		15		
10		18		
11		21		
12		24		
13		27		
14		30		
15				

The set-up of a 2-dimensional Multiple Operations table that computes the values of output variable W = X1 * X2 for given values of two input variables X1 and X2, as shown in *Figure A-9*, is illustrated as follows:

1. Prepare a column containing the candidate values 3, 6, …, 30 for the first input variable X1 in cells C6:C15.
2. Prepare a row containing the candidate values 2, 4, …, 10 for the second variable X2 in cells D5:H5.

3. Enter output formula W = X1 * X2 into cell C5 as =C2*C3. Note that for a 2-dimensional Multiple Operations, only one output variable can be computed. Its formula in Excel's DataTable must reside at the upper left-hand corner, where the candidate values of X1 and X2 intersect. In Calc, the formula can be elsewhere, but for compatibility we have positioned it there as well. As in the 1-dimensional Multiple Operations, this formula must refer to two cells outside of the table as X1 and X2. In this example, the two cells for X1 and X2 are C2 and C3, respectively.
4. Select the whole table in C5:H15 (inclusive of the row and column input values) and activate the computation by selecting **Data/Multiple Operations** from the main menu.
5. In the dialog, you will be prompted to enter the **Formulas, Row Input Cell**, and **Column Input Cell**. Enter C2 as the Column Input Cell and C3 as the Row Input Cell; both refer to variables X1 and X2, respectively, in the formula. Enter C5 as the input in **Formulas**.
6. Results for output variable W will be computed automatically and placed in the corresponding positions in the table.



	A	B	C	D	E	F	G	H
1								
2		X1	1					
3		X2	2					
4								
5			=C2*C3	2	4	6	8	10
6			3					
7			6					
8			9					
9			12					
10			15					
11			18					
12			21					
13			24					
14			27					
15			30					
16								

Figure A-9

Two-dimensional Multiple Operations Table

Quick Tip

Multiple Operations, unlike Excel's DataTable, cannot be used to collect replication results in *Monte-Carlo* simulations. This is because Calc does not seem to refresh the random values and therefore gives the same result for every replication. You have to write a macro to emulate Multiple Operations but remember to refresh the random values at each replication.

Tool 30 (TableSolve) in Chapter 9 provides macros that emulate Multiple Operations operations and extend them. They allow you to do multiple operation computations that concurrently involve more than two input variables and more than one output variable. Also, the *TableSolve* subroutines can be set to call any subroutine of your choice with every change in data values in its operations. This is particularly useful when the computation involves any form of search or optimization.

It is interesting to note that the Multiple Operations operation is depicted by Calc as a function. If you put your cursor in cell D6 of the Multiple Operations table in *Figure A-9*, you will find the formula =MULTIPLE.OPERATIONS($C$5;$C$3;$C6;$C$2;D$5). The other cells in the table are similarly constructed with cells correctly referenced. You can also construct the table by keying the formula for cell D6, and **Copy** and **Paste** to the other cells. The **MULTIPLE.OPERATIONS** function is however not found in Calc's list of functions. Multiple Operations may not work with all Calc functions. In particular, it does not work well with the **SUMPRODUCT** function.

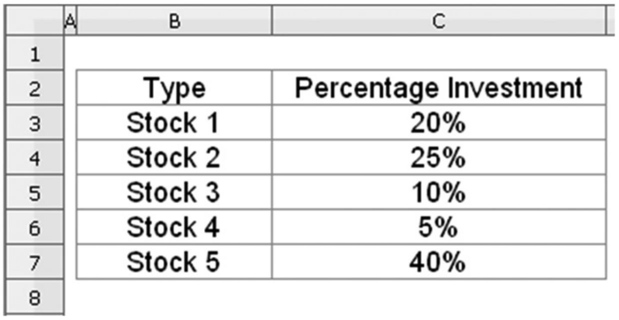
Note 8: CHART Charts are visual representations of data points that can in most cases display results more effectively than tables. Some of the commonly used charts in spreadsheet include bar chart, pie chart, and XY scatter plot. Calc provides a **Chart Wizard** which will guide you in a 4-step process to create the chart of your choice.

Let us construct a column chart for a stock investment portfolio given in *Figure A-10*.

1. Put your cursor anywhere in the table with the data and select **Insert/Chart** to activate Chart Wizard. The whole data set will be automatically selected.
2. Select the **Column** chart type in step 1 as shown in *Figure A-11a*. Once selected, Chart Wizard will provide a visual representation of the chart.
3. Click **Next** to proceed to step 2. In step 2, modify if needed the input in **Data Range**, as shown in *Figure A-11b*.
4. Click **Next** to proceed to step 3. In step 3, enter the data series name and cell ranges, as shown in *Figure A-11c*.
5. Click **Next** to proceed to step 4. In step 4, enter the **Chart Title**, **X-axis** and **Y-axis** labels as shown in *Figure A-11d*.
6. Click **Finish** and the **Column** chart is completed.

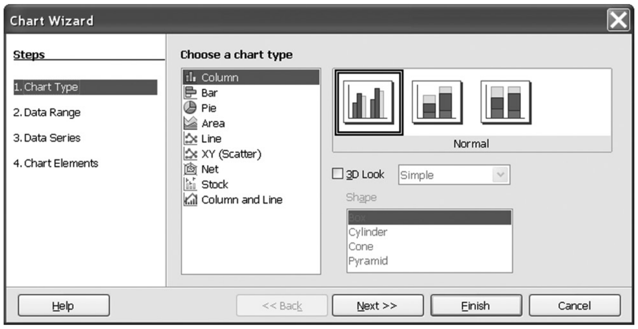
Figure A-10

Stock Investment Portfolio Data



	A	B	C
1			
2		Type	Percentage Investment
3		Stock 1	20%
4		Stock 2	25%
5		Stock 3	10%
6		Stock 4	5%
7		Stock 5	40%
8			

Unlike Excel, there is no selection to put the chart on a separate **Chart Sheet**. To put the chart into another sheet in Calc, you have to right-click on a sheet to use **Insert Sheet** to create a new sheet and then move back to the sheet with the chart, right-click on the chart to select **Cut,** move over to the new sheet and right-click to **Paste** your chart over.



Choose a chart type

Column

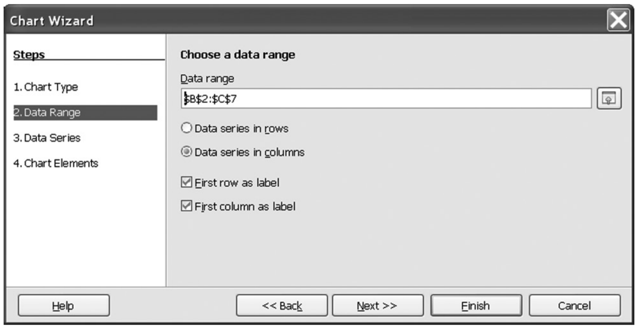
Normal

☐ 3D Look Simple

Shape

Figure A-11a

Step 1 of Chart Wizard



Choose a data range

Data range $B$2:$C$7

○ Data series in rows

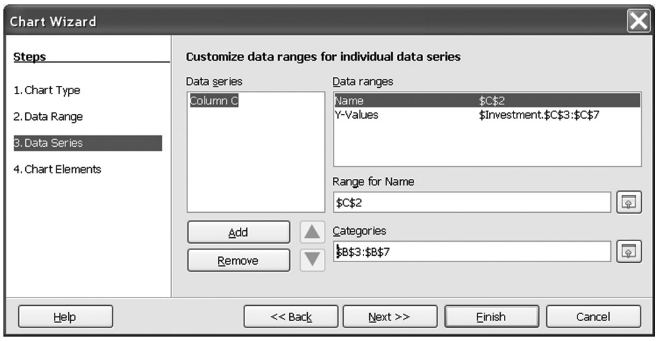
● Data series in columns

☑ First row as label

☑ First column as label

Figure A-11b

Step 2 of Chart Wizard



Customize data ranges for individual data series

Data series: Column C

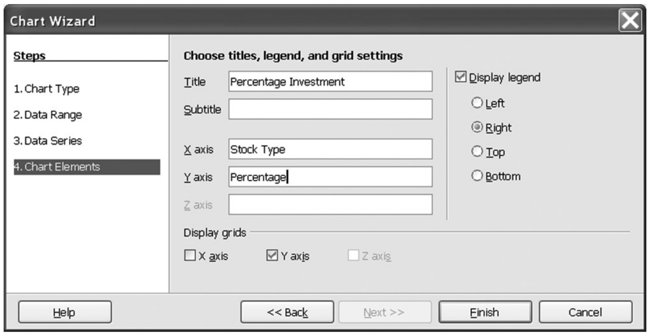
Data ranges	
Name	$C$2
Y-Values	$Investment.$C$3:$C$7

Range for Name $C$2

Categories $B$3:$B$7

Figure A-11c

Step 3 of Chart Wizard



Choose titles, legend, and grid settings

Title Percentage Investment

Subtitle

X axis Stock Type

Y axis Percentage

Z axis

☑ Display legend

○ Left

● Right

○ Top

○ Bottom

Display grids

☐ X axis ☑ Y axis ☐ Z axis

Figure A-11d

Step 4 of Chart Wizard

In **Scatter** charts, you can right-click on the data markers or plotted line to **Insert Trend Line.** In the dialog for this, there are various options available, including setting the trend type, equation, R-square and format.

Note 9: DATE, TIME, AND TEXT

Date and Time

There are three date systems in Calc. In the default **12/30/1899 date system**, 30 December 1899 is day 0, 31 December 1899 is day 1, 1 January 1900 is day 2, and so on. Similarly, in Calc's **1/1/1900 date system**, 1 January 1900 is day 0 and 2 January 1900 is day 1. Thirdly, in Calc's **1/1/1904 date system**, 1 January 1904 is day 0, and 2 January 1904 is day 1. The name of each of Calc's date systems (in the date notation used in the United States) denotes day 0 of that system. The date system for a workbook can be set in **Tools/Options/OOo Calc/Calculate.**

Now, why does Calc have so many date systems? In short, it is to stay compatible with Excel. Excel has two date systems: the default **1900** and the alternative **1904**. In Excel's 1900 date system, the imaginary 0 January 1900 is day 0, 1 January 1900 is day 1, 2 January 1900 is day 2, and so on. This date system erroneously assumes that 1900 is a leap year, but 29 February 1900 does not exist! Therefore, all dates after 29 February 1900 are in fact assigned an integer one larger than they should be. Calculations using dates that straddles 29 February 1900 will be off by a day. Excel's 1904 date system corrects for this mistake by starting instead with 1 January 1904 as day 0.

Calc's 1/1/1900 date system, and its other two date systems as well, correctly recognizes that year 1900 is not a leap year and therefore can be used for all dates from 1 January 1900. But this Calc date system is totally incompatible with Excel's. You can choose to use Calc's 1/1/1904 date system, which is the same as Excel's 1904 date system and therefore totally compatible with it, provided you do not use dates before 1 January 1904. It is recommended that you keep Calc and Excel at their default date systems. With this, all dates from 1 March 1900 onwards will have the same underlying numerical value in the two spreadsheet applications and they are therefore fully compatible. But be extra careful when you deal with dates before 1 March 1900 since it would be correct in Calc but not in Excel.

There are several functions in Calc that allow us to manage dates and days:

- **TODAY**() returns the current date.
- **YEAR**(*serialNumber)* returns the year corresponding to the *serialNumber.*
    - Wrong: YEAR(14-Jan-05)
    - OK: YEAR("14-Jan-05")
    - OK: YEAR(B15) where B15 has the value 14-Jan-05
    - OK: YEAR(39014) = 2006
- **MONTH**(*serialNumber*) returns the month and **DAY**(*serialNumber*) returns the day corresponding to the serial number.
    - MONTH(39014) = 10, DAY(39014) = 24 since 39014 is 24 October 2006 in Calc.
- **DATE**(*year*; *month*; *day*) returns the serial number.

QuickTip

When dealing with subtraction of dates, the minus sign is easily mistaken as the dash used in dates. So, please take note of the correct and wrong ways to denote dates subtraction.

- Wrong: 14-Jan-05 – 23-Sep-04
- OK: "14-Jan-05" – "23-Sep-04" = 113
- OK: DATE(2005;1;14) – DATE(2004;9;23) = 113

Time is also managed in a serial number format, except time is stored as the fractional part of the serial number, that is, the digits to the right of the decimal point. A 24-hour day will make up a whole number 1, so part of a day, say 12 hours, will be 0.5. Let us look at some of the functions that can help us manage time.

- **NOW**() returns the current date and time
- For example, if now is 8:00AM 25 December 2005, then the value of NOW() is 38711.3333333 where
  - 38711 is the day, which is 25 December 2005, and
  - .3333333 is the time of the day, which is 1/3 of the day.
- So, 8:00AM 26 December 2005 is 38712.3333333.
- **TIME**(*hour; minute; second*) returns the serial number to the right of the decimal point in the 0.####### format.
- **HOUR**(*serialNumber*), **MINUTE**(*serialNumber*), and **SECOND**(*serialNumber*) return respectively the hour, minute, and second of *serialNumber*.
  - HOUR(39461.847) = 20
  - MINUTE(39461.847) = 19
  - SECOND(39461.847) = 41

Text Management

Other than values and formulas, Calc can also manage texts (or commonly known as strings, denoting in short a string of characters). Important text functions include **CONCATENATE, LEN, FIND, LEFT,** and **RIGHT**.

CONCATENATE(*string1*; *string2*; …)

- Combines end-to-end text strings *string1, string2,* and others. The alternative approach is to use the concatenation operator **&**.
- Therefore, "ABC" & "DEF" will yield "ABCDEF". This can be done to text or cells with values. For example, ">" & E20 yields ">5" when cell E20 contains the value 5. In this context, Calc implicitly converts number into text before completing the concatenation.

- This operation is helpful for specifying the criteria in functions such as **COUNTIF, COUNTIFS, SUMIF,** and **SUMIFS**. These functions are discussed in the last section of *Note 10.*

LEN(*string*)

- LEN returns the number of characters in text *string*.

FIND(*findString*; *string*; [*startNumChar*])

- **FIND** searches for text *findString* in text *string*, starting from the *startNumChar* position in *string*. Integer *startNumChar* is assumed to be 1 if it is not specified. If successful, it returns the position of the first character of *findString* in text *string*, counting from the left. If there are multiple occurrences of *findString* in *string*, then the first occurrence will be the one found.
- For example, FIND("o"; "Flower shop") returns 3 and FIND("er"; "Flower shop") returns 5.

LEFT(*string*; [*numChar*])

- LEFT returns the left-most *numChar* characters in text *string*. Integer *numChar* is assumed to be 1 if it is not specified.

RIGHT(*string*; [*numChar*])

- RIGHT returns the right-most numChar characters in text string. Integer numChar is assumed to be 1 if it is not specified.

All these text functions can be creatively applied to do many interesting things in your spreadsheets. It can, for example, split a full name into first name and last name, or extract postal codes from full addresses.

Note 10: LOGICAL FORMULA AND FUNCTION

A logical formula is one that returns a TRUE or FALSE value. For example, a cell with formula =(B4=C5) will return a TRUE when the result contained in cell B4 is the same as that of C5 and a FALSE otherwise. In logical formulas, several operators other than the equal operator (=) can be used. These include:

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
< >	not equal to

There are also logical functions such as **AND, OR, NOT**, and **IF.** AND and OR can combine the results of more than one logical test to depict more complex evaluations. AND returns a TRUE if and only if all logical tests within it are TRUE; OR returns a

TRUE when at least one logical test it evaluates is TRUE; NOT transforms a FALSE to a TRUE, and a TRUE to a FALSE. The results of evaluation using AND, OR, and NOT with two arguments can be summarized as follows:

Function	Result
AND(TRUE, TRUE)	TRUE
AND(TRUE, FALSE)	FALSE
AND(FALSE, TRUE)	FALSE
AND(FALSE, FALSE)	FALSE

Function	Result
OR(TRUE, TRUE)	TRUE
OR(TRUE, FALSE)	TRUE
OR(FALSE, TRUE)	TRUE
OR(FALSE, FALSE)	FALSE

Function	Result
NOT(TRUE)	FALSE
NOT(FALSE)	TRUE

Logical formulas can be used in conjunction with arithmetic computations. In such cases, TRUE will be evaluated as a 1 and FALSE as a 0. Therefore, formula =(B4=C5)*1 will return a 1 when the result contained in cell B4 is the same as that of C5 and a 0 otherwise. Also, the multiplication of two logical tests would be equivalent to an AND operation. That is, =(B4=C5)*(D5>F2) would give the same conclusion as AND(B4=C5, D5>F2), except that the former will return a 0,1 result and the latter a TRUE, FALSE result.

IF(*logicalTest*; *resultIfTrue*; *resultIfFalse*)

The IF function is designed to evaluate a *logicalTest* condition, that should return a TRUE or FALSE. If *logicalTest* is TRUE, then *resultIfTrue* will be displayed; if FALSE, then *resultIfFalse* will be displayed. The *logicalTest* condition would be as in a logical formula, except without the = sign prefix. Both *resultIfTrue* and *resultIfFalse* can be values, cell references, or formulas. For example, to test if the age of a person who is over 18 to enter a club is IF(age>18; "Welcome"; "You are underage!").

Quick Tip 1

An IF function with another IF function as its *resultIfTrue* or *resultIfFalse* is known as a nested IF function. Calc permits a maximum of seven levels of nesting.

A more complex example would be a club that requires female members to be over 18 years of age while male members must be over 21. The IF function for this will be IF(OR(AND(age>18;gender="Female");AND(age>21;gender="Male"));"Welcome"; "You are underage!").

Quick Tip 2

MAX(*value*; 0) and **MIN**(*value*; 0) are shorter and more elegant substitutes to IF(*value* > 0; *value*; 0) and IF(*value* < 0; *value*, 0), respectively. So indirectly, MAX and MIN, though classified by Calc as statistical functions, are also logical functions.

Logical Count and Sum

The **COUNT** function counts the number of cells in the specified cell range that contains numerical values, while **COUNTA** counts the number of cells that contains numerical or text values.

COUNTIF(*range*; *criteria*)

- Counts the number of values in the specified cell range that satisfies the criteria.
- The criteria is a simple logical expression expressed (in quotes) as a text. For example, COUNTIF(A1:A20; ">5") counts the number of values that is greater than 5 in cells A1 through A20.
- Now using the **&** concatenation operator and setting the value in cell E20 to 5, COUNTIF(A1:A20; ">"&E20) becomes equivalent to COUNTIF(A1:A20; ">5"). The advantage of the former is that the value in cell E20 can be arbitrarily changed to suit your purpose.

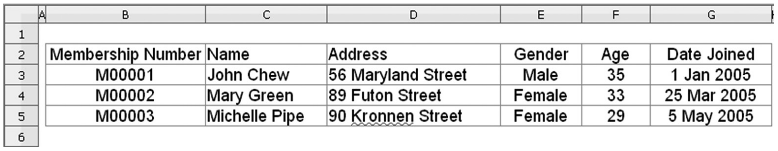
SUMIF(*criteriaRange*; *criteria*; [*sumRange*])

- Sums the number of values in *sumRange* for all corresponding values in *criteriaRange* that satisfy the criteria. *sumRange* may be omitted if it is the same as the *criteriaRange*.
- The criteria is a simple logical expression expressed (in quotes) as a text. For example, SUMIF(A1:A20; ">5") sums the number of values that is greater than 5 in cells A1 through A20.
- Now using the **&** concatenation operator and setting the value in cell E20 to 5, SUMIF(A1:A20; ">"&E20; B1:B20) becomes equivalent to SUMIF(A1:A20; ">5"; B1:B20). The advantage of the former is that the value in cell E20 can be arbitrarily changed to suit your purpose.

Note 11: DATA MANAGEMENT

Data List

A **data list** is a list of records, with each record occupying a row and its attributes distributed over the columns. Each column corresponds to a data field, which is a variable in a record. *Figure A-12* shows an example of a data list of club membership information.

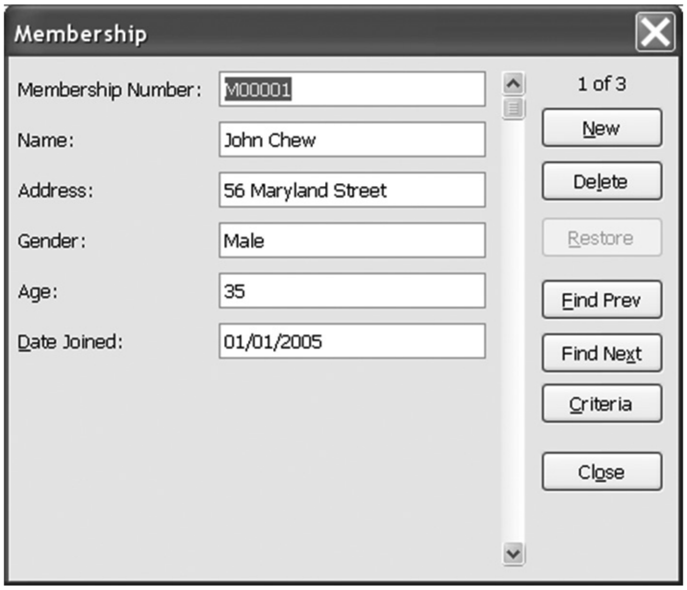


Membership Number	Name	Address	Gender	Age	Date Joined
M00001	John Chew	56 Maryland Street	Male	35	1 Jan 2005
M00002	Mary Green	89 Futon Street	Female	33	25 Mar 2005
M00003	Michelle Pipe	90 Kronnen Street	Female	29	5 May 2005

Figure A-12
Example of a Data List

Unlike Excel, **DataForm** is only available in Calc as an **Add-in**. You can download this Add-in from http://extensions.services.openoffice.org/en/download/3727. After it is downloaded, you need to **Close** Calc and **Open** it again. Using DataForm, users can quickly enter and edit data in a data list. Follow the steps below to create the table in *Figure A-12*.

1. Enter the field labels as table headers into cells B2:G2.
2. Put your cursor anywhere in the table and select **Data/Form** from the menu. An entry form dialog will appear, as shown in *Figure A-13*.
3. To add a new record
   a. Click **New** to clear all current data in the input boxes.
   b. Enter your information into the form.
   c. Click **New** again to append your input to the data list, and continue with more new records, or click **Close** to append the data and end.
4. To edit an existing record
   a. Locate the record using **Find Next** and **Find Previous**.
   b. Edit the information and click **Close** to update.



Membership Number:	M00001
Name:	John Chew
Address:	56 Maryland Street
Gender:	Male
Age:	35
Date Joined:	01/01/2005

Figure A-13
Data Form for Club Membership Information

After experimenting with DataForm, you may notice its several limitations. Here are some we have found:

- No option for adding a drop-down selection list to simplify data entry. This would be particularly useful for fields like *Gender*.

- Does not validate input values. The *Age* field requires an integer input and so a quick check to ensure only valid values within a reasonable range will help ensure data integrity.
- Fixed input box width. A field like *Address* needs a longer input box, whereas *Postal Code* should have a shorter one.

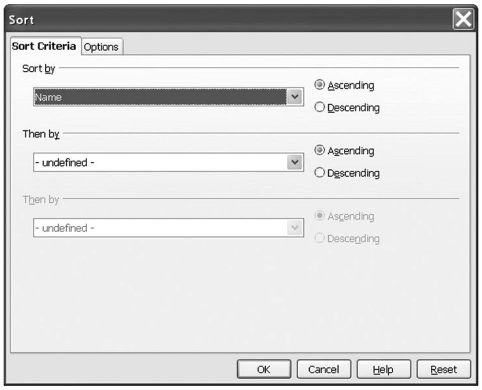
Sort

To sort values in ascending order in a **data list,** put your cursor in a cell in the column of interest inside the table and click the $\substack{A\\Z}\downarrow$ icon. Similarly, you can click on $\substack{Z\\A}\downarrow$ to sort in descending order. A quick way to properly sort a table with data entry that has ties in values in the fields is to move your cursor to the relevant columns to sort and each time click on the sort ascending or sort descending icon. The order in which the columns are sorted is important. You should start with the least significant column, then move progressively to the most significant one.

Alternatively, select all the data you want to sort and select **Data/Sort** from the main menu to call up the **Sort** dialog and work from there. If you have instead put your cursor anywhere in the data list before you do so, the whole cell range that contains the data will be also automatically selected and a dialog will appear, as shown in *Figure A-14*. Calc will automatically detect if the data list has a header row or not. If there are headers, the **Sort by** selections will be made according to the header labels. Otherwise, the sheet column labels will be used instead. Again, you can choose whether to sort each attribute in ascending or descending order.

Figure A-14

Data Sort



Sort by: Name — [✓ Ascending] Descending

Then by: - undefined - — [✓ Ascending] Descending

Then by: - undefined - — [✓ Ascending] Descending

AutoFilter

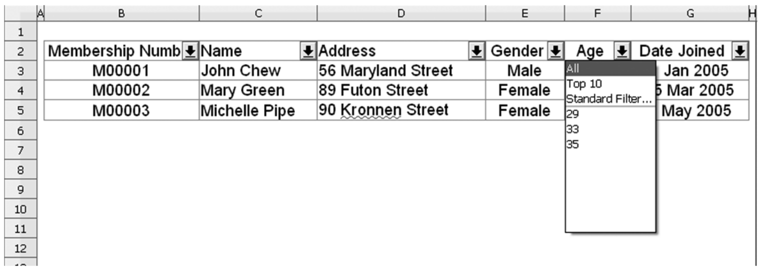
AutoFilter allows us to filter a large number of records in a data list for simple user-selected criteria. This feature is particularly useful when the number of records has grown so large that it is no longer possible to view all records at the same time without excessive scrolling. For example, a club with 2,000 members wants to organize a ladies' event and therefore needs to view the data of female members within an age range to work out a reasonable invitation list.

To set up the AutoFilter for the data in *Figure A-12*, follow the steps below.

- Select any cell within the data list.
- Select **Data/Filter/AutoFilter** to activate it. With this, drop-down tabs will be added to the field labels in the data list header for you to select the various filtering operations for each data field.

It will be a good idea to increase the height of the header row and format the cells there with the text alignment set to top so that the drop-down arrows provided by **AutoFilter** will not hide away the field names in the header. The filtering operations available are:

- **All** – displays all records.
- **Top 10 –** displays for the selected field the top items of the records, as shown in *Figure A-15.*
- **Standard Filter** (See the discussion below.)
- Individual possible entries – show all records of the select entry value in the data field.



Membership Numb	Name	Address	Gender	Age	Date Joined
M00001	John Chew	56 Maryland Street	Male		Jan 2005
M00002	Mary Green	89 Futon Street	Female		5 Mar 2005
M00003	Michelle Pipe	90 Kronnen Street	Female		May 2005

Figure A-15

Top 10 AutoFilter

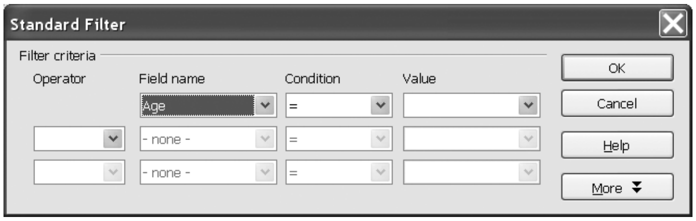
Standard Filter

- Allows you to create a standard filter with up to three criteria, which can be combined by either an **AND** or **OR** logical operator (see *Figure A-16*).
- The criteria include:

=	<=	Largest	Largest %
<	>=	Smallest	Smallest %
>	<>		

- Allows you to use wildcards such as **?** and * to represent any single character or a string of characters, respectively.
- For example, we may want to filter for members whose ages ranged between 30 and 39. You can set the filter for criteria as **"Is greater than or equal to"** 30 **AND "Is less than"** 40.

Figure A-16
Standard Filter



Filter criteria

Operator	Field name	Condition	Value
	Age	=	
	- none -	=	
	- none -	=	

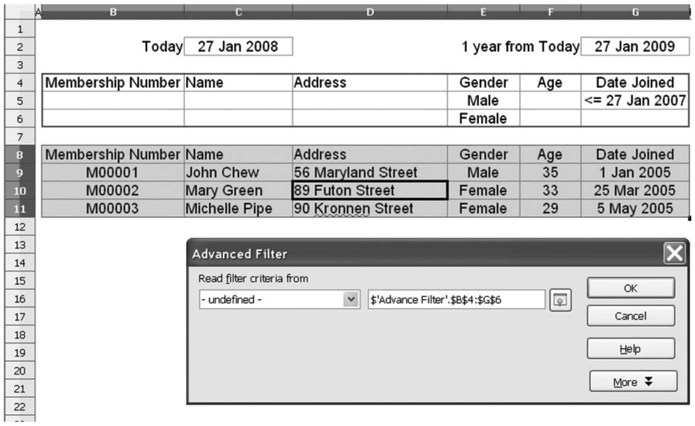
Advanced Filter

Consider now that the club would like to remind its members who have joined for at least one year to collect a special gift. The filtering criteria to set up could be to compare their *Date Joined* with a date that is one year ago from today. If their *Date Joined* is less than or equal to this date (which means it is earlier than or equal to this date), then the member would have joined for at least one year. As this threshold date (one year ago from today) is a value that depends on the date today, you cannot use **AutoFilter** or **Standard Filter**. You will have to use **Advanced Filter**.

In this example as shown in *Figure A-17*, you can set up the spreadsheet model as follows:

1. The criteria range is defined in rows 4 and 5, where row 4 contains the selected field labels as a header and row 5 contains the criteria values for the corresponding fields. The field label to use in the filtering is *Date Joined*, and the criteria value is set up as a formula in cell G5 as ="<=" & TEXT(G2;"d MMM yyyy"), where cell G2 contains the date for one year ago from today.

Figure A-17
Advanced Filter



	A	B	C	D	E	F	G
1							
2		Today	27 Jan 2008		1 year from Today		27 Jan 2009
3							
4		Membership Number	Name	Address	Gender	Age	Date Joined
5					Male		<= 27 Jan 2007
6					Female		
7							
8		Membership Number	Name	Address	Gender	Age	Date Joined
9		M00001	John Chew	56 Maryland Street	Male	35	1 Jan 2005
10		M00002	Mary Green	89 Futon Street	Female	33	25 Mar 2005
11		M00003	Michelle Pipe	90 Kronnen Street	Female	29	5 May 2005

Advanced Filter

Read filter criteria from: - undefined - | $'Advance Filter'.$B$4:$G$6

2. Put your cursor anywhere in the data list and then activate the advanced filter by selecting **Data/Filter/Advanced Filter**. The dialog prompts for the criteria range. Fill in cell range B4:G5 in the second space available and click **OK**.

   Here are some rules to follow when using Advanced Filter:

1. There must be at least one empty row separating the criteria range and data list range.
2. There must be at least one field header in the criteria range that matches a field header in the data list range.
3. Criteria field labels can be arranged in any order in their header and they need not be directly above the corresponding field label in the data list header.
4. An empty criteria field value implies no match restriction for the field.
5. All field entries in the criteria range are combined within a row by the **AND** operator, and criteria of multiple rows are combined across rows by the **OR** operator. For example, in the criteria table in cells B4:G6 in *Figure A-17*, we have two filtering criteria rows, where the first criterion row of male *Gender* member AND *Date Joined* earlier than or at 27 January 2007 is OR to the second criterion of any member who is of the female *Gender*. All records that match the second set of criteria will be appended to records that match the first set of criteria.
6. Do not include an empty criteria row unless your intent is for Advanced Filter to show all records in the data list.

Database Functions

Filtering data in a **data list** hides some of the data. When you use functions such as **COUNT, SUM, AVERAGE,** and **STDEV**, the statistics they provide include all the values, including those that have been filtered out.

Like **COUNTIF** and **SUMIF** types of functions, there are also **DAVERAGE, DMAX,** and other database functions that allow you to compute the results of the data under the specified field (i.e., the header label or equivalent, the column position in the table) that complies with the criteria you give, as is done in Advanced Filter. In all these functions, the data in the table are all visible to the user. To filter and then compute the results for the filtered data, you have to use the SUBTOTAL function.

SUBTOTAL(*functionNum*; *range*)

Depending on the value you specify for *functionNum*, **SUBTOTAL** returns a statistic of the values in cells *range*. If the cells are in a data list where the data may be autofiltered, the statistic you want compute will ignore the hidden values as given by *functionNum*. *functionNum* constants ranging from number 1 to 11, are as follows:

Function Number	Function
1	AVERAGE
2	COUNT
3	COUNTA
4	MAX
5	MIN
6	PRODUCT

Function Number	Function
7	STDEV
8	STDEVP
9	SUM
10	VAR
11	VARP

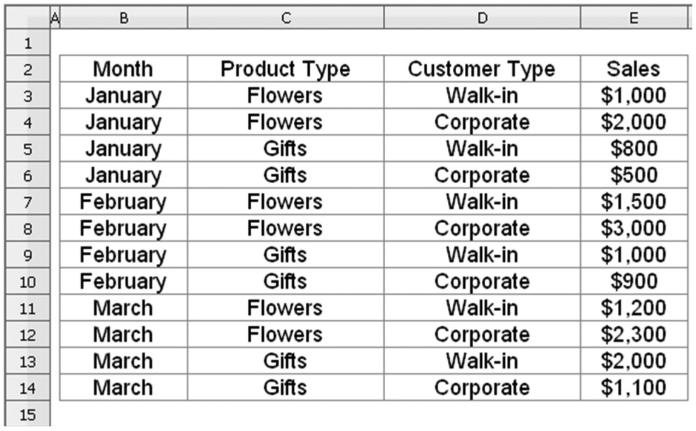
Rows or columns you hide using the **Row Hide** or **Column Hide** format commands are not considered by SUBTOTAL as hidden. This function may be incompatible with Excel's SUBTOTAL function.

DataPilot

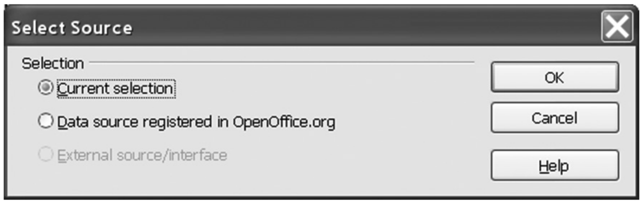
DataPilot table and chart (known as **PivotTable** and **PivotChart** in Excel) help us to create table and chart reports with automatic computation of certain results of interest, such as sum and average, and organize the results according to different attributes. The attributes can be in rows, columns, or pages. Consider the sales performance of a small flower shop for three months as shown in *Figure A-18*. There, sales figure for different product and customer types are given for the months of January to March. This information would be better organized if January, February, and March are themselves headers. This argument extends to using the individual product types as headers and customer types as headers. The only data entries in the example are really the sales figures. This would be a 3-dimensional table. DataPilot can take the given data to automatically create a table such as the one shown in *Figure A-20*.

Figure A-18

Data for DataPilot

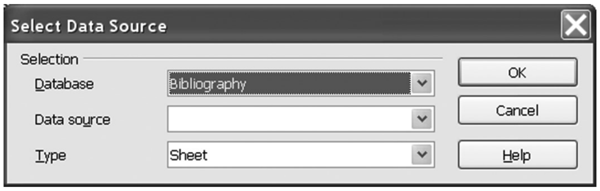


	A	B	C	D	E
1					
2		Month	Product Type	Customer Type	Sales
3		January	Flowers	Walk-in	$1,000
4		January	Flowers	Corporate	$2,000
5		January	Gifts	Walk-in	$800
6		January	Gifts	Corporate	$500
7		February	Flowers	Walk-in	$1,500
8		February	Flowers	Corporate	$3,000
9		February	Gifts	Walk-in	$1,000
10		February	Gifts	Corporate	$900
11		March	Flowers	Walk-in	$1,200
12		March	Flowers	Corporate	$2,300
13		March	Gifts	Walk-in	$2,000
14		March	Gifts	Corporate	$1,100
15					



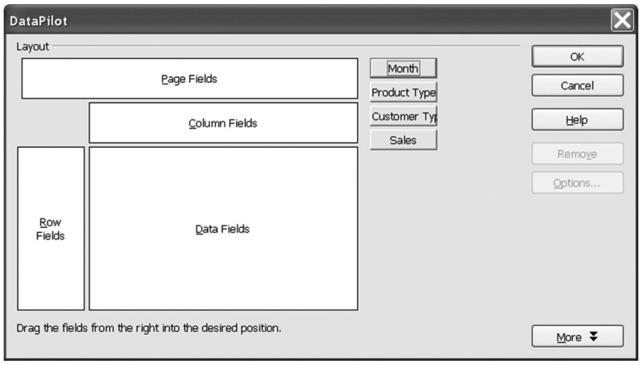
Current selection [✓]
Data source registered in OpenOffice.org
External source/interface

Figure A-19a
DataPilot Source Selection



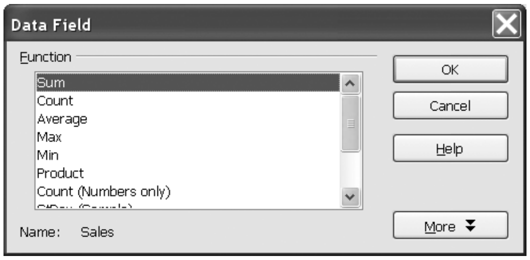
Database: Bibliography
Data source:
Type: Sheet

Figure A-19b
DataPilot Alternative Data Source



Drag the fields from the right into the desired position.

Figure A-19c
DataPilot Table Layout



Name: Sales

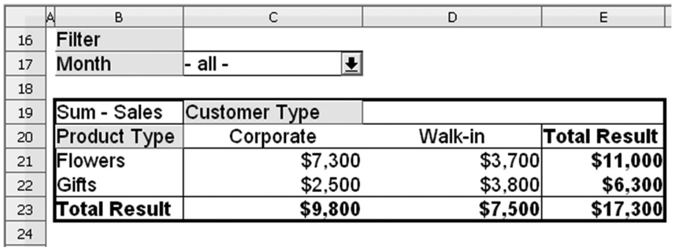
Figure A-19d
DataPilot Table's Data Field Functions

You can do create a DataPilot table and chart as follows:

1. Put your cursor anywhere in the data list, activate **DataPilot** by selecting **Data/ DataPilot** from the main menu and click **Start**. The whole data list will be automatically selected. You can also select the whole data list first, though this is unnecessary, before you activate DataPilot.
2. A dialog will appear as shown in *Figure A-19a*. Since the data source is already selected, you can click **OK**. You can also choose the second option to use an alternative data source. In that case, the dialog will be as shown in *Figure A-19b*.

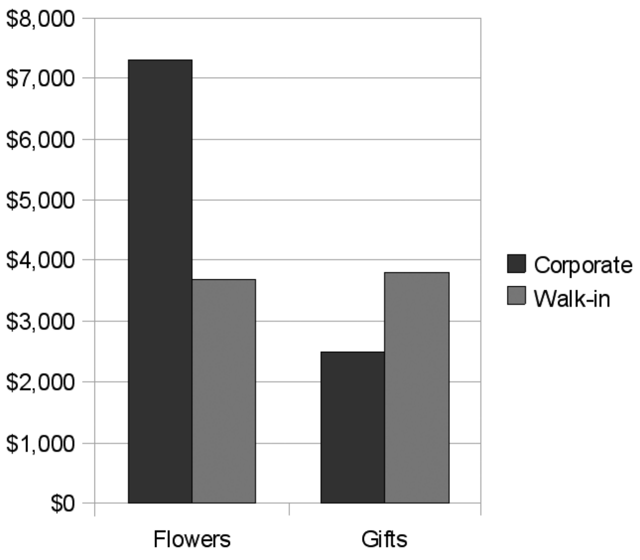
3. After you click **OK**, another dialog appears to let you should define the layout of the DataPilot table as shown in *Figure A-19c*.
   - The field labels from the data source are listed as buttons on the right.
   - Drag these labels, one at a time, into the Page, Row, Column or Data areas according to your preference. In this example, put *Month* in **Page**, *Product* in **Row**, *Customer* in **Column**, and *Sales* in **Data**.
   - Calc by default selects **Sum of Sales** as the method to compute the *Sales* values in the Data area. Double-click on **Sum of Sales** to show another dialog for you to change the computation method. The options available include **Count**, **Average**, **Max**, and **Min**, as shown in *Figure A-19d*. Click **OK** when done.
4. Finally, back at the DataPilot table layout dialog, you can click **OK** to complete the process. The DataPilot table created, placed below your data list, is as shown in *Figure A-20*. Unlike Excel's PivotTable, DataPilot does not have an option for creating the associated chart. You can however apply the steps shown in *Note 8* to produce a chart for the DataPilot table. This is shown in *Figure A-21*.

Figure A-20
DataPilot Table



	A	B	C	D	E
16		Filter			
17		Month	- all -		
18					
19		Sum - Sales	Customer Type		
20		Product Type	Corporate	Walk-in	Total Result
21		Flowers	$7,300	$3,700	$11,000
22		Gifts	$2,500	$3,800	$6,300
23		Total Result	$9,800	$7,500	$17,300
24					

Figure A-21
DataPilot Chart



Put your cursor anywhere in your DataPilot table and right-click your mouse. The pop-up side menu gives you the options to refresh the data, change refresh intervals, set up column and row totals, and others. You can try working with DataPilot in the *Grand Grocery* exercise in Chapter 8.

Note 12: TARGET AND OPTIMIZATION

Goal Seek

We are all too familiar with computing outputs for given inputs. **Goal Seek** is a spreadsheet operation that allows us to work backwards to find the value of a single input variable that gives the desired value for an output variable. The input variable cell contains only a value, while the output variable cell contains a formula. This inverse operation is done in the background by Calc using iterative calculations, much like what one would do by trial and error but smarter. You therefore do not have to change the formulas in the cells to achieve this. You can access the operation by selecting from the main menu **Tools/Goal Seek** in Calc.

Consider the same flower shop incurring a fixed cost of $100 and a variable cost of $5 for each flower bouquet sold. Selling price for each bouquet is $8. How many bouquets must this shop sell to break even? To answer this question, let us first list the key relationships among the variables:

- Profit = Total Revenue – Total Cost
- Total Revenue = Unit Price * Number of Units Sold
- Total Cost = Fixed Cost + (Unit Cost * Number of Units Sold)
- To break even, Total Revenue = Total Cost or Profit = 0

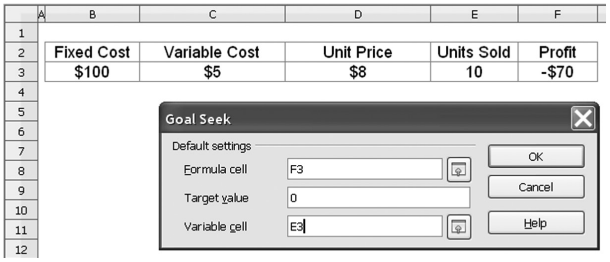
As shown in *Figure A-22*, to find the breakeven price, you can use Goal Seek to set the profit in cell F3 to a target value of 0, by changing the number of bouquets sold in cell E3. Goal Seek will compute and leave the breakeven price in cell E3. We could have in this case easily manipulated the equations to determine the solution directly. However, there will be many other cases where this is not so easy. Even when it is, maintaining two sets of formulas, one in the forward direction and another in reverse, is troublesome.

It is important to note that the underlying Goal Seek algorithm may not converge in its search. Another complication is that there may be more than one solution for the problem posed. The input variable initial value therefore needs to be set appropriately so that the search may end within a reasonable time and find the desired result. To set limits on the search in **Iterative Calculations** in Calc, you can change the maximum iterations and maximum change parameter values in **Tools/Options/OOo Calc/Calculate**. Iterative calculation is the topic explored in the *Charity Donation* exercise in Chapter 4.

The algorithm ends successfully when the difference between the results of two consecutive iterations is less than the specified maximum change. The algorithm will

Figure A-22

Goal Seek



	A	B	C	D	E	F
1						
2		Fixed Cost	Variable Cost	Unit Price	Units Sold	Profit
3		$100	$5	$8	10	-$70

Goal Seek

Default settings

Formula cell: F3

Target value: 0

Variable cell: E3

converge faster when a larger maximum change parameter value is used, at the expense of lower precision in the solution value. You do not normally need to change the default values. In Excel, these parameters would also apply to their Goal Seek computations. It is unclear if this is also true for Calc.

Solver

Solver is available as a standard **Add-in** in Excel, but in Calc, it is directly available. It can do the operation of **Goal Seek** and more. First, you can use it to search for the values of many input variables that give the desired value for an output variable. This is an improvement over Goal Seek that permits only one input variable. Next, Solver allows you to find the maximum or minimum value of an output variable (commonly referred to as the *objective function*) subject to the constraints you specify. These constraints can include lower and upper limits on individual input values or their combinations (e.g., budget constraints).

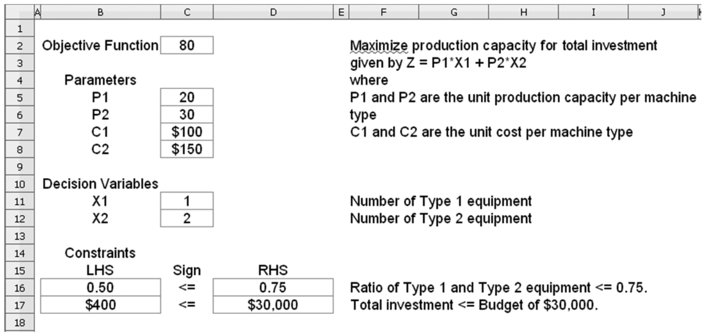
Solver uses mathematical programming methods to determine the solution. These include linear programming, nonlinear programming, integer programming, and combinatorial optimization. In all such optimizations, there are five possible result outcomes:

- The solution found is the unique optimal solution.
- The solution found is one of the many local optimal solutions.
- The solution found is a point of inflection, i.e., neither a minima nor a maxima.
- The minimum (maximum) solution found is $-\infty$ ($+\infty$).
- There are no feasible solutions.

Consider a simple problem faced by a machine shop, as shown in *Figure A-23*. This machine shop is planning to acquire two types of machines. Machine type 1 costs $100 and has a production capacity of 20 units, while machine type 2 costs $150 and has a production capacity of 30 units. If the budget for acquiring these machines is $30,000 and the ratio of machine type 1 to machine type 2 must be less than or equal to 0.75, how many of each machine type should the machine shop purchase in order to maximize its total production capacity?

To solve this problem using Solver, we can set up the Excel model as follows:

1. Objective function
   - o This equation defines the objective of the problem, which is to be minimized or maximized.



	B	C	D	F
1				
2	Objective Function	80		Maximize production capacity for total investment
3				given by Z = P1*X1 + P2*X2
4	Parameters			where
5	P1	20		P1 and P2 are the unit production capacity per machine
6	P2	30		type
7	C1	$100		C1 and C2 are the unit cost per machine type
8	C2	$150		
9				
10	Decision Variables			
11	X1	1		Number of Type 1 equipment
12	X2	2		Number of Type 2 equipment
13				
14	Constraints			
15	LHS	Sign	RHS	
16	0.50	<=	0.75	Ratio of Type 1 and Type 2 equipment <= 0.75.
17	$400	<=	$30,000	Total investment <= Budget of $30,000.
18				

Figure A-23

Solver

- o In this example, cell C2 with the formula =C5*C11+C6*C12 has the *objective function*, computing the total production capacity for the given number of machines. This cell value is to be maximized.

2. Decision variables
   - o These input variables of the model can be binary, integer or real values.
   - o In this example, they are variables X1 and X2 in cells C11 and C12, for the number of machine type 1 and type 2, respectively.
3. Constraints
   - o These equations specify their left-hand side to be <=, >=, or = to their right-hand side. The right-hand side is usually a resource limit value.
   - o Before you can enter constraints into Solver, you need to set up the left-hand-side and right-hand-side cells to store their values or formulas.
   - o In the above example, there are two constraints. The first is the ratio of machine constraint, where the left-hand-side cell B16 has the formula =C11/C12 and the right-hand-side cell D16 has the value 0.75.
   - o The second is the budget constraint, where the left-hand-side cell B17 has the formula =C7*C11+C8*C12 and the right-hand-side cell D17 has the value $30,000.

Once the model is set up, select **Tools/Solver** to activate **the Solver** dialog as shown in *Figure A-24* and make the following selections:

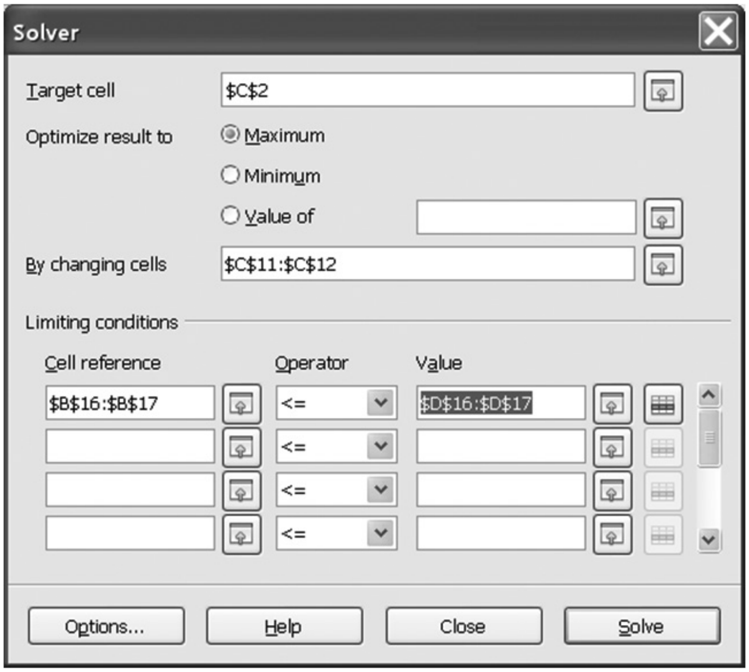
1. Set **Target Cell** to C2.
2. Select **Maximum** as the objective criterion.
3. Enter **By Changing Cells** as C11:C12.
4. Add constraint B16:B17 <= D16:D17.
5. Click **Solve**.

If the problem is well defined, the initial values of the decision variables in **By Changing Cells** are reasonable and the number of decision variables and constraints are not exceedingly large, an optimal solution will soon be found.

The termination parameters for the number of iterations and maximum change can be found by clicking **Options** in the Solver dialog. You do not normally need to change their default values.

Figure A-24

Solver Dialog


Solver

Target cell: $C$2

Optimize result to: [✓ Maximum] Minimum Value of

By changing cells: $C$11:$C$12

Limiting conditions

Cell reference	Operator	Value
$B$16:$B$17	<=	$D$16:$D$17
	<=	
	<=	
	<=	

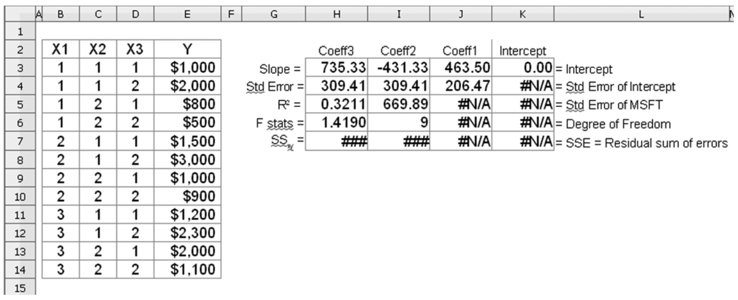
Note 13: DATA ANALYSIS

Statistical Functions

The functions in Calc that currently support statistical analysis include **AVERAGE, STDEV, MIN, MAX, PERCENTILE, RANK, INTERCEPT, SLOPE, TREND, GROWTH, RSQ, NORMDIST, EXPONENTIAL, FDIST, CORREL, CHITEST, CONFIDENCE, ZTEST, TTEST, FTEST,** and **LINEST**. You can call up these functions by clicking the *fx* button in the **Formula** bar and select the **Statistical** category. You can refer to **Help** in Calc to examine in detail how they work.

The most complex function in this list is probably **LINEST**. Array function LINEST(*knownY's; knownX's*; [*const*]; [*stats*]) returns the statistics that describe the *least squares method* for fitting a straight line to the given data set comprising *knownY's* and *knownX's*. It is called an array function because it returns results, not in a single cell but, in a range of cells. If the optional variable *const* is TRUE or omitted, the intercept value is computed. Otherwise, the intercept value is assumed to be 0. If the optional *stats* is TRUE or 1, four additional rows of statistics are computed. If FALSE or omitted, then only the coefficients and the intercept of the fitted straight line are computed.

The example in *Figure A-25* illustrates the use of LINEST. The formula for range H3:K7 is {=LINEST(E3:E14;B3:D14;;1)}. To key in the formula, first select all the cells that are to display the results. If *stats* is FALSE or omitted, this means one row and number of X's plus 1 (for the Y) columns are to be selected. If *stats* is TRUE, this means 5 rows and number of X's plus 1 (for the Y) columns are to be selected. Key the formula =LINEST(E3:E14;B3:D14;;1), without the { } brackets and then key **Shift + Ctrl + Enter**. Brackets { } are provided automatically by Calc to indicate that it is an array function (that is, with **Shift + Ctrl + Enter** applied).



	A	B	C	D	E	F	G	H	I	J	K	L
1												
2		X1	X2	X3	Y			Coeff3	Coeff2	Coeff1	Intercept	
3		1	1	1	$1,000		Slope =	735.33	-431.33	463.50	0.00	= Intercept
4		1	1	2	$2,000		Std Error =	309.41	309.41	206.47	#N/A	= Std Error of Intercept
5		1	2	1	$800		R^2 =	0.3211	669.89	#N/A	#N/A	= Std Error of MSFT
6		1	2	2	$500		F stats =	1.4190	9	#N/A	#N/A	= Degree of Freedom
7		2	1	1	$1,500		SS_{xy} =	###	###	#N/A	#N/A	= SSE = Residual sum of errors
8		2	1	2	$3,000							
9		2	2	1	$1,000							
10		2	2	2	$900							
11		3	1	1	$1,200							
12		3	1	2	$2,300							
13		3	2	1	$2,000							
14		3	2	2	$1,100							
15												

Figure A-25

LINEST Statistics

Array formulas

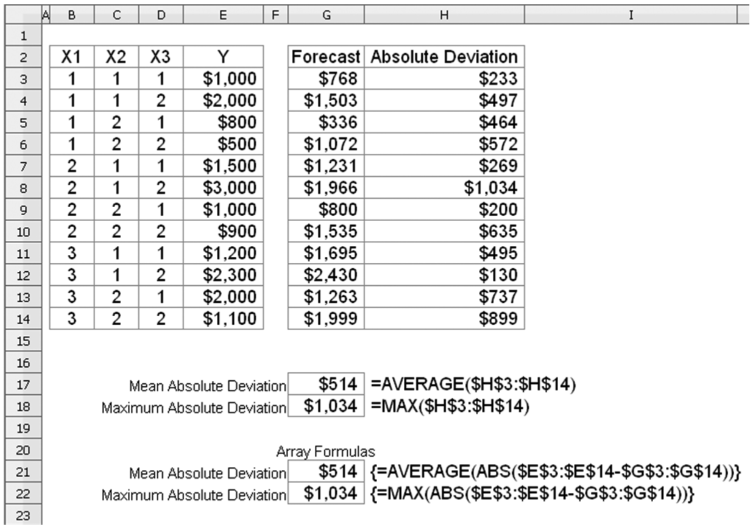
Array formulas do not have to involve array functions. They all have the distinctive { } brackets, which arise after you type the formula with **Shift + Ctrl + Enter**, instead of the usual **Enter** key. Many interesting computations can be done as array formulas and they usually give spreadsheet models that are more compact. If you understand array formulas, such models may in fact be easier to comprehend. *Figure A-26* shows two ways for computing the errors of forecasting variable Y using a linearly fitted equation given by the coefficients and intercept values from the table in *Figure A-25*.

Cell G17 gives the average absolute deviations, comparing each value in cells E3:E14 with the corresponding forecast in G3:G14. Similarly, G18 gives the maximum absolute deviation. The same results can be found in cells G21 and G22. The formula for cell G21 is {=AVERAGE(ABS($E$3:$E$14–$G$3:$G$14))} and for cell G22 is {=MAX(ABS($E$3:$E$14–$G$3:$G$14))}. These formulas do not reference the values in cells H3:H14 and therefore do not need them to be computed if these array formulas are used instead of those in cells G17 and G18.

Even though both G21 and G22 are single cells, array formulas have to be used because of the calculations involved. Let us examine these formulas in detail. The first computation step in these formulas are $E$3:$E$14–$G$3:$G$14, which means $E$3–$G$3, $E$4–$G$4, …, and $E$14–$G$14. These 12 values form an array, on which we next apply the **ABS** function to find absolute differences. The results are collectively still an array. The functions **AVERAGE** or **MAX** are then applied on this array. In general, array formulas involve array subcomputation results or array functions and can, but do not necessarily need to, have array results.

Figure A-26

Array Formulas



X1	X2	X3	Y	Forecast	Absolute Deviation
1	1	1	$1,000	$768	$233
1	1	2	$2,000	$1,503	$497
1	2	1	$800	$336	$464
1	2	2	$500	$1,072	$572
2	1	1	$1,500	$1,231	$269
2	1	2	$3,000	$1,966	$1,034
2	2	1	$1,000	$800	$200
2	2	2	$900	$1,535	$635
3	1	1	$1,200	$1,695	$495
3	1	2	$2,300	$2,430	$130
3	2	1	$2,000	$1,263	$737
3	2	2	$1,100	$1,999	$899

Mean Absolute Deviation $514 =AVERAGE($H$3:$H$14)

Maximum Absolute Deviation $1,034 =MAX($H$3:$H$14)

Array Formulas

Mean Absolute Deviation $514 {=AVERAGE(ABS($E$3:$E$14-$G$3:$G$14))}

Maximum Absolute Deviation $1,034 {=MAX(ABS($E$3:$E$14-$G$3:$G$14))}

Note 14: EXCEL 2003 VS EXCEL 2007 In case you are still using Excel 2003 or older versions of Excel and considering migrating to Excel 2007 or newer versions, here is a quick summary comparing Excel 2007 to Excel 2003. First, the worksheet in Excel 2007 is now much larger with 1,048,576 rows and 16,384 columns (A to XFD). It can handle more than the 1 GB memory limit in Excel 2003, constrained only by the memory allocated by your computer's operating system.

Other than larger memory, some features are also extended. For example, cells can now have **Conditional Formatting** beyond the previous three conditions. There are many more new and easy-to-use features like **Data Bars, Color Scales**, and **Icon Sets**. There is also now a **Rules Manager** that allows you to manage the conditional formatting of the cells in the worksheet all at once. The IF function can now be nested up to 64 levels, much more than the 7-level limit in Excel 2003 and earlier versions. **COUNTIF,** which allows only one criteria range and criteria pair, has an extended version called **COUNTIFS**. This new function can cater up to 127 criteria ranges and criteria pairs. There are also the **SUMIFS, AVERAGEIF, AVERAGEIFS,** and other new functions.

The main difference between the Excel versions is the menu system, which has been reorganized. The **Office Button** in the extreme top-left corner now provides access to **New, Open, Save, Print,** and other file management tools. Author's name and organization information are entered in **Prepare** under **Properties**. Note that **F1** is still the **Help** key. **Help** can also be accessed by clicking the **?** icon in the top-right corner of the menu. To the right of **Office Button** is the **Quick Access** toolbar, where you can put the icons of the tools you frequently use but are not available in the main menu. Placed there by default are **Save, Undo,** and **Redo**.

Instead of vertical drop-down menus to the main menu items as in Excel 2003, Excel 2007 has made the main menu items into tabs in the **Ribbon**, which organize the tools in groups horizontally. The **Home** and **Page Layout** tabs, the first and third tabs, cover most of the cell and worksheet formatting work. In the bottom-right corners of these groups is a tiny diagonal down-right arrow icon. Clicking this icon will bring up the Excel 2007 **Format Cells** dialog, which looks exactly like the one in Excel 2003.

The second **Insert** tab has the **PivotTable, Pictures, Charts, Hyperlinks,** and **Text** groups of tools, all largely related graphics features. **Formulas** is a whole menu item (tab) by itself, focusing mostly on cell calculation features. Listed here are also the **Define Names, Formula Auditing,** and **Calculation** tools. Data access from external sources and **Data Sort, Filter, Validation,** and **Analysis** tools are in the **Data** tab.

In the **Data Tools** group of the Data tab, there is a **What-If Analysis** button. **Scenario Manager, Goal Seek,** and **Data Table** now reside somewhat obscurely here. **Solver** is now grouped with **Data Analysis Tools** in the **Analysis** group, the last in this tab. This group is only present if it had first been added. To add **Solver**, click the **Office Button,** then **Excel Options** (in the bottom-right corner) and **Add-ins** option group. In the **Manage** box, select **Excel Add-ins** and click **Go**. Check **Solver**. You might as well at the same time check **Analysis Toolpak,** and then click **OK**.

Comments and **Protection** features are now under the **Review** tab. **Custom Views**, worksheet gridlines and header options, window split, freeze and zoom, and access to macros are now in the View tab. The last tab, the **Add-Ins** tab, has property tools. Less regular features are no longer in toolbars and have to be added singly. For example, to use text-to-speech commands, you need to get to **Quick Access Toolbar**. To do this, click the **Customize Quick Access Toolbar** icon, which is just next to **Quick Access Toolbar**. Click **More Commands**. In the **Choose** commands from list, select **All Commands**. Scroll down to find **Speak Cells**, click **Add,** and click **OK**.

Many of the option settings previously found in **Tools/Options** of the main menu in Excel 2003 are now more remotely placed in **Office Button**. The settings there have to be initialized to your preference, at least once after installing Excel. To do this in Excel 2007, click **Office Button**, and then click **Excel Options** in the bottom-right corner of the pop-up panel. In the dialog, click **Popular**. You will want to ensure the **Show Developer tab** option is checked. This makes the macro recording and forms features more accessible. Click **OK**.

Now, review other option groups carefully, one at a time. Here are some more suggestions. In **Formula**, set workbook calculation to **Automatic** and disable **Iterative Calculation**. In **Proofing**, set the language dictionary to your preferred language. In **Save**, set your preferred **Auto Recovery** time intervals and file locations. Next in **Trust**, click **Trust Center Settings** and then **Macro Settings**. Select the second option of disabling all macros with notifications.

Finally, you can open Excel 2003 files (named with extensions **xls, xlt,** and **xla**) in Excel 2007. After working on them, Excel 2007 will by default still save them in their original format. A new workbook started in Excel 2007 will on the other hand by default be saved in the Excel 2007 format (named with extensions **xlsx, xlsm, xltx, xltm,** and **xlam**). It can still be **Save As** into a file of one of the earlier formats. You may encounter some incompatibility difficulties working in Excel 2007 when the workbooks are in Excel 2003 format. A particular one to note is that when you use Excel 2007 features that do not exist in Excel 2003, the changes may not be saved. Therefore, you will soon want to convert your current Excel files to the new Excel 2007 format.

A special point to note in the file format migration is that while all **xls** files can be **Saved As** to become **xlsx** files, those with macros will have their macros stripped away. To retain the codes, you will have to convert workbooks with macros into **xlsm** files, with **m** for macro. Similarly, Excel template files with macros should be saved as **xltm** files. By definition, all Excel **Add-in** files have macros and thus they have to be **xlam** files. This thus explains the larger collection of new extensions in Excel 2007.

Note 15: MIGRATING TO OPENOFFICE CALC

Calc is the spreadsheet in OpenOffice. It has a user interface practically the same as that of Excel 2003.

Here are the few minor differences between Excel and OO Calc 3:

- **Align, Group, Flip,** and other graphics manipulation tools found in Excel 2003's **Drawing Toolbar** are in **the Format** menu.
- **Conditional Formatting** in Calc is similar to Excel 2003's. Calc uses **Formatting Styles** and you need to create the style if you are not using the standard ones.
- Arguments in Excel functions are separated by commas; the ones in Calc are separated by semi-colons.
- **Formula Auditing** is called **Detective** in Calc.
- There is no **Data Form** in Calc. It is available as an **add-in**.
- **PivotTable** in Excel is called **DataPilot** in Calc.
- **Solver** is directly available in Calc, but the tool is possibly not as rich or mature as the **add-in** in Excel.
- The equivalent **macro** language in Calc is OO **Basic** (also referred to as **StarBasic**) which, like **Visual Basic** (VB) used in Excel, is derived from the same underlying **BASIC** programming language. It however uses a different object library to access spreadsheet components. **VBA** codes are ported over automatically when Excel files are opened as OO Basic codes in Calc. All the codes are made into comment statements (with the addition of `REM` to the start of every line), which render the programs useless. You can use the Basic editor to automatically replace "REM" with "". The spreadsheet developers then have to manually revise the codes,

which may not be a trivial task. This is because the linkage between OO Basic and Calc is not as established as that of VB with Excel to give VBA.

- You can use VBA codes in Calc by inserting an additional `Option VBASupport 1` line at the start of each module. However, not all the spreadsheet features can be manipulated in OO Basic yet. The good news is that **Solver** can be called in OO Basic to automate model optimization.
- To give a sense of the differences in the macro codes, here is an example of the same program written in the two languages:

```
Sub VBA_Example()
  Range("B3").Select
  Range("B3") = "Hello"
End Sub

Sub OOBasic_Example()
  oSheet = ThisComponent.CurrentController.ActiveSheet
  oCell = oSheet.getCellRangeByName("B3")
  ThisComponent.CurrentController.select(oCell)
  oCell.String = "Hello"
End Sub
```