SOCIETY OF ACTUARIES

Article from:

Actuary of the Future

November 2014 ï Issue 37

Excel Corner

By Shirley Ai Yan Wu

The "Problem Solving and Excel Best Practice" webcast sponsored by the Actuary of Future Section on July 22 was a great success! Special thanks to our two speakers, Nick Komissarov and Aisling Metcalfe, for sharing their expertise on Excel spreadsheet building and documentation best practice. Inspired by all the great questions and feedback from the audience, we will be starting an "Excel Corner" to share tips and tricks on Excel spreadsheet and macro development relevant to the actuarial career.

To begin, we will recap the best practices for spreadsheet building advised by Nick in the webcast.

- Objective and procedure clearly stated
- Version control and history
- Clearly labeled inputs
- Named ranges
- Inputs and outputs clearly separated
- Clear formatting
- No hardcoded values on the calculation tab
- Output consistency checks.

These are great tips that will be proven useful in any spreadsheet building exercise.

In this first article, I will share some tips on performing **Value Search** using Excel, as feedback from the webcast reflected a great interest in this topic. In particular, I will explain **VLOOKUP/HLOOKUP** vs. **INDEX & MATCH** vs. **OFFSET & MATCH.**

Most actuarial models involve dealing with large data, such as looking up the mortality rates for some specific issue characteristics from a collection of mortality rates. One very common and easy function to use is **VLOOKUP/HLOOKUP**—I still remember getting first exposure to this function when I was a co-op and thought it is a very cool function! The syntax is as follows:

VLOOKUP (lookup_value, table_array, col_index_num, range_lookup)

HLOOKUP(lookup_value, table_array, row_index_num, range_lookup)

VLOOKUP involves two steps:

1) It locates the position (row number) of a key, lookup_value, within the first column of a specified table, the table_array.
2) It returns a data point from the same table that is in the row position returned from 1) but from a user-specified column, col_index_num.

Tip: You can think of it as searching for a key downwards within first column of a table, then moves to the right by some columns specified by the user.

range_lookup is an optional Boolean to specify whether an exact match is needed, and the default value is FALSE, meaning it will look for the next largest value that is less than *lookup_value* if no exact match is found.

Tip: Your data MUST be sorted in ascending order if you use approximate match, because the function stops searching as soon as it finds the next largest value that is less than the key. This also means if you know there will be an exact match in the data, you can enhance the calculation speed with using approximate match on sorted data. If the *range_lookup* is TRUE but an exact match is not found, the function returns an "#N/A" error.

The V in **VLOOKUP** stands for vertical. A similar function, **HLOOKUP** (H for horizontal) is available if the goal is to return a data point that is in the same column of the *lookup_value*, i.e., searching left to right in first row then moves downward by a specified row number.

There are limitations to the **LOOKUP** functions:

1) The column/row reference that you specify is static: If you insert or delete columns/rows that affect the table array input of a **LOOKUP** formula, you MUST go back to change the column/row reference to accommodate for the shifted position. This can cause inefficiency in model building when you have multiple formulas pointing to the same table array.

2) The function only allows for a uni-directional search: **VLOOKUP** always searches in the leftmost column

Shirley Ai Yan Wu, FSA, MAAA, is currently a Candidate for Master of Finance at the Judge Business School of Cambridge University. She can be reached at aywushirley@gmail.com.

CONTINUED ON **PAGE 16**

of a table then moves to the right to find the data point. It doesn't allow you to do a search in the middle of a table array then move left/right. Similarly, for **HLOOKUP** you can only search in first row then move down.

One alternative to the **LOOKUP** functions is the combination of **INDEX** and **MATCH**.

INDEX(array, row_num, column_num)

MATCH(lookup_value, lookup_array, match_type)

INDEX returns a value in a specified row and column of a table array. **MATCH** is similar to step 1 in the **LOOKUP** functions, i.e., finds the position of a lookup_value within a specified lookup_array. The *match_type* is an optional Boolean and is similar to the *range_lookup* indicator in the **LOOKUP** functions.

Tip: if you leave *row_num* as zero in the **INDEX** function, it will return the whole column of the array, and similarly for leaving *column_num* as zero if you want an entire row. To do so, you must select a horizontal (vertical) range for the row (column) output from **INDEX**, enter the formula, then press CONTROL+SHIFT+F9.

When **INDEX** and **MATCH** are used together, it is more powerful and flexible than the **LOOKUP** functions. First of all, it solves the problem of performing only a uni-directional search within a specified table, because now you can feed in different arrays for **INDEX** and **MATCH**. You can use **MATCH** to locate the position within any column of a table then using **INDEX** to find the value from a different column of a table, similarly for search within rows. At this point, you should also recognize that **INDEX** and **MATCH** together can perform both **VLOOKUP** and **HLOOKUP** functions. Additionally, using the **MATCH** function to locate the column and row positions allows for dynamic reference in data searching, instead of knowing the exact row/column to look for. Of course you can set up the **LOOKUP** functions with **MATCH** to achieve similar results. Last but not least, the **INDEX & MATCH** combination is proven to have faster processing speed than the **LOOKUP** functions, especially when you are dealing with huge data. The **LOOKUP** functions have to evaluate an entire large data table, whereas **INDEX & MATCH** functions only need to evaluate what you take as the input.

Another option for value search is to use the **OFFSET** function.

OFFSET(reference, rows, cols, height, width)

The function returns the value that is located in a specified number of rows and columns away from the reference value. Height and width are optional inputs and allow the user to further specify the number of rows and columns to be returned.

OFFSET can also be used with **MATCH** to achieve dynamic referencing. However, be mindful that the **OFFSET function** is a volatile function and slows down spreadsheet recalculations.

Tip: Excel supports the concept of a volatile function, that is, one whose value cannot be assumed to be the same from one moment to the next even if none of its arguments (if it takes any) have changed. Excel re-evaluates cells that contain volatile functions, together with all dependents, every time that it recalculates. For this reason, too much reliance on volatile functions can make recalculation times slow.

Reference: http://msdn.microsoft.com/en-us/library/office/bb687891(v=office.15).aspx

Since **OFFSET** is volatile, I suggest using **INDEX** if you are dealing with a complex Excel model to enhance the processing speed.

Hope you find the tips and explanations useful! For future issues, please feel free to send in requests for any Excel-related questions to *aywushirley@gmail.com.* ☆