ORIE 4820: Spreadsheet-Based Modeling and Data Analysis
Acme Order Analysis Lab Exercise (Part II)
Spring 2013

This is the second part of a two-part exercise on ***fundamental data analysis***. The template file is ***Acme-order-analysis-Part-II.xlsm***. Download a copy of the file from the course Blackboard site and *save it on your computer*.

There are three worksheets in the workbook:

- ***Customer Orders*** contains the same 2012 *CallSign* order transaction dataset that we augmented as part of last week's lab exercise.
- ***Product Summary By Month*** and ***Product Summary By Customer*** are worksheets that we will develop today.

Topics and tools we will cover in Part II:

- Retrieving data using ***lookup and reference*** functions: **vlookup**, **index, match, offset**
- Using ***text concatenation*** to create dynamic headers and titles: **&**, **concatenate**
- Using ***data validation*** to create drop-down lists and to restrict cell values
- Referencing named datasets with text: **indirect**
- Using ***conditional logic*** functions for extraction and summarization: **if**, **sumifs, averageifs**
- Using ***conditional formatting*** to control how cell values are displayed
- Using ***array functions*** to retrieve and summarize ranges of data: **offset**, **frequency**, **if**
- Creating ***dynamic charts and histograms***
- Attaching ***spinners*** to cells to facilitate changing values

If you have problems or questions at any point during the exercise, please raise your hand.

Background:

As we saw last week, PivotTables and PivotCharts are extremely powerful tools for summarizing and examining transactional data in a dynamic manner. However, it is frequently the case that a particular fixed data partition is important for decision making purposes (e.g., by Product and Month), and hence, there is often the need to build separate spreadsheet analysis tools that focus on those fixed structures. Moreover, certain types of data visualization tools (such as frequency histograms) can be tedious to create using PivotTables and PivotCharts.

On the worksheets *Product Summary By Month* and *Product Summary By Customer*, we will build flexible summarization tools for these two specified data partitions as well as dynamic visualization tools to facilitate analysis and decision making.

There are several things to note about the *Product Summary By Month* worksheet:

- This worksheet contains five ***named cell ranges*** (four single cells, and one table). To view all of the defined names in the workbook, along with their associated range addresses and properties, from the ribbon, select Formulas->Defined Names->Name Manager. The fact that the source data columns on the *Customer Orders* worksheet are named will play a key role in enabling us to create flexible analysis tools.

- The named cell Selected_Summary (E3) has been restricted so that the user can populate it with values from a specified drop-down list (and *only* those values). This restriction has been accomplished using ***data validation***. To see how this is done, highlight the cell, and from the ribbon, select Data->Data Tools->Data Validation. On the Settings tab, the Allowed values have been restricted to a "List" and the Source has been populated with a string of text values that are the defined range names of the columns of data on the *Customer Orders* worksheet.

- ***Text concatenation*** has been used in several places (D6, C8, D28, I24) to create dynamic labels or titles. To see this, press Ctrl-` (accent grave) to toggle between a regular view and a formula view of the worksheet. The **&** operator has been used to accomplish this in all cells except for I24. In cell I24, the **concatenate** function has been used.

Section 1: Using INDIRECT and SUMIFS to Dynamically Partition Data

The function **sumifs**(*sum_range, crit_range_1, crit_1, crit_range_2, crit_2, …*):

- Is a conditional logic function that adds together the elements of *sum_range* whose corresponding entries in *crit_range_1*, *crit_range_2*, etc, respectively meet the criteria designated by *crit_1*, *crit_2*, etc.
- Up to 127 *crit_range_#*/*crit_#* pairs can be specified, and each *crit_range* must contain the same number of rows and columns as *sum_range*.
- If the corresponding entries to an element do not meet *all* of the criteria specified, then that element of *sum_range* is excluded from the summation.
- There are corresponding functions **countifs** and **averageifs** that count and average, respectively, the elements in a range that meet specified criteria (instead of summing them).

Given a user-selected dataset name in cell E3, we want the Product_Summary_By_Month_Table to display a partition of that dataset by Product and Month. We will accomplish this using the **sumifs** function with two *crit_ranges*: the named Product and Order_Month datasets.

(1) ***Pre-populate cell D9*** with "=SUMIFS(Order_Quantity,Product,D$8,Order_Month,$C9)" and press Enter. Note that the *sum_range* argument is **hard-coded** to be Order_Quantity, so that at present, the result will be the total Order_Quantity (of Alpha in January). We will come back and fix this shortly so that the other datasets (Sales_Revenue, COGS, Gross_Margin) will also be viewable according to the dataset specified in E3.

(2) Use the PivotTable or the AutoFilter tool on the Customer Orders dataset to verify the conditional sum result for Order_Quantity.

(3) In cell E3, select "Order_Quantity", and in cell F3, type "=SUM(Selected_Summary)" and press Enter. Note that the result in F3 is *not* the sum total of the Order_Quantity column.

This is because the entry in cell E3 is simply *text* (like "Bob"), and the formula in F3 does not recognize this text as a defined name. We can fix this using using the ***indirect*** function.

(4) The **indirect** function in Excel allows you to ***refer to a named cell range indirectly***. For example, go back to cell F3, and enter "=SUM(INDIRECT(Selected_Summary))". Now you will get the sum total of the Order_Quantity column.

(5) Armed with this information, we can ***fix the entry in cell D9 and populate the table***:

   a. In cell D9, replace the hard-coded **Order_Quantity** that is the *sum_range* argument for the **sumifs** function with "INDIRECT(Selected_Summary)" and press Enter.

   b. Copy this formula across and down the table.

   c. In cell E3, select "Sales_Revenue". Use the PivotTable or the AutoFilter tool on the Customer Orders dataset to verify the table results are correct.

(6) Note that the table entries display in a generic number format, regardless of the dataset selected in cell E3. Since all of the datasets of interest besides Order_Quantity are dollar values, it would be nice if we could ***automatically change the display format of the cells in the table*** whenever anything other than Order_Quantity is selected. We can accomplish this using ***conditional formatting***:

   a. Highlight the cells in the table, including the Totals.

   b. From the ribbon, select Home->Styles->Conditional Formatting.

   c. Select New Rule, and in the upper Rule Type box, highlight "Use a formula to determine which cells to format".

   d. In the "Format values where this formula is true" area, enter the formula "=($E$3<>"Order_Quantity")".

   e. Click Format, and on the Number tab, select "Currency".

   f. Click OK twice. Verify the formatting rule by selecting different values in cell E3.

   g. Select cell D9 and press Ctrl-C to copy.

   h. Using the Ctrl-key, highlight cells K24 and E30:E42.

   i. From the ribbon, select Home->Clipboard->Paste dropdown.

   j. Under Other Paste Options, select "Formatting", and click OK. This will apply the conditional formatting rules to the selected cells.

Section 2: Using INDEX, MATCH, and OFFSET to Retrieve Data

Now that the Product_Summary_By_Month_Table is populated and properly formatted, in the bottom half of the worksheet we want to give the user an area to focus on a *particular product*.

Specifically, for any Selected_Summary (cell E3) and any Selected_Product (cell E24), we want to display the corresponding data from the table in cells E30:E41, along with a column chart showing the time series. Also, for any selected Product_Start_Month and Product_End_Month selected in cells H24:H25, we want to display for the user (in cell K24) the *partial sum of the summary values* for the product across the selected range of months.

(1) First, let's ***use data validation to create dropdown boxes*** in cells E24 and H24:H25 so that the user can select, respectively, any of the product names and months in the table:

   a. Click on cell E24, and from the ribbon select Data->Data Tools->Data Validation. (Note that this cell has already been named Selected_Product.)

   b. On the Settings tab, select "List" in the Allow dropdown.

   c. In the Source box, specify the range D8:M8 (the product name headers in the Product_Summary_By_Month_Table) and click "OK".

   d. Use data validation on cells H24:H25 so that the list of months may be selected.

(2) Next, let's ***make sure that the user is flagged*** if the selected Product_Start_Month is *after* the Product_End_Month. We will do this two ways:

   a. Use ***conditional formatting*** on cells H24:H25 so that these cells display with *red fill and bold italicized font* if the selected Product_Start_Month is after the Product_End_Month. This can be accomplished by populating the "Format values where this formula is true" area with the following formula, which uses the **match** function to compare the selected month indices:

=MATCH(Product_Start_Month,$C$9:$C$20,0)>MATCH(Product_End_Month,$C$9:$C$20,0)

   b. Populate cell H23 with an **if** statement to display an appropriate error message if the selected Product_Start_Month is after the Product_End_Month.

The table in cells D28:E42 is intended to be the dynamic data source for a column chart (which needs a fixed cell range as a source). We have already seen one way of populating cells E30:E41 with the corresponding values from the Product_Summary_By_Month_Table using the **vlookup** function in conjunction with the **match** function. An alternate way uses the **index** function in conjunction with the **match** function.

The function **index**(*matrix*, *i*, *j*) returns the entry in the i^{th} row and j^{th} column of the two-dimensional range *matrix*. If *matrix* is one-dimensional, the *j* parameter is optional.

(3) ***Populate the cells E30:E41*** using the **index** function in conjunction with the **match** function on the Product_Summary_By_Month_Table:

   a. In cell E30, type:

   "=INDEX(Product_Summary_By_Month_Table,MATCH($D30,$C$9:$C$20,0),MATCH(Selected_Product,$D$8:$M$8,0))" and press Enter.

   b. Copy this formula down to cell E41.

Note that **match** is being used *twice* here: once to identify the row index of the corresponding month, and once to identify the column index of the Selected_Product.

(4) To ***create a column chart showing the time series***:

   a. Select cells D28:E41, and from the ribbon select Insert->Charts->Column (or Alt->n->c). Select the first 2D Column chart.

   b. Right click on the chart and then click on "Select Data…".

c. Under the "Legend Entries (Series)", click on the series and select Edit. Select cells D28:E28 to be the name of the series, and click OK.

d. Under the Horizontal (Category) Axis Labels, click "Edit", select the cells D30:D41 as the "Axis Data Range", and click OK. With the chart selected, use the "Chart Tools" ribbon tabs to change the display options as you wish.

In order to compute the *partial sum of the summary values* for the Selected_Product between the selected Product_Start_Month and Product_End_Month, we first need a way to isolate the appropriate *subset* of cells within the range E30:E41. To do this, we need the **offset** function.

The **offset** function in Excel allows you to refer to a cell or a range of cells that is ***offset*** a specified number of rows and columns from a reference point. The syntax is:

offset(*reference, rows, cols, height, width*)

where *rows* and *cols* indicate the number of rows and columns, respectively, that your target range is offset from the *reference*, and *height* and *width* indicate the size (in rows and columns) of the target range you want to return. Unlike the **vlookup** function, which returns the value of a single cell only, **offset** allows you to ***extract an entire range of cells*** by specifying height and width parameters greater than 1. If you want to return the value of a single cell only, the height and width parameters can both be set to 1.

(5) To see how the **offset** function works, ***pre-populate cell K24*** with "=SUM(OFFSET($E$29,***5***,0,(***7-5***)+1,1))" and press Enter. Note that the number ***5*** is **hard-coded** to represent a Product_Start_Month = May, and the number ***7*** is **hard-coded** to represent a Product_End_Month = July, so that at present, the result should be the partial sum of the Selected_Summary for the three-month period May-July. (Verify this using quick view summarization source table.) We will come back and fix this shortly.

In order to make this **offset** function fully dynamic, note that:

- The second argument (*rows*) needs to be set as the *index of the Product_Start_Month*, since relative to the *reference* cell E29, this is the number of rows we need to "move down" before we start retrieving cell values. We can use the **match** function to achieve this.
- The third argument (*cols*) can remain at 0, since relative to the *reference* cell E29, we are already in the correct column and do not need to "move over" any.
- The fourth argument (*height*) needs to reflect the *total number of months (inclusive) between the Product_Start_Month and Product_End_Month*, since this is the number of cell values we want to add up. We can take the difference between two **match** function calls (and add back 1) to achieve this.
- The fifth argument (*width*) can remain at 1, since we only want to add up the cell values in a single column.

(6) ***Fix the entry in cell K24*** by updating the **offset** function arguments as follows***:***

a. Replace the second argument with "MATCH(Product_Start_Month,$D$30:$D$41,0)" .

k. Replace the fourth argument with "MATCH(Product_End_Month,$D$30:$D$41,0)-MATCH(Product_Start_Month,$D$30:$D$41,0)+1" and press Enter.

We are now ready to move on to the *Product Summary By Customer* worksheet. Before we begin, there are several things to note:

- This worksheet contains six ***named cell ranges*** (five single cells, and one table). To view all of the defined names in the workbook, along with their associated range addresses and properties, from the ribbon, select Formulas->Defined Names->Name Manager.
- The following named cells have had their values list-restricted using ***data validation***:
  - Selected_Summary_2 (E3) has been restricted to the same defined values as Selected_Summary on the *Product Summary By Month* worksheet.
  - Selected_Customer_Product (F17) has been restricted to the list of valid products.
  - Selected_Customer (F18) has been restricted to the list of valid customers.
- ***Text concatenation*** has been used in several places (D6, C8, E21:E24, C26) to create dynamic labels or titles.
- ***Conditional formatting*** has been used on several cell ranges to control how the number format is displayed (as plain or currency). To see the cells this affects, from the ribbon, select Home->Styles->Conditional Formatting. Click Manage Rules, and under "Show Formatting rules for", select This Worksheet.

Section 3: INDIRECT, SUMIFS, and INDEX Revisited

Recall that **sumifs**(*sum_range*, *crit_range_1*, *crit_1, crit_range_2, crit_2, ...*) is a conditional logic function that adds together the elements of *sum_range* whose corresponding entries in *crit_range_1*, *crit_range_2*, etc, respectively, meet the criteria designated by *crit_1*, *crit_2*, etc.

Given a Selected_Summary_2 in cell E3, we want the Product_Summary_By_Customer_Table to display a partition of that dataset by Product and Customer. We will accomplish this using the **sumifs** function with two *crit_ranges*: the named Product and Customer datasets.

(1) ***Populate the*** Product_Summary_By_Customer_Table using **sumifs** and **indirect**:

   a. In cell D9, enter

   "=SUMIFS(INDIRECT(Selected_Summary_2),Product,D$8,Customer,$C9)".

   b. Copy this formula across and down the table.

(2) Use a PivotTable or the AutoFilter tool on the Customer Orders dataset to verify the conditional sum results in the table for various Selected_Summary_2 values.

In the bottom half of the worksheet, for any Selected_Summary_2, any Selected_Customer_Product (F17), and any Selected_Customer (F18), we want to display the following:

(3) In cell F21, the corresponding cell entry from the Product_Summary_By_Customer_Table. This can be accomplished using either the **vlookup** function or the **index** function in conjunction with the **match** function. For instance:

"=INDEX(Product_Summary_By_Customer_Table,MATCH(Selected_Customer,$C$9:$C$13,0),MATCH(Selected_Customer_Product,$D$8:$M$8,0))"

(4) In cell F22, the ***average daily value*** of Selected_Summary_2 for Selected_Customer_Product and Selected_Customer. This can be accomplished using the **averageifs** function with the named Product and Customer datasets as *crit_ranges*:

"=AVERAGEIFS(INDIRECT(Selected_Summary_2),Product,Selected_Customer_Product, Customer,Selected_Customer)"

Section 4: Using Array Functions to Compute Summary Statistics

In cells F23 and F24, respectively, we want to display the ***single day minimum and maximum values*** of Selected_Summary_2 for Selected_Customer_Product and Selected_Customer. Unfortunately, there are no **minifs** or **maxifs** functions in Excel. Instead, we will compute the necessary values using a combination of ***array formulas*** with the Product and Customer datasets.

Unlike a regular formula or function, which returns a *single* value, ***an array formula returns an entire set of values***. To enter a function as an array formula, you must ***select the entire output range*** that you want populated, type in the function or formula, and press ***Ctrl-Shift-Enter*** (not just Enter). Once entered, array formulas are shown as being enclosed in brackets. (You do *not* actually type the brackets – they are only a visual indicator generated automatically by Excel to signify that an array formula has been entered.)

It is easiest to illustrate the concept of array formulas back on the *Customer Orders* worksheet:

(1) In cell J21, enter a temporary column label such as "Temp".

(2) Select all of the cells in the Temp column that are adjacent to populated cells in the Gross Margin column. (Shortcut: Populate the top cell with a junk function like "=5", copy it down the column by double-clicking on the square handle, then press the Delete-key.)

(3) With all of the Temp cells highlighted, type "=(Product="Alpha")" and press ***Ctrl-Shift-Enter***. The result will be an array of (TRUE,FALSE) values that indicate whether or not the corresponding entry of the named range Product (in column C) is equal to "Alpha".

(4) Now, with all cells in the Temp column still selected, press F2 to edit, modify the formula to be "=(Product="Alpha")*(Customer=1)" and press ***Ctrl-Shift-Enter***. The result will be an array of (0, 1) values that indicate whether each row is an Alpha order from Customer 1 (1) or not (0). Note that the "*" operator does *pairwise multiplication* on the (TRUE,FALSE) array values and converts each result to a number (0 or 1); hence each entry of the resulting array will be 1 ONLY if both conditions are satisfied.

(5) Again, with all cells in the Temp column still selected, press F2 to edit, modify the formula to be "=IF((Product="Alpha")*(Customer=1)=1,Order_Quantity,"")" and press ***Ctrl-Shift-Enter***. The column will now be populated with an array of order quantities (for those rows corresponding to Alpha orders from Customer 1) and blanks (otherwise).

(6) In cell J20, type "=MIN(J22:J18321)" and press Enter. The result will be the ***single day minimum Alpha Order_Quantity by Customer 1***. This example illustrates how we can populate cells F23 and F24 back on the *Product Summary By Customer* worksheet.

(7) In cell F23 of the *Product Summary By Customer* worksheet, type

"=MIN(IF((Product=Selected_Customer_Product)*(Customer=Selected_Customer)=1,INDIRECT(Selected_Summary_2),""))" and press ***Ctrl-Shift-Enter***.

Note that if you just press Enter in cell F23, an error will result. Even though the result of the final **min** function is a single value, you must press ***Ctrl-Shift-Enter*** for the array formulas in the interim calculations to be recognized.

(8) In cell F24 of the *Product Summary By Customer* worksheet, type "=MAX(IF((Product=Selected_Customer_Product)*(Customer=Selected_Customer)=1,INDIRECT(Selected_Summary_2),""))" and press ***Ctrl-Shift-Enter***.

Section 5: Creating Scalable Histograms

Finally, we would like to be able to view the values of Selected_Summary_2 for Selected_Customer_Product and Selected_Customer in a scalable ***frequency histogram***. As we have just seen, the values resulting from the array formula:

"=IF((Product=Selected_Customer_Product)*(Customer=Selected_Customer)=1,INDIRECT(Selected_Summary_2),"")"

are precisely the data source that we need for this histogram. To construct the histogram, we will use Excel's built-in **frequency** function. The **frequency** function is one of several ***array functions*** that *returns an entire set of values* (in this case, the number of Selected_Summary_2 values that fall into *each* histogram bucket). Like any array formula, the **frequency** function must be entered with ***Ctrl-Shift-Enter***.

The cells specifying the Lower and Upper Limits for the histogram "buckets" in C28:D38 are already populated (Ctrl-` to toggle to formula view). Note that the Upper Limit cells in column D are linked to the Histogram_Step_Size in cell J18, so that the user will be able to change the granularity of the histogram *dynamically*. We will eventually link the first of the Upper Limit cells (D28) to the Histogram_Start_Value in cell J17 as well, so that the user can dynamically modify the upper limit of the first histogram bucket. Finally, the "Range" column has been populated with concatenated text to serve as descriptive labels for the chart we will create.

(1) To ***complete the frequency table and histogram***:

a. Select *all* of the cells in the "Frequency" column that are adjacent to the "Range" labels (F28:F39). Type:

"=FREQUENCY(IF((Product=Selected_Customer_Product)*(Customer=Selected_Customer)=1,INDIRECT(Selected_Summary_2),""),$D$28:$D$38)"

and press ***Ctrl-Shift-Enter***. The selected cells should now be populated with frequencies.

b. To create the chart, select the data in the "Frequency" column, and from the ribbon select Insert->Charts->Column (or Alt->n->c). Select the first 2D Column chart.

c. Right click on the chart and then click on "Select Data…".

d. Under the "Legend Entries (Series)", click on "Series 1" and then Edit. Select cell C26 to be the Series Name, and click OK. Under the Horizontal (Category) Axis Labels, click "Edit", select the labels in the "Range" column to be the "Axis Data Range", and click

OK. With the chart selected, use the "Chart Tools" ribbon tabs to change the display options as you wish.

(2) To give the user more flexibility, let's ***create a spinner*** so that the step size of the histogram can be easily changed, and let's allow the user to ***specify the starting bucket of the histogram*** so that the first bucket either ends at the value entered in cell J17 or, if no value entered in cell J17, to default to the minimum dataset value in cell F23 (rounded down to the nearest multiple of 100):

a. From the ribbon select Developer->Controls->Insert. (If you do not see the Developer tab, click File, then Options. Under Customize Ribbon, make sure that "Developer" is *checked* on the Main Tabs ribbon.)

b. Select the spinner icon (the up-and-down arrow pair with no space in between) under "Form Controls", and then create a spinner next to the "Step Size" cell (J18) by holding down your left mouse button and dragging a rectangle shape on the screen. Click away from the spinner when you have finished.

c. Now, right-click on the spinner, select "Format Control", and set Minimum value = 1, Maximum value = 100, and Incremental change = 5. In the Cell Link field, select J18, and press OK. The spinner should now change the value in cell J18 by increments of 5.

d. In cell D28, type:

"=IF(Histogram_Start_Value<>"",Histogram_Start_Value,ROUNDDOWN($F$23,-2))"
and press Enter. The second parameter of the **rounddown** function specifies how many significant digits to the right of the decimal point should be rounded (so -2 means to round two places to the *left* of the decimal point -- i.e., to the nearest hundred).

Note: *The spinner control form only limits the values that can be entered into the target cell using the spinner*. A user can still manually enter a value into a spinner-linked cell that is out of the range specified by the spinner. In order to restrict the values that a cell is allowed to contain, you must use ***data validation***.