ORIE 4820: Spreadsheet-Based Modeling and Data Analysis
Service Parts Optimization
Spring 2013

During this exercise, we will begin building a decision support tool for Swifty, a regional provider of maintenance support for office equipment and specialized printing systems. This tool will enable Swifty to determine how many service parts of each type to stock at its stores to support the *Plotter* product. The primary purpose is to give you additional practice ***using VBA to implement algorithms within the Excel environment***.

The template file for this exercise is ***service-parts-optimization.xlsm***. Download a copy of the file from the course Blackboard site and *save it on your computer*. There are three worksheets in this workbook: *Store CSL Evaluation*, *Store Parameters*, and *All Store Base Stock Levels*. Today's lab will focus primarily on *Store CSL Evaluation* (Sections 1-3). Instructions are provided for you to work with the remaining two worksheets as time permits (Section 4).

<u>Topics/Tools we will cover:</u>

- ***Constructing the model "pieces"*** needed to evaluate product customer service levels (i.e., relative demand rates and fill rates for different service part types)
- ***Using greedy marginal analysis*** to determine the next "best" service part type to increment
- ***Writing code to automate the marginal analysis optimization algorithm*** using VBA
    - ***Defining, initializing, and updating variables*** within VBA subroutines
    - ***Using logical and looping constructs*** within VBA subroutines
- ***Writing code to determine optimal base stock levels for all stores*** using VBA
    - ***Defining, initializing, and updating arrays*** within VBA subroutines
    - ***Using looping constructs with arrays*** within VBA subroutines

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

<u>Background:</u>

Please refer to Monday's handout and lecture notes for an overview of Swifty's business model and a description of the service parts optimization problem it needs to address for the Plotter:

$$\textbf{minimize} \left(\sum_{\text{items } i} \sum_{\text{Stores } j} c_i \cdot s_{ij} \right) \longleftarrow \textbf{System Inventory Investment}$$

$$\textbf{subject to:} \sum_{\text{items } i} \underbrace{w_{ij}}_{\substack{\textbf{Relative likelihood that}\\ \textbf{service part } i \textbf{ will be}\\ \textbf{demanded at store } j}} \cdot \underbrace{f_{ij}(s_{ij})}_{\substack{\textbf{Immediate fill rate of service part}\\ i \textbf{ at store } j \textbf{ (depends upon the base}\\ \textbf{stock level of part } i \textbf{ at location } j\textbf{)}}} \ge F_j \longleftarrow \substack{\textbf{Service level}\\ \textbf{guarantee at}\\ \textbf{store } j} \quad \forall \textbf{ stores } j$$

$$s_{ij} \ge 0 \text{ and integer } \forall i,j$$

Since this problem is <u>*separable by store*</u>, the Store CSL Evaluation worksheet focuses on evaluating and optimizing stock level decisions for a single store. (Right now, demand and lead time parameters for Store 1 have been hard-coded in the designated cells.) Note that the problem to be solved for store 1 involves an objective function that is *linear* in the s_{ij}'s (i.e., the part stock levels) and a *single* constraint. This constraint has a *nonlinear* function of the s_{ij}'s on its left-hand side (LHS). The right-hand side is simply a constant corresponding to the service level guarantee (95%).

The worksheet Store CSL Evaluation has three areas: *Service Part Parameters*, *Customer Service Level and Inventory Cost*, and *Marginal Analysis Area*. For each service part type $i \in \{A, B, C, D, E, F\}$, the Service Part Parameters area depicts:

- The average daily demand rate for the part at the store (parts/day): λ_{ij}
- The lead time from the part manufacturer to the store: T_{ij}
- A base stock level for the part at the store: s_{ij} (These are the decision variables.)

Given these input parameters, the components needed to evaluate the LHS of the customer service constraint -- the relative demand weights $w_{ij} = (\lambda_{ij} / \Sigma_i \lambda_{ij})$ and the fill rate functions $f_{ij}(s_{ij})$ -- will be computed in the *Customer Service Level and Inventory Cost* areas. (The objective function value is already computed in this area in cell D20.) The *Marginal Analysis Area* will be used for optimization.

<u>Section 1: Computing the Customer Service Level (i.e., the LHS)</u>

Given the demand rates, lead times, and base stock levels entered in the *Service Part Parameters* area, the *Customer Service Level and Inventory Cost* area should display:

(1) In cells D24:I24, the ***average part demand over the lead time from the part manufacturer*** that the store experiences for each service part type: $\lambda_{ij}*T_{ij}$

(2) In cells D25:I25, the ***weight*** of each Plotter service part with respect to evaluating customer service. This is the ***proportion of Plotter demand at the store that is attributable to that part type***: $w_{ij} = (\lambda_{ij} / \Sigma_i \lambda_{ij})$

(3) In cells D26:I26, the ***immediate fill rate*** at the store for each service part type: $f_{ij}(s_{ij})$ The immediate fill rate for a service part is ***the likelihood that an incoming demand for that service part can be fulfilled from stock on hand***. Taken together, the facts that (a) the demand for each part type is Poisson, and (b) a base stock replenishment policy is followed, we have that that $\boldsymbol{f_{ij}(s_{ij}) = P[X_{ij} < s_{ij}]}$**, where** $\boldsymbol{X_{ij} \sim \text{Poisson}(\lambda_{ij}*T_{ij})}$. (Note that a check should be made here for $s_{ij} = 0$, since an error will result if the **poisson** function is passed a value of -1.)

(4) In cells D30:I30, the ***unit cost by service part type*** (this is input data): c_i

(5) In cells D31:I31, the ***inventory investment by service part type*** at the store: c_i*s_{ij}

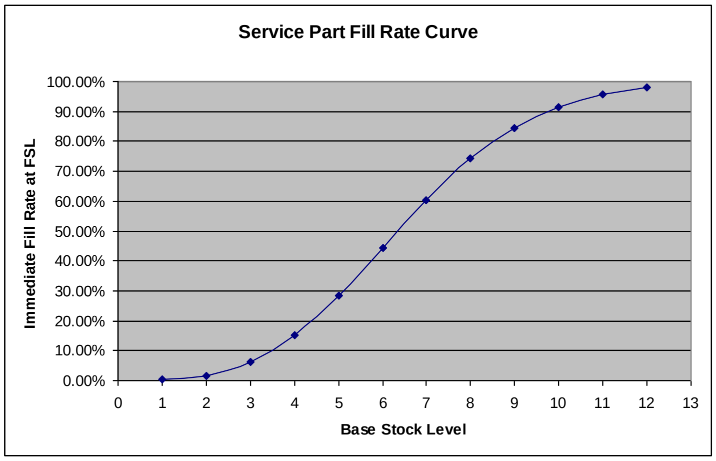
(6) In cell D20, the ***total inventory investment*** at the store: $\Sigma_i (c_i*s_{ij})$, and

(7) In cell D19, the ***customer service level*** achieved at the store: $\Sigma_i (w_{ij} * f_{ij}(s_{ij}))$

Note that cell E19 indicates whether or not the customer service target in cell H19 has been met. Also, conditional formatting has been used on cell D19 so that it turns green when the customer service target is met.

Section 2: Building the Marginal Analysis Tool

The *Marginal Analysis Area* contains a table that we will populate with meaningful metrics for each of the six service parts. ***The purpose of this area is to sequentially guide the user in choosing stock levels to increment that result in the most "bang for the buck"*** (i.e., the largest increase in customer service per dollar spent on inventory). It can be shown that if the starting stock level vector is set as $s_{ij} = \lfloor \lambda_{ij} * T_{ij} \rfloor$ for all parts *i*, then incrementing the base stock levels in the sequence indicated by a greedy marginal analysis algorithm will yield a near-optimal part stocking strategy for the store. To see why this is the case, note that in general, fill rate curves have an "S" shape, as follows:



Specifically, the fill rate curves $f_{ij}(s_{ij})$ are concave in s_{ij} for $s_{ij} \geq \lfloor \lambda_{ij} * T_{ij} \rfloor$. Setting the starting stock level vector to these values essentially ignores the left tail of the "S" curve, so that the only base stock levels considered are those at and beyond the point where the fill rate curve becomes concave (s = 6 in the graph above). For problems with this structure, a greedy marginal analysis algorithm will yield a near-optimal solution. Practically speaking, one typically would want to hold at least $\lfloor \lambda_{ij} * T_{ij} \rfloor$ of each part type anyway. Moreover, for parts with extremely low demand rates, $\lfloor \lambda_{ij} * T_{ij} \rfloor$ will be zero, so all stock levels will be considered. ***For store j = 1, the greedy marginal analysis algorithm can be summarized as follows***:

```
(1) For every part type $i$, set $s_{i1} = \lfloor \lambda_{i1} T_{i1} \rfloor$
(2) While $\sum_{\text{items } i} w_{i1} \cdot f_{i1}(s_{i1}) < 95\%$:
    (a) For every part type $i$, compute:
        $\Delta f_{i1}(s_{i1}+1) = f_{i1}(s_{i1}+1) - f_{i1}(s_{i1})$
    (b) Increment $s_{i1}$ for the part type $i$
        with maximum $\Delta_{i1}$, where:
        $\Delta_{i1} = \frac{w_{i1} \cdot \Delta f_{i1}(s_{i1}+1)}{c_i}$
```

(1) To determine the best incrementing step, we need to compute Δ_{i1} for every part type *i*. We will do this is two steps:

   a. In cell N10 of the table, compute the change to the LHS of the service level constraint if part A is incremented: $w_{A1}*\Delta f_{A1}(s_{A1}+1)$, where the fill rate $f_{A1}(s_{A1})$ = $P[X_{A1} < s_{A1}]$, and X_{A1} ~ Poisson($\lambda_{A1}*T_{A1}$). Copy the formula across to cells O10:S10.

   b. In cell N11, divide cell N10 by c_A to get Δ_{A1}, part A's current "bang for the buck," and copy the formula across to cells O11:S11.

(2) Now we are ready to populate cell N6 (Next_Part) so that it indicates *which* part type ought to be incremented next (i.e., the one giving the biggest "bang for the buck"). You can do this easily using an **index** function with **match** embedded.

Section 3: Automating the Marginal Analysis Algorithm

In this part of the exercise, ***you will complete three VBA subroutines*** that collectively automate the greedy marginal analysis algorithm described in the previous section.

First, press Alt-F11 to open the Visual Basic Editor (VBE). In the left-hand side project window, select Modules within the project hierarchy and double-click on ***Module 1*** to open it. This module contains skeleton code for the subroutines that you will be working on. ***In writing your code, NO cell addresses should be hard-coded*** (i.e., nothing like Range("H19").Value).

You will begin with `Initialize_Stock_Levels`.

(1) `Initialize_Stock_Levels`: This subroutine's purpose is to initialize the store base stock levels in row 12 to be equal to the integer floors of their respective average lead time demands (i.e., the integer floors of the corresponding values displayed in row 24). Using the skeleton code in `Initialize_Stock_Levels` as a starting point, it remains for you to:

   a. Use a "For Each" loop to cycle through Part_List and initialize the six base stock level cells in row 12 to their proper starting values using the Excel **floor** function. To invoke this function, you will need to use an assignment statement of the form:
   result_cell = Application.WorksheetFunction.Floor(*input_cell_value*, 1)

   b. When you finish coding the subroutine, place your cursor anywhere within it in the VBE window, and execute the code line by line using F8.

(2) `Increment_Stock_Level`: This subroutine's purpose is to perform a single incrementing step within the marginal analysis algorithm. Using the mechanism already in place in the *Marginal Analysis Area* of the worksheet, the code simply needs to ***check which service part type is displayed in Next_Part and increase the corresponding base stock level by one unit***. (This subroutine will be called repeatedly by `Run_Marginal_Analysis`.)

(3) `Run_Marginal_Analysis`: This subroutine's purpose is to run the entire marginal analysis algorithm once. It will do so by calling `Initialize_Stock_Levels`, then iteratively calling `Increment_Stock_Level` until the target customer service level displayed

in cell H19 is reached. Using the skeleton code in `Run_Marginal_Analysis` as a starting point, it remains for you to:

a. Include a check to ensure that the user-entered target service level in cell H19 is strictly less than 100%. If it is not, then a message box should be displayed to the user and the subroutine should terminate:

   MsgBox "*Message to display*"
   Exit Sub

b. Implement a **Do While** loop that calls the `Increment_Stock_Level` subroutine as long as the target service level has not been reached.

Section 4: Determining Base Stock Levels for All Stores

Now that you have a mechanism in place to optimize base stock levels at Store 1, your next task is to ***expand the functionality to all stores*** and to ***automate the reporting of the optimization results*** for all stores on a summary sheet.

(1) On the *Store CSL Evaluation* worksheet:

a. Modify cell D6 so that the user can select any valid store from a dropdown list.

b. Given a store in cell D6, the worksheet should ***dynamically display the demand rates and lead times for the selected store*** for all six Plotter service parts in the designated cells in rows 10 and 11, respectively. (The cells in this area currently have Store 1's values hard coded in them.) This data resides on the *Store Parameters* worksheet.

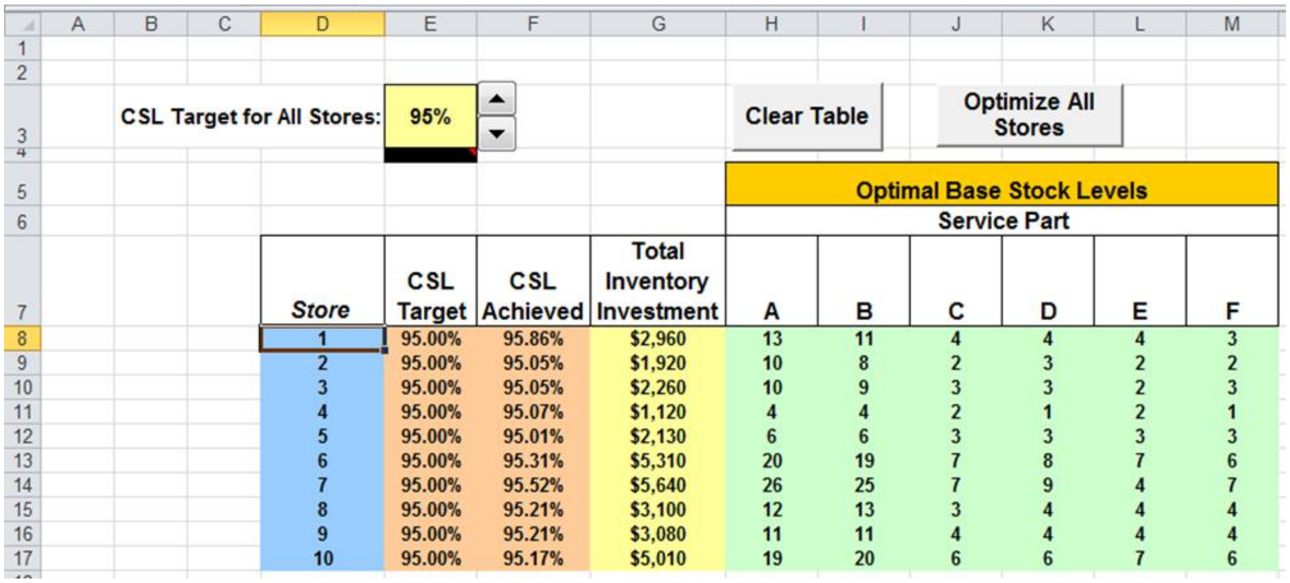
(2) On the *All Store Base Stock Levels* worksheet, the user should have access to ***two macro-linked buttons***. ***You will need to write new code for this.*** In doing so, you may assume that the first four columns of the output table will contain the designated information and that the remaining columns will contain service part stock levels, but you may NOT assume that there will always be 6 service parts, and you may NOT assume that the first cell of the output table will always be D11 (i.e., no hard-coding the number of parts or cell addresses).

a. Given the choice for a CSL Target in cell E3, the first button should ***populate the table with the optimal base stock levels for every store***. "Optimal" here means the values computed using the marginal analysis algorithm. Each table row should also include the *actual* CSL achieved and the total inventory investment required for that store at the target level. The formatting of the table should be as follows:

- All table cells should be centered and have **bold** font.
- The first column should be colored light blue (ColorIndex = 37).
- The second and third columns should be colored tan (ColorIndex = 40) and have a percentage format with 2 decimal places.
- The fourth column should be colored light yellow (ColorIndex = 36) and have a currency format with 0 decimal places.
- Columns 5 through 10 should be colored light green (ColorIndex = 35)

Upon completion of the macro, ***the state of the Store CSL Evaluation worksheet should be identical to what it was before the button was pressed*** (i.e., you must restore the user's previous settings for store, stock levels, and CSL target).

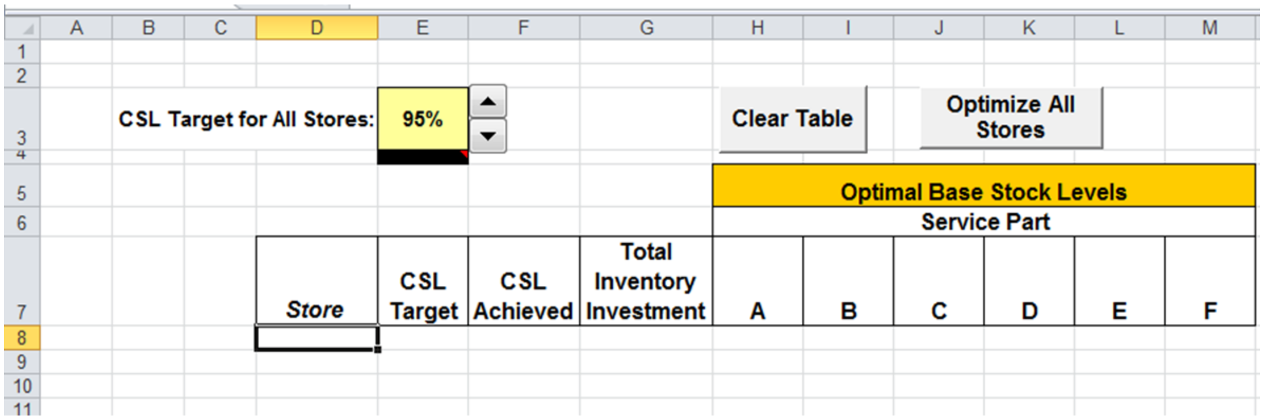
<u>Pressing the "Optimize…" button with the target shown should yield:</u>


CSL Target for All Stores: 95%

				Optimal Base Stock Levels					
				Service Part					
Store	CSL Target	CSL Achieved	Total Inventory Investment	A	B	C	D	E	F
1	95.00%	95.86%	$2,960	13	11	4	4	4	3
2	95.00%	95.05%	$1,920	10	8	2	3	2	2
3	95.00%	95.05%	$2,260	10	9	3	3	2	3
4	95.00%	95.07%	$1,120	4	4	2	1	2	1
5	95.00%	95.01%	$2,130	6	6	3	3	3	3
6	95.00%	95.31%	$5,310	20	19	7	8	7	6
7	95.00%	95.52%	$5,640	26	25	7	9	4	7
8	95.00%	95.21%	$3,100	12	13	3	4	4	4
9	95.00%	95.21%	$3,080	11	11	4	4	4	4
10	95.00%	95.17%	$5,010	19	20	6	6	7	6

b. The second button should ***clear the table rows completely*** (including colors, text formats, and number formats). The headers should remain intact, as shown below.

<u>Pressing the "Clear Table" button should yield:</u>


CSL Target for All Stores: 95%

				Optimal Base Stock Levels					
				Service Part					
Store	CSL Target	CSL Achieved	Total Inventory Investment	A	B	C	D	E	F