VBA Final Project-Elizabeth Hilton

In Search of a More Accurate Share Price

Executive Summary

Prior to business school, I dabbled in a start-up business in the racing and exercising industry. During my MBA education, I was involved in assisting with the Miller New Venture Challenge (BYU's business competition), interning with Pelion Venture Partners (a top decile VC firm in Salt Lake City), and serving as part of Cougar Capital (BYU's student-run venture capital fund). In all of these endeavors I worked extensively with revenue projections and felt that the tools to which I had access were lacking. After taking this VBA class, I knew I could do something about that problem.

An entrepreneur relies heavily on her revenue projections. The majority of her business decisions revolve around what will bring in the most revenue. In the business world, the only person more concerned about her revenue projections is perhaps the Venture Capitalist who invests in her business. The VC often uses the revenue projections to determine how much the company is worth. Unfortunately, the company value is only as accurate as the assumptions driving the projections' model. If one adjusts these assumptions even slightly they give an entirely different projection and could completely change whether or not a business is predicted to be profitable and whether or not Venture Capitalists are likely to get a return on the money they invest in the company.

Several years ago, BYU's Business Plan Competition Director created a simple model to help entrepreneurs determine their revenue projections. This rudimentary model is driven by customer acquisition rates through email. The assumptions behind the revenue are calculated by merely having the entrepreneur arbitrarily list three inputs: (1) how many people she thinks will open her emails, (2) how many of those people will click the link in the email, and (3) how many of those people will actually convert to a user of the entrepreneur's product. Then, those inputs are used to predict yearly revenues over five years. These revenue forecasts are then transmuted into a balance sheet that calculates owners' equity. This equity is used to calculate price per share each year, determining how much the company is worth. Because the entire model is driven by the entrepreneur's inputs for marketing information around customer acquisition rates, this model is very sensitive to change, making it highly subjective to the entrepreneur's assumptions around various conversion rates.

In order to make this model a better estimate of what might actually happen to the business' value over the next five years, I coded a ribbon modification that runs a Monte Carlo selection procedure of the entrepreneur's three inputs. The simulation then projects probable revenue ranges and share values to analyze the forecasts. The inputs for the simulation are recorded through a user form that gathers each marketing assumption's initial value and low and high ranges for those values. It also asks for how many iterations the user would like to run. Obviously, a more thorough and accurate analysis will require more iterations. Nevertheless, the user form has an exit button to allow the user to exit the procedure in the event that the user inputs too many iterations (e.g. near or outside the Integer variable type range) and

the processing ends up taking too long. After each iteration the balance sheet rebalances in order get the updated ownership percentages and share price. Each iteration of data is recorded, and the code creates a chart object to visually display a histogram that continually updates while the simulation is running. Incidentally, this chart is added into a worksheet and formatted entirely programmatically – this was more difficult than either creating a chart through the Excel interface or creating a chart object as its own worksheet object. The user form gives the user the option of viewing the histogram changes as each iteration runs or unchecking this option to speed up the simulation and only display the final results. Either way, the user can look at the histogram and see the likelihood of a range of various five-year share price growth rates. From these growth rates, she can determine the most likely scenario, input the data in the assumptions tab and rebalance the data. This data will now be more than just a random guess, allowing both her and the VCs to make better decisions.

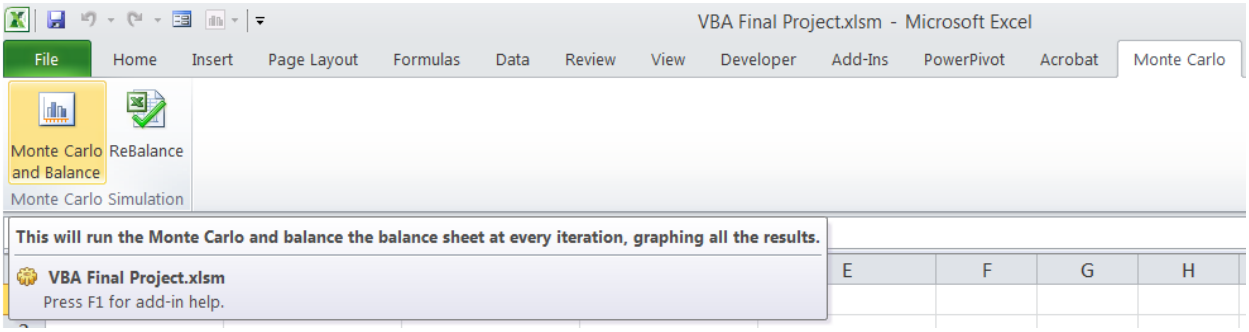
As part of creating this tool to build more accurate projections, I also fixed the existing code (only five lines of goal seek function calls) in the original model because it was clunky in that it required the user to manually run a balance sheet macro several times in order to balance the balance sheet. Now the balance sheet macro is called automatically at the end of each iteration in the simulation.

With this VBA coding, the model now gives a much more accurate forecast of revenues that in turn give a much more robust analysis of the share price. Using this code the entrepreneurs can better see if their business is likely to be profitable and VCs can better estimate their return on investment.

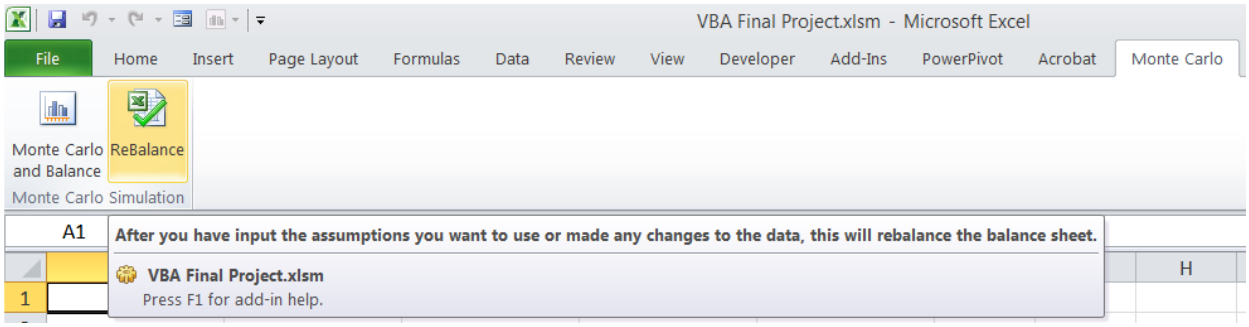
Implementation Documentation

The first step to creating this VBA model was to fix the balance sheet calculation method. In the original model, the only existing code in the entire model was written so that it required the user to run a balance sheet macro several times in order to balance the balance sheet. I started by modifying the balance sheet macro so it was contained in its own *While* loop so it could be effectively called programmatically. I also included it at the end of my simulation *For* loop so it will run during each iteration of my Monte Carlo routine.

I wanted a custom ribbon in this model. I went to the Custom UI Editor for Microsoft Office and inserted code that created a new tab called Monte Carlo. Then I created two buttons: one called Monte Carlo and Balance and the other called ReBalance:



Because I didn't want any confusion, I added a Screen Tip to each button:



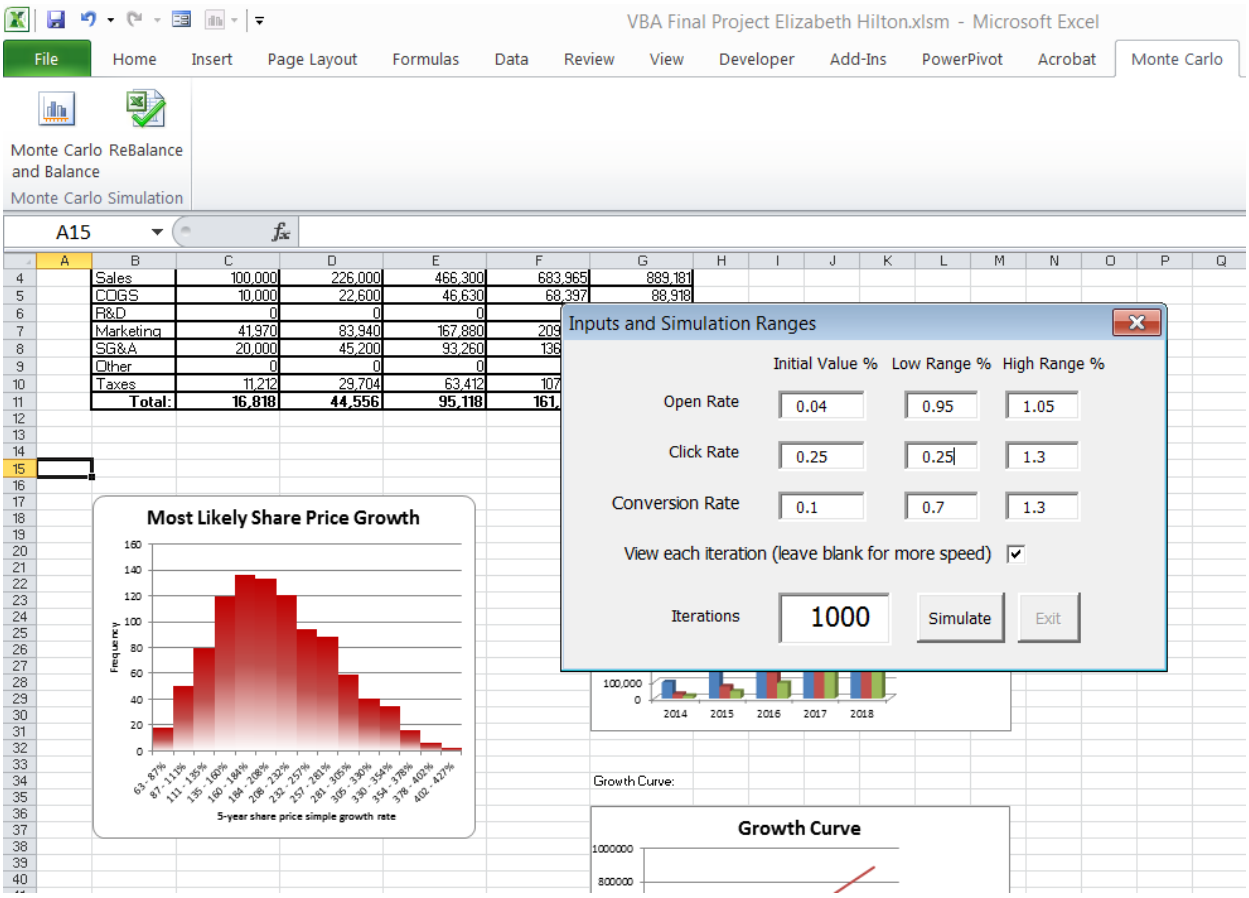
The Monte Carlo and Balance button runs the Monte Carlo simulation through a user form box (which includes calling the ReBalance logic on each iteration) and records both the input data and the output share prices in a hidden worksheet. This data is therefore persistent and leaves the model with a clearly readable histogram when it is opened later. Once the entrepreneur has reviewed the histogram, determined the validity of the assumptions, and decided on the best assumption for her business, she can change the model's inputs and click the ReBalance button. This will only balance the balance sheet and not rerun the Monte Carlo. Additionally, an entrepreneur might need to make manual alterations to the balance sheet, so the ReBalance button will also work for that case.

A user form box is the easiest way to run the Monte Carlo and define the appropriate ranges for its random selections. It is important that the user is able to set parameters for the simulations, or it will take too long to run and not be as accurate. An entrepreneur will usually know a range of marketing assumptions that she feels she can hit. The Monte Carlo will show her the profit probabilities within that range so she can make better financial projections. Therefore, the user form focuses on three marketing assumptions and asks for three inputs for each of the three marketing assumptions: Initial Value %, Low Range %, and High Range %. The initial value is the original assumption. For example, if the entrepreneur estimates that 4% of the emails she sends will be opened, then this box should contain 0.04 for the initial value input. The high and low values are the ranges over which the Monte Carlo simulation will select random numbers. For example, if the entrepreneur is very confident in the 4% open rate based on past experience, the low value could be 0.95 (which means the bottom of the random number range will be 95% of the initial value) and the high value could be 1.05 (which means the top of the random number range will be 105% of the initial value).

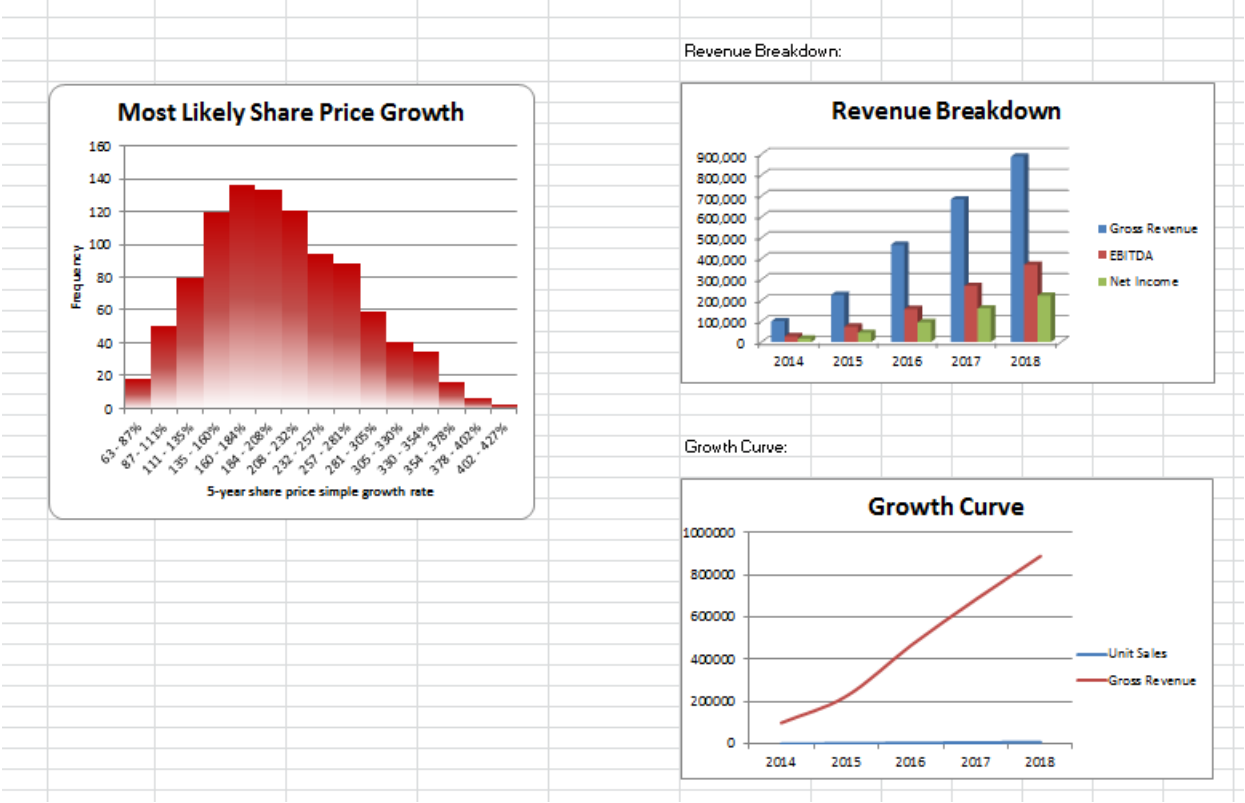
The user form loads by running UserForm_Initialize (frmSimulate_Initialize in this case specifically) and preloads the form with the model's current assumptions as well as some default ranges for the random number selection, but the user can and should edit any of these fields to her preference. Then, the user must specify how many iterations she wants to run. A good number would probably be about 1,000, but she could input any number. The number of iterations is directly proportionate to the accuracy of the projections. More iterations equal more accurate projections. I also decided to include an Exit button so that if iterations end up taking too long to populate, the user can exit the simulation. The exit function relies on a Boolean global variable that flags whether or not the iterations should be canceled, and there is a DoEvents function in each iteration to check this flag and exit the sub routine if necessary. Finally,

the user form includes a check box that will toggle screen updating for the histogram on and off. For many iterations or in other circumstances when speed is desirable, unchecking this box speeds up the simulation significantly, but the default value is set to checked because it is fun to watch the histogram update in real time throughout the Monte Carlo routine.

In my tests, I found that it takes about one minute for every 1,000 iterations when the screen updating box is unchecked and about two minutes per 1,000 iterations when checked. Leaving the box checked is preferred because the extra minute is a small price to pay to see the graphs update. However, I understand some people might not have that much time to run the iterations. A screen shot of the user form is included below:



The programmatically created histogram, revenue breakdown, and growth curve displaying the results of the last simulation are also shown below:



A single iteration of the Monte Carlo simulation uses the RandBetween function over the low to high range on each of the three input assumptions. In order to maintain good resolution on the random number selection, the Long type inputs are multiplied by 100 and rounded before calling the RandBetween function. The result is then divided by 100 again before being re-input into the model to make that iteration's projection. The iterations are recorded in the data tab (which is hidden in the model and also cleared from all previous data at the beginning of each simulation) and graphed in the graphs tab. Then the user can view the histogram and determine the validity of her assumptions and the most likely scenario. At this point she will want to alter her original assumptions if necessary, make any manual edits to the balance sheet, and either run a new Monte Carlo simulation or hit the rebalance button to get her final share prices.

Now for the fun part—the struggles with the actual code behind the simulation, buttons, user form, and chart object.

Learning and Difficulties

Creating code to insert the graphs was the most difficult part. It took a lot of research and trial and error to figure out how to do it. Originally, I thought I would just put all the data in a data area that was already coded to populate existing graphs. After talking to you, I got the impression that you wanted me to make the graphs appear as part of my VBA code. That was a much more difficult graph, and since I wanted it to appear in a logical place in the model, I couldn't just use Charts.Add to make a new tab for a chart object. It took a lot of recording macros, research, and trial and error to get the code to create

the graph and format it correctly for each simulation (as well as check for its existence and delete it if necessary before creating a new one).

Even after all that, I still could not find a way to properly configure the msoShadow property the way I wanted to – setting this property did not crash the code, but the shadow would not display – so as this was just a simple cosmetic feature that had nothing to do with the core functions of my project, I left it out to become a topic of research for another day. Visuals are obviously less important than the functionality of the model, and it still works and looks great without the shadow, but I feel like the shadow just makes it look a little more finished, professional, and trustworthy.

Assistance

I obviously used my VBA Modelers book and the internet quite a bit. Googling is really the fastest way to figure something out. If I have a question, the chances that someone else has had the same question are pretty high. I also got some help with figuring out some of my graph problems from my husband. But the only code I copied was from our previous assignments and the documents from class you loaded on Learning Suite.