Making Financially-Sound Life Decisions

A Final VBA Project by:

Nathan Davis

Winter 2016

IS 520

Executive Summary

According to a study conducted by the Economic Policy Institute in the year 2015, college graduates still face challenging employment prospects. The study found that 7.2 percent of recent graduates are unemployed, with another 14.9 percent of recent graduates being underemployed. These facts, coupled with an average student loan balance of $35,051 for 2015 graduates, indicate that students need to make wise decisions regarding majors, careers, and living locations.

Upon entering college, students dream of finding a major that matches their passions and hobbies with hopes to make their “mark” on the world. However, in making these choices, many students end up with semi-worthless degrees that do not yield promising career choices. This project is designed to help inform students about their potential salaries and expenses based on various career choices and location preferences.

This tool pulls in salary and cost of living data from reliable government sources. Then, with this information, a student can compare salaries across careers and expenses across cities to find the best options for them. The end product of this program is a PDF document showing a student a summary of their results, a sample budget, and a projected retirement account balance at the end of their working life.

Implementation

This project took several weeks and over forty hours to complete. As such, detailing every facet would be quite difficult. However, in an attempt to cover the entire breadth of how the program was created and finished, the discussion of its implementation will focus on *data collection*, *core program functions*, *program outputs,* and *other functionalities.*

Data Collection

The obvious challenge with this type of program is getting the proper data for a large amount of cities across the United Cities and for a plethora of different occupations. The next challenge would be to find some ratio or coefficient that would allow us to normalize this data in order to be able to compare one city or career against another. The goal was to create a valid comparison, rather than provide useless data.

After doing some research, data were found from a few government agencies that provided the information in question. Most of this information was easily collected using the built-in web scraping capabilities of Excel. However, the cost of living data were not organized in a table-format which was easy for Excel to interpret. Thus, for this specific data, the agent scraper was used by searching the HTML code for the desired data and then the data was collected. This process of collecting the data takes around one minute, depending on connection speed, and a progress bar is shown while the data are collected.

As shown in Figure 1, four different data tabs are created during this process. The "EmploymentSalaryData" tab contains the salary information for over 1,000 different jobs. The "CostofLivingData" tab contains cost of living information for over 120 cities across North America. However, this tab presented an issue because it simply provides cost of living ratios that are based on New York City. Thus, the ratio for NYC is 1.0 and the ratio for other cities are less than or greater than 1.0 depending on its cost of living relative to New York City.

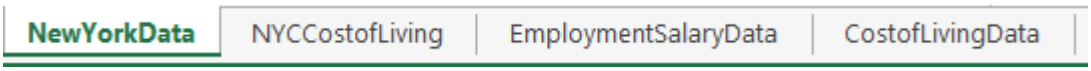


Figure 1: The output tabs from the data collected from the web

This issue created the need for the "NewYorkData" and "NYCCostofLiving" tabs. These tabs contain the web output for salary and specific cost of living items just for NYC. With these tabs, the cost of living data in *dollars* were created for NYC. With these dollar figures, the ratios provided on the "CostofLivingData" tab became useful. With all of this data, an analysis could then be done on salaries and expenses that are adjusted for living location.

With this data finally collected from the web, the next step was to collect data about the user. The program assumes that the user desires to be marry, meaning that the family size used was no less than two. Similarly, the program assumes that the user will save a reasonable amount of his or her income after expenses toward retirement. However, information such as risk tolerance, desired number of children, and age still needed to be collected in order to customize the program for each user. As shown in Figures 2 and 3, UserForms were created to collect this data.

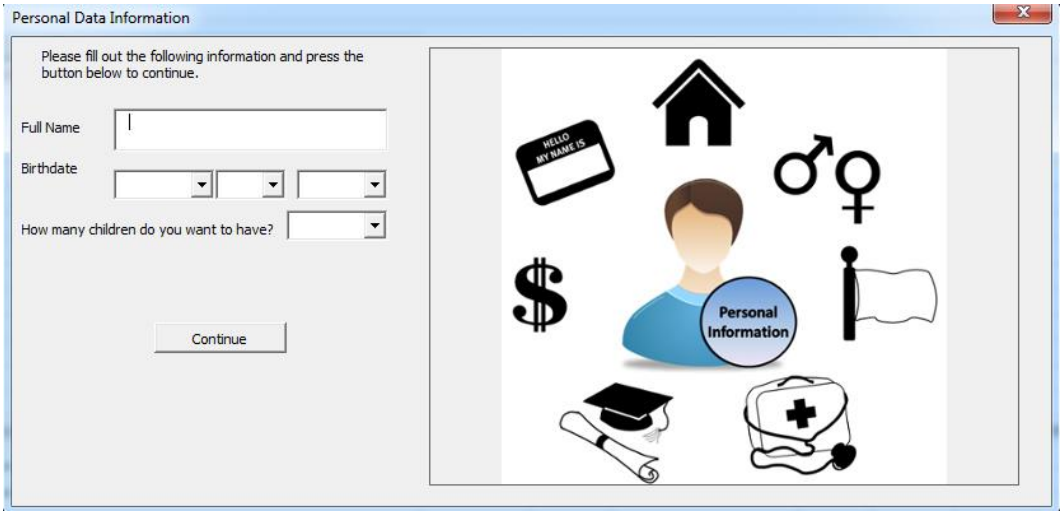


Figure 2: The UserForm created to collect personal information.

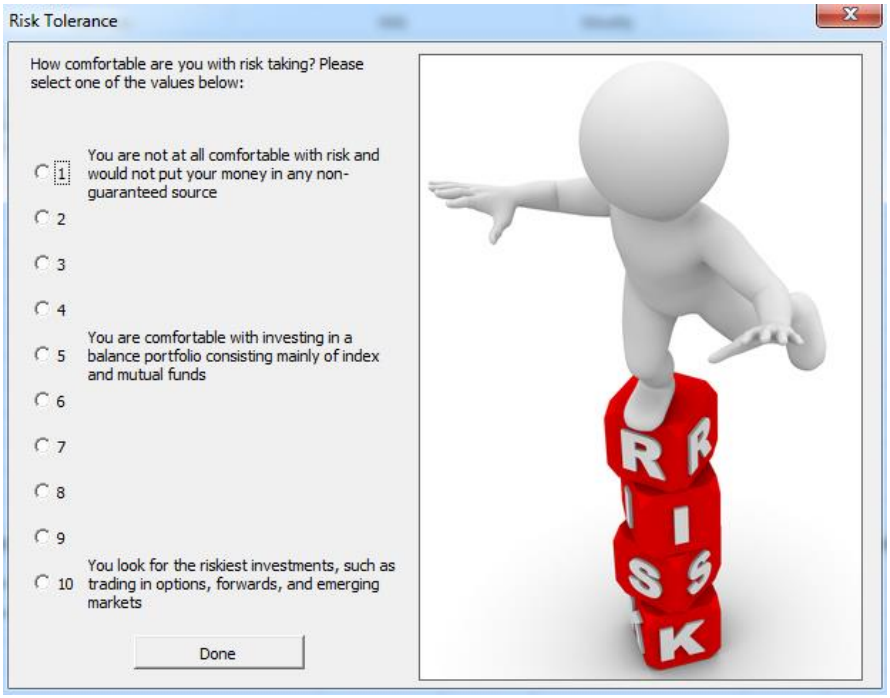


Figure 3: The UserForm created to determine the user's risk tolerance.

The information collected from these UserForms in then used to create a "Welcome" sheet that summarizes the personal information of the user. The information on this tab is referenced throughout the program in calculating age, years to retirement, family size, and return on portfolio.

Core Program Functions

With all of the data collected and stored on appropriate sheets, the next step was to determine which functionality the user wanted the program to perform. The program has three core functions:

1. Comparing Two Different Careers
2. Comparing Two Different Cities
3. Projecting Financial Performance based on One City and One Job

The user selects which functionality he or she desires the program to perform by selecting an option from the UserForm shown in Figure 4. This UserForm is presented after the user inputs his or her personal information and risk tolerance.

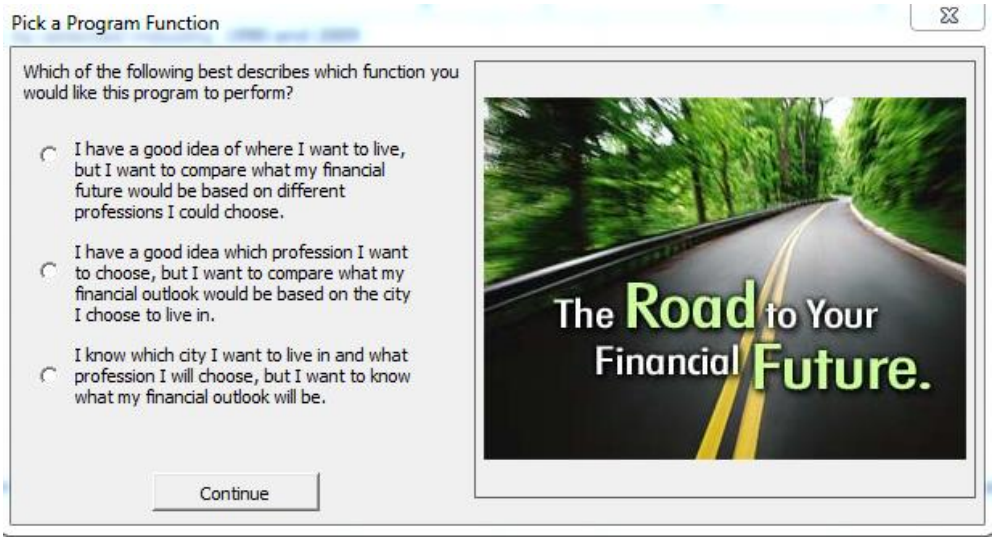


Figure 4: The UserForm created to pick program function to perform

Each of the three program function operates in a very similar manner. Thus, the process for the first function, comparing two different careers, will be shown. However, each function draws on similar programming and has similar outputs.

Upon selecting the first option, a macro called “InsertCompareProfessionsSheet” is called that inserts the sheet shown in Figure 5. The key elements of this sheet are three different Combo Boxes that are inserted. Each of the boxes is linked to the list of either cities or professions that are found on the tabs that were downloaded from the web. The user then needs to select the professions that he or she wants to compare and in which city those jobs would be located.

Once the user has made a selection in each of these drop-down combo boxes, a macro is run entitled “CompareProfessions” that performs all of the calculations based on the information provided by the user on this sheet.

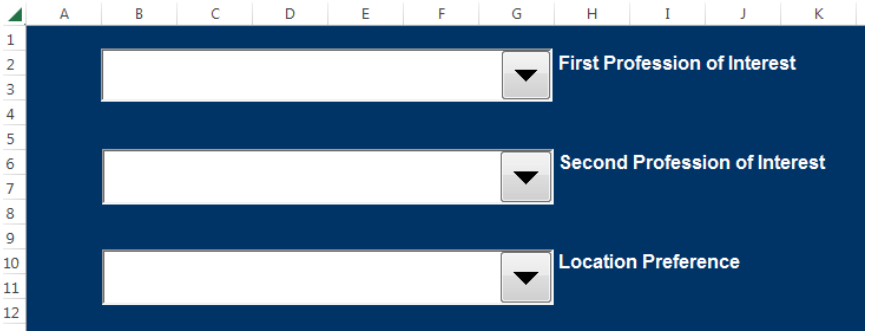


Figure 5: The “Compare Professions” worksheet

The conceptual process behind this macro is quite simple. The first step is to collect and store the information about the careers and city that the user selected in variables such as “FirstProfSalary” and “SecProfSalary.” In an effort to simplify and organize this code, several functions were created to assist in processing this information. For example, in order to provide an accurate financial outlook, taxes needed to be figured into the equations. Thus, functions entitled “calculatetaxes” and “ficataxes” were created that calculate the federal and withholding taxes, respectively.

With accurate salary and tax information, the macro then determines the expenses based on the location preference. A value is stored for groceries, other expenses, and mortgage payments. This information is then used to calculate a net income level for each month. If one of the selected professions does not produce enough income to cover expenses, a message box is used to inform the user that such is the case. However, assuming a positive net income for each month, a time value of money calculation is performed.

Earlier in the program, the user indicated a level of risk tolerance. If the user indicated an extremely low risk-tolerance level, a rate of return of 1% is used, indicating the user would keep all extra earnings in savings and CD accounts. If the user indicated a moderate risk-tolerance level, an average rate of return of 10% is used. If the user indicated an extremely high risk-tolerance level, a random return between 0.1% and 20% is selected by the macro. This rate of return is then used to determine the future value of savings at retirement based on the user’s current age and years to retirement.

Many other calculations are involved in this process, but the aforementioned functions are the major highlights. However, after having performed all of these calculations, the macro is ready to present its findings to the user and export the data.

Program Outputs

The summary of the findings is sent to a sheet created by the macro entitled “Result.” The sheet is initially formatted per the standard Excel settings, but then a macro entitled “FormatCareerComparisonSheet” is run that formats the sheet in a more visually-appealing manner. After the formatting is complete, a macro entitled “SaveasPDF” is called that prompts the user to pick a location on his or her computer where to save the results file as PDF, as shown if Figure 6. The default file name is “Result” with the current date. Upon picking the location, the PDF document is opened that shows the user his or her results (See Figure 7).

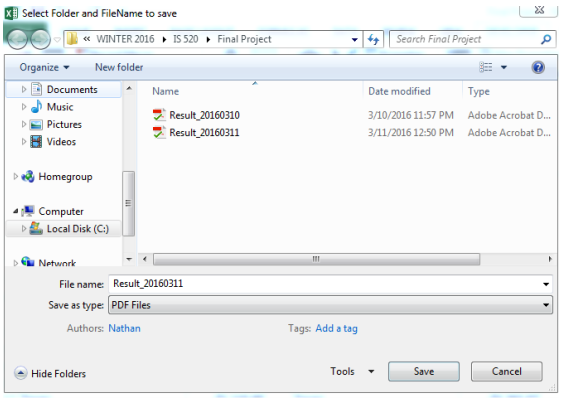
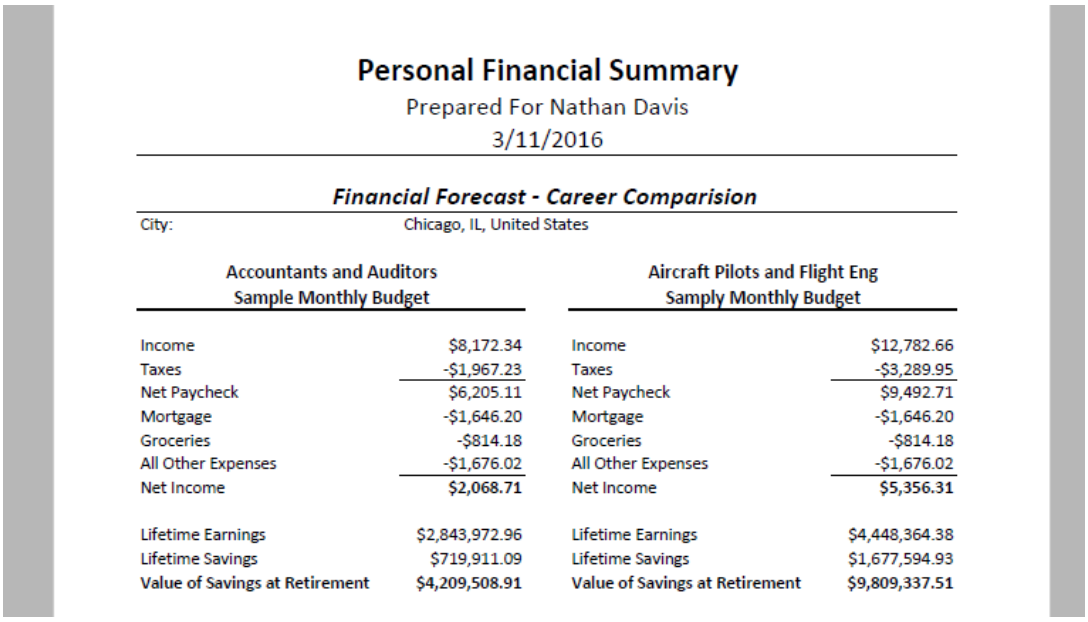


Figure 6: The dialog box to save the results as PDF



Personal Financial Summary

Prepared For Nathan Davis

3/11/2016

Financial Forecast - Career Comparision

City: Chicago, IL, United States

Accountants and Auditors Sample Monthly Budget		Aircraft Pilots and Flight Eng Samply Monthly Budget	
Income	$8,172.34	Income	$12,782.66
Taxes	-$1,967.23	Taxes	-$3,289.95
Net Paycheck	$6,205.11	Net Paycheck	$9,492.71
Mortgage	-$1,646.20	Mortgage	-$1,646.20
Groceries	-$814.18	Groceries	-$814.18
All Other Expenses	-$1,676.02	All Other Expenses	-$1,676.02
Net Income	**$2,068.71**	Net Income	**$5,356.31**
Lifetime Earnings	$2,843,972.96	Lifetime Earnings	$4,448,364.38
Lifetime Savings	$719,911.09	Lifetime Savings	$1,677,594.93
Value of Savings at Retirement	**$4,209,508.91**	**Value of Savings at Retirement**	**$9,809,337.51**

Figure 7: The PDF output of the career comparison process

Other Functionalities

Initially, the thought was that the program would end there. However, after running the program several times, it became clear that a user might want to switch between program functionalities, such as starting with comparing cities but then switch to comparing professions.

In an effort to accommodate such capability, a custom ribbon was created to allow the user to perform these functions without having to start from the beginning and wait for the web queries to re-run. This ribbon, as shown in Figure 8, has several different options for the user.

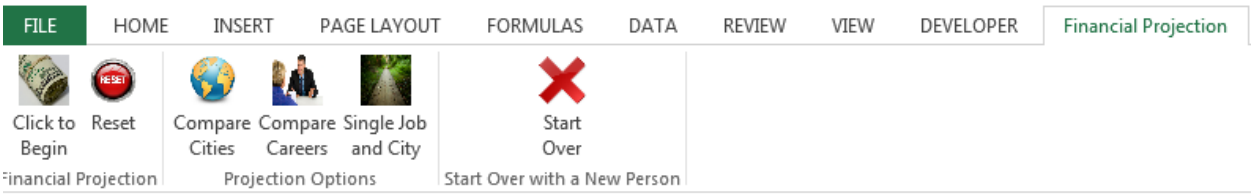


Figure 8: The custom ribbon designed to give the user more control of the program

Three main groups exist on this ribbon and each group serves a different purpose and each group has distinct buttons. The “Click to Begin” button is the initial button a user will select upon opening the file. The “Reset” button deletes the results and allows the user to move forward with a different functionality. However, while it hides the data sheets, they are not deleted so a new web query is not needed.

The group entitled “Projection Options” allows the user to perform each of the three core functionalities of the program. However, if these buttons are selected without first having clicked on the “Click to Begin” button, a message box appears directing them to do so. Thus, this group is useful only *after* the user initially enters their personal information and the data are collected from the web.

The last group is entitled “Start Over with a New Person.” The situation may arise where one person uses the program, but then another person would want to use it afterward. Thus, this “Start Over” button deletes all of the personal information previously entered and allows a new user to start the program from scratch. Thus, as has been shown, this customized ribbon is dynamic and allows the user more control of the program.

Learning and Conceptual Difficulties

My goal with this project was to create something that would be useful for myself and for other students. This was a challenging project for me, but I learned an immense amount as I completed it. I graduate at the end of this semester, so my goal was to have this project done early. This goal perhaps created the largest amount of difficulty because I was attempting to do things before I had even learned them in class. For example, I wrote user forms and created a custom ribbon before the class lecture and did so by simply reading the book and figuring it out on my own. While this was difficult, it taught me that the best way to learn VBA programming is simply to do it and practice it. The biggest lessons I learned from this project are the following:

1. The Power of VBA and its Usefulness
    a. I am starting a job with Deloitte in a few months that revolves around tax-software programming and tax technology solutions. I did an internship for this company last summer and almost my entire internship was spent trying to work in Excel. After finishing this project and creating something in Excel that was simply an idea, I am extremely confident that these skills will prove useful in my job. This project has given me a significant competitive advantage.

2. Many Paths Exists to Accomplish the Same Goal
    a. I spent many hours on various portions of this project trying to get the program to behave exactly how I wanted it to. However, what usually occurred after those hours is that I realized there was an easier way to accomplish what I was trying to do. While this seems like a basic idea, it taught me not to just “jump into” programming. Rather, brainstorming and thinking conceptually about what I want to accomplish is extremely helpful. I could have saved myself some time if I would have drawn a map of exactly what I wanted to do. The project made it clear to me that if I know the “what,” the “how” would be easy to determine.

Overall, this project and this class have been beyond useful and I look forward to furthering my VBA knowledge and skillset through study and practice.

Assistance

This project was created and written by me. I did use the textbook and other sources for clarification on certain ideas. The code that I received help with beyond a simple clarification is outlined below; all other code was authored by myself for this specific project.

1. “SaveasPDF” Macro: I searched Google for VBA code that would save the Active Workbook as a PDF document. While I made adjustments to the code that I found, the outline that I found was extremely useful.
2. Progress Bar: Dr. Gove Allen assisted me in the creation of the progress bar that is shown and updates as the web queries are performed.