EXECUTIVE SUMMARY

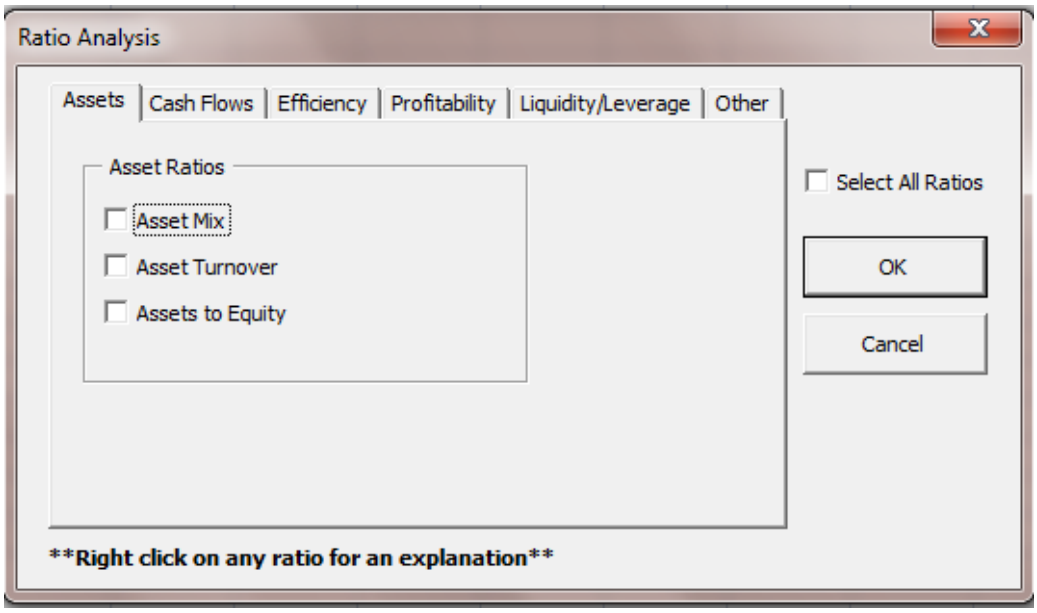
Financial ratios are commonly used to analyze any given company's financial strength and ability to continue as a viable company. Financial analysis benefits many different groups, such as management, company employees, investors, banking institutions, internal audit, and external audit. Since there are hundreds of ratios used in financial analysis, it is often difficult to remember all of the different formulas. Even if one does have all of the necessary formulas memorized, it can be time consuming to calculate all of the different ratios manually.

This project is an Excel Add-in built to simplify the financial statement analysis process through a series of user forms. The first user form allows the user to choose from 28 of the most common financial statement ratios. Based on which ratios the user selected, the second form displays the inputs required to calculate the selected ratios. This form also allows the user to attempt to auto-fill the inputs based on any financial statements in the active worksheet. After validating the data inputs to ensure accuracy and checking the reasonableness of ratios calculated, the user then decides where in the active workbook to place the data output.

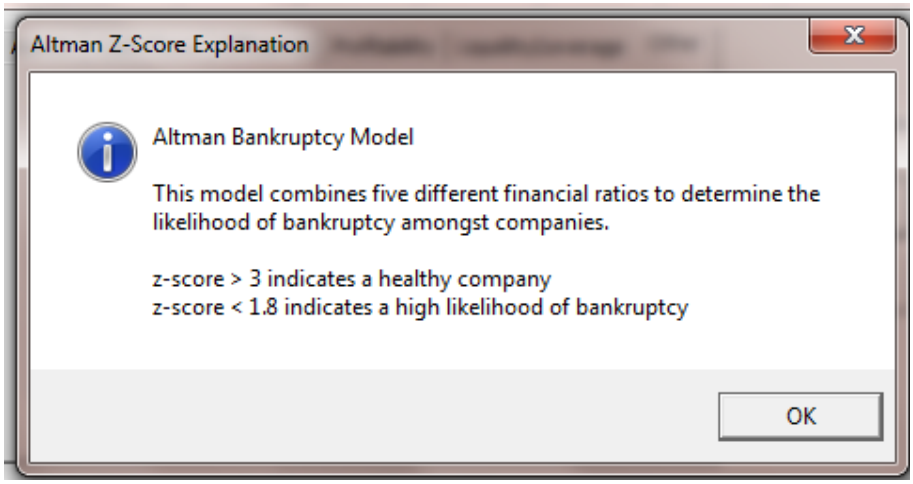
EXCEL ADD -IN

After downloading and installing this add-in, the user can run the program by clicking on the "Ratios" button, located under the Data tab in a custom group called "Ratio Analysis."

RATIO OPTIONS FORM



As can be seen in the screenshot above, the first user form presents the user with all of the ratio options to choose from. The multi-page format divides all 28 ratios into their respective categories. When the user form is initialized, the focus is set to the "Asset Mix" ratio in the "Assets" category. If the user right-clicks on any ratio, a message box is displayed giving the formula used in the calculation of that ratio and an explanation of the ratios. For the more complex models such as the Altman Bankruptcy Model and the Beneish Fraud model, the actual formula is not included in the description. An example of one of these message boxes is listed below.

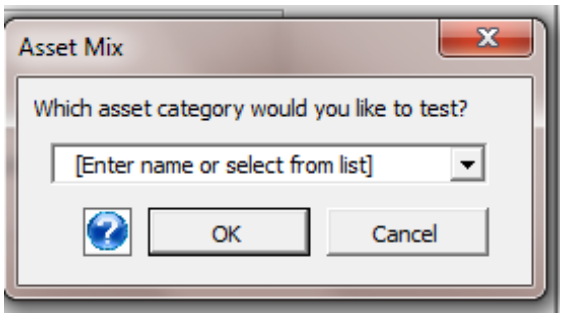


The user is also given the option to "Select All Ratios." As the name implies, this will select all of the checkboxes. If the "Select All Ratios" is deselected, all of the checkboxes will be deselected.

If the user selects the "Cancel" button or the "X" in the upper –right corner, the user form is unloaded and the project code is finished using an "End" statement.

If the user selects the "OK" button on the Ratio Analysis form, the next form is then set up. Specifically, for each ratio selected, the corresponding labels and textboxes for the required inputs are enabled and properly formatted (See the Inputs User Form section for more details). If the user has not selected any ratios when the "OK" button is clicked, the user will be asked to select at least one ratio.

Asset Mix User Form



If the user selects the "Asset Mix" ratio, an additional user form is displayed. Asset mix is the percentage of any given asset category compared to total assets. The user is asked which asset category it would like to use for this ratio calculation.

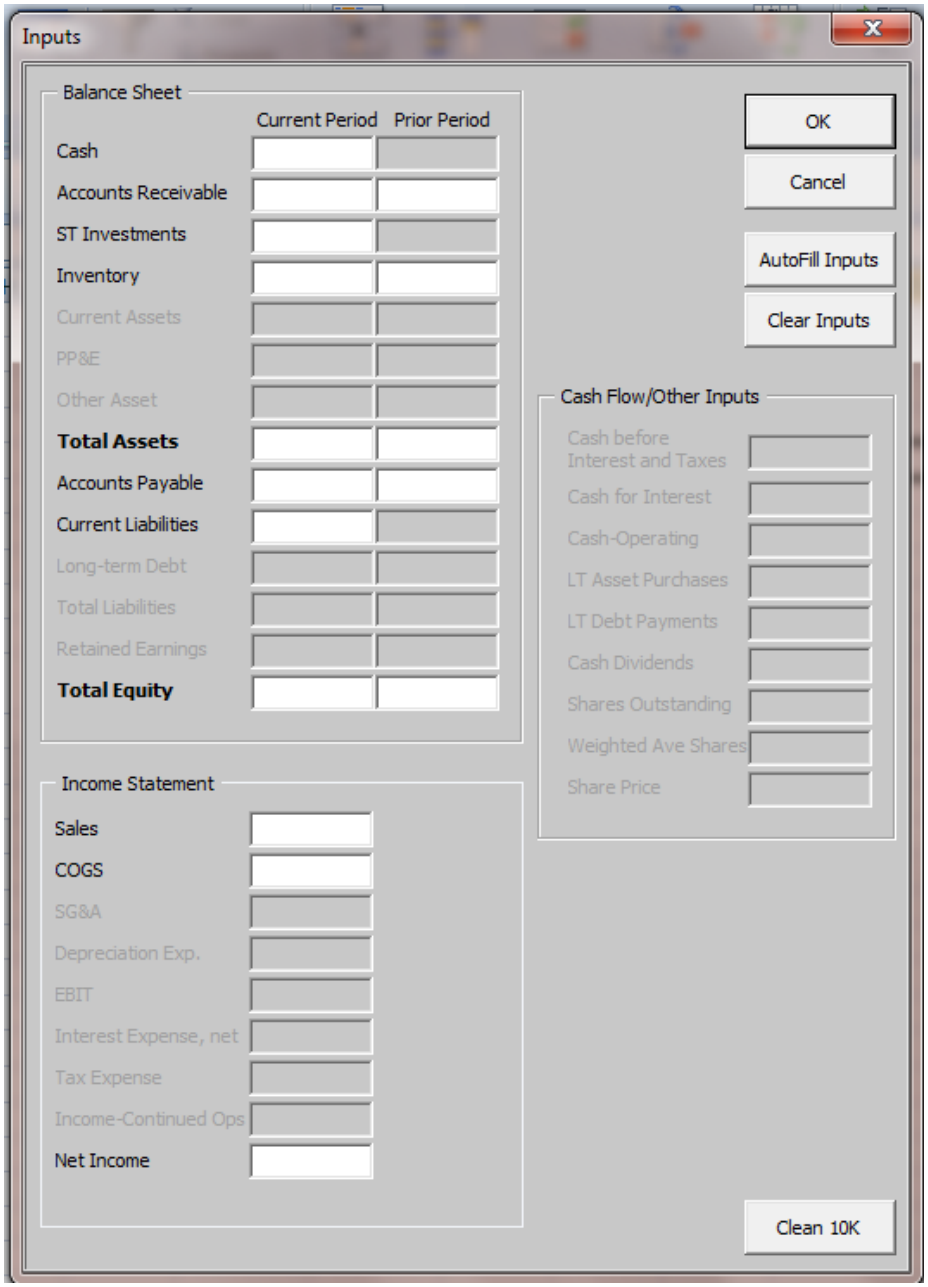
As the screenshot on the left shows, the user is asked to enter the name of an asset category to test. When the user clicks on the combo box, the preset text will be erased, allowing the user to enter in the name of the asset category to test. The user can also select an asset category from a list of commonly used assets.

If the user clicks on the image of a question mark, a message box will be displayed explaining the asset mix ratio. The "Cancel" button and "X" in the corner will both unload the Asset Mix form and return to the Ratio Options form.

The "OK" button verifies that the user has entered a value in the combo box and that the value is not numeric. Similar to the Ratio Options "OK" button, it then enables and formats the corresponding textboxes and labels on the "Inputs" user form.

Inputs User Form

The “Inputs” user form displays all of the potential ratio inputs, grouped into three different sections: Balance Sheet, Income Statement, and Cash Flow/Other Inputs. The inputs are presented in an order that is typical of most financial statements. By default, all of the input labels and text boxes are a gray color and the text boxes are disabled. The “Ratio Options” and “Asset Mix” forms change the color of the labels and text boxes and enable only the text boxes that correspond to the ratios the user selected. One important aspect of this user form is that the user is allowed to click on the active worksheet. Since some of the inputs, such as “Earnings Before Interest and Taxes” (EBIT), often require calculations, the user can use Excel to do the calculations and then click back on the user form to input the data.



CLEAN 10K BUTTON

The easiest way to obtain financial statements for public companies is to use a web query to import the balance sheet, income statement, and cash flows statement from the company's annual report (form 10K) listed on the sec.gov database (See http://sec.gov/edgar/searchedgar/companysearch.html). However, the formatting of the web query data is not conducive to financial statement analysis. The following shows the partial results of a web query:

	Year Ended May 31,		
	2011	2010	2009
	(In millions, except per share data)		
Revenues	$ 20,862	$ 19,014	$ 19,176
Cost of sales	11,354	10,214	10,572
Gross margin	9,508	8,800	8,604
Intangible and other asset impairment (Nc	—	—	202
Interest expense (income), net (Notes 6, 7	4	6	(10)
Other (income), net (Note 17)	(33)	(49)	(89)

In order to facilitate data analysis, the "Clean 10K" button will clean up the data. Specifically, it will take off any leading and trailing spaces for each cell in the used range of the active worksheet. It will delete the cells that only contain "$" or only contain ")". It will add a close parenthesis to the negative numbers: the "(33" above will become "(33)". Any extra blank columns will be deleted. The following is the result of the "Clean 10K" button:

	Year Ended May 31,		
	2011	2010	2009
	(In millions, except per share data)		
Revenues	20,862	19,014	19,176
Cost of sales	11,354	10,214	10,572
Gross margin	9,508	8,800	8,604
Intangible and other asset impairment (Note	0	0	202
Interest expense (income), net (Notes 6, 7 a	4	6	-10
Other (income), net (Note 17)	-33	-49	-89

If the data is already determined to be "clean", pressing the "Clean 10K" button will not do anything.

CLEAR INPUTS BUTTON

The Clear Inputs button clears all of the values in the text boxes and resets the label values to "Current Period" and "Prior Period." Any textboxes that have been highlighted yellow are returned to the default white. (See Data Validation and Verify Inputs Form below for information on why text boxes are highlighted yellow)

CANCEL BUTTON

The "Cancel" button and the "X" in the top right corner unload the current form and end the code using an "End" statement.

AUTOFILL BUTTON

The "Autofill" button attempts to automatically fill in the enabled inputs based on the data in the active worksheet. If the user has already entered data into a textbox, this feature will *not* overwrite the value in the text box. If the Autofill does not appear to have worked correctly, users are encouraged to clear all of the inputs, press the "Clean 10K" button, and try the Autofill again.

The first step the Autofill does is to determine in which order the financial statement data is presented on the spreadsheet. That is, some companies will display financial data with the years in ascending order (2008 - 2009 - 2010) and some will display the data in descending order (2010 - 2009 - 2008). The Autofill will then replace the balance sheet labels "Current Period" and "Prior Period" with the corresponding years.

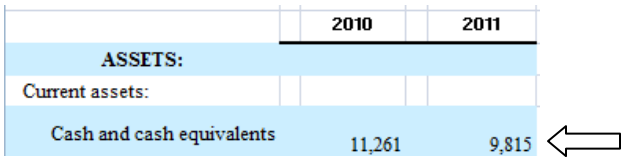
After determining if the years are presented in ascending or descending order, the Autofill uses Excel's Find function to attempt to locate the labels that correspond to each input. Because financial statements have a wide variations in wording, if the Find function does not find a particular search criteria, it will try again based on different criteria. For example, if the Find function does not find the term "Sales," it will then search for "Revenues." After locating the label, it then locates the data directly to the right to fill in the appropriate values for each input. The following code is an example of how the search finds the current period "Cash" value.

```
If .txtCash.Enabled And .txtCash = "" Then .txtCash = Cells.Find(What:="cash and", After:=Range("A1"),
        LookIn:=xlFormulas, LookAt:=xlPart, SearchOrder:=xlByColumns, SearchDirection:= _
        xlNext, MatchCase:=False, SearchFormat:=False).End(xlToRight).Value
```

The following balance sheet clip (from Apple, Inc.) illustrates the functionality of the code above. In this case, the years are in descending order (2011 – 2010). The find function will locate "cash and" and then move to the data on the right to fill in the current period "Cash" value of 9,815. The code for the prior period is very similar, except that it moves to the right twice to find the value of 11,261 (line of code ends with ".End(xlToRight).End(xlToRight).Value").

	2011	2010
ASSETS:		
Current assets:		
Cash and cash equivalents ⇨	9,815	11,261

In the case where the years are presented in ascending order (2010 – 2011 - 2012) as shown below, the same find function is used, but it locates the data on the far right for the current period, and moves to the left. The end of the line of code would be ".Offset(0,8).End(xlToLeft)" to locate the value of 9,815.



	2010	2011
ASSETS:		
Current assets:		
Cash and cash equivalents	11,261	9,815

LOCATION FUNCTION

Because Excel's find function does not entirely meet the needs of the Autofill, there is also a custom function called Location that is used in the Autofill. The Location function returns the address of the cell that contains the search criteria. This function allows a user to be able to search for multiple search terms, as well as specify text not to be included in the search result. For example, one of the inputs is "Income from Continuing Operations." Often, there is also a line item called "Income from Continuing Operations Before Income Tax." Also, some companies use the phrase "contin*uing* operations," and some use "conti*nued* operations." The location function could be used to find a cell that contains "continu" but **not** "tax."

The following shows the arguments for the location function:
Function Location(***SearchTerm1*** As String, Optional ***Exclusion1*** As String, Optional ***Exclusion2*** As String, Optional ***SearchTerm2*** As String, Optional ***SearchTerm3*** As String, Optional ***ExactMatch*** As Boolean) As String

This is an example of how the location function is used to find the "Income from Continuing Operations" input described above:

```
.txtIncomeContinued = Range(Location("continu", "tax")).End(xlToRight).Value
```

Because of a wide variation in financial statement wording and formatting, the Autofill will not be able to fill in all of the values. The user will then have to enter the remaining values. For more information on the location function, the code in its entirety is posted in the Appendix.

OK BUTTON

The "OK" button on the Inputs form validates user inputs and then stores the inputs as variables. It then calculates all of the different ratios, checks to see if the ratios are within a reasonable range, and then continues to the Output form assuming the ratios are reasonable. If one or more or the ratios are outside a normal range, the "Verify Inputs" form is displayed before continuing to the Output form.

DATA VALIDATION

Data validation checks for the following for enabled inputs:

- ✓ All inputs must be numerical values
- ✓ No inputs may be left blank
- ✓ Balance sheet accounts (except retained earnings and total equity) must be greater than or equal to zero.
- ✓ Most income statement accounts and many of the cash flow inputs must be greater than or equal to zero.
- ✓ The sum of current asset components must be less than or equal to total current assets
- ✓ The sum of current liability components must be less than or equal to total current liabilities
- ✓ The sum of all assets must be less than or equal to total assets
- ✓ The sum of all liabilities must be less than total liabilities
- ✓ Total assets must equal the sum of total liabilities and total equity

As soon as the data validation finds an invalid input, that input is then highlighted in yellow on the user form. As the user changes the value of the invalid input, the color is returned to white. The user can then press the "OK" button again to continue validating all other inputs. The user cannot continue until all inputs are considered valid.

STORE INPUTS AS VARIABLES

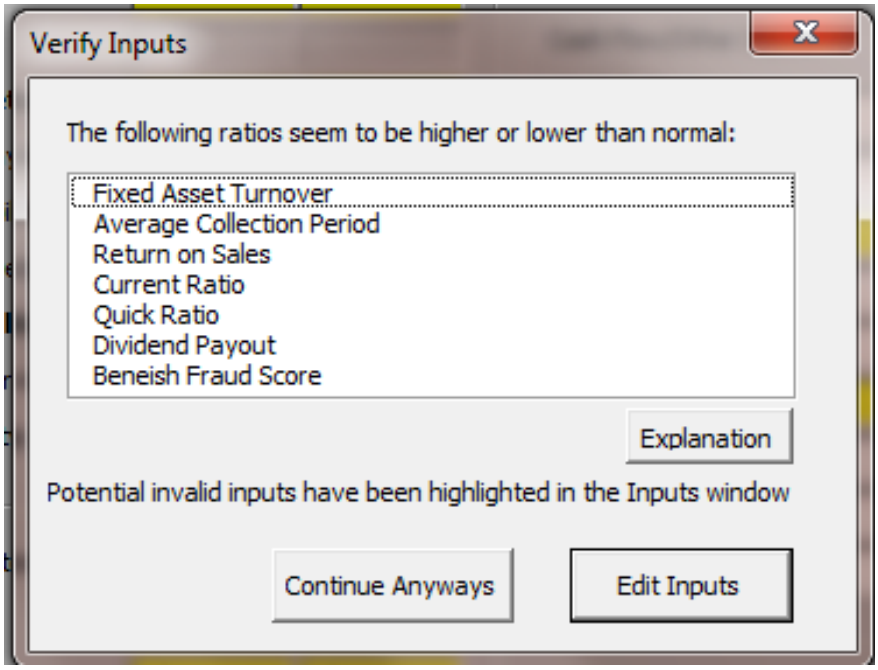
After inputs are validated, the inputs are stored as public variables before calculations are performed. Most of the variables are stored as a currency type, with the exception of shares outstanding and weighted average shares outstanding, which are stored as type single. The purpose of storing inputs as variables is twofold. First, all inputs are now in the correct format. That is, instead of numbers stored as text, the inputs are now of type currency. Second, the "Calculations" sub procedure is now much easier to read and edit than it would be trying to do calculations directly from the text boxes.

CALCULATE RATIOS

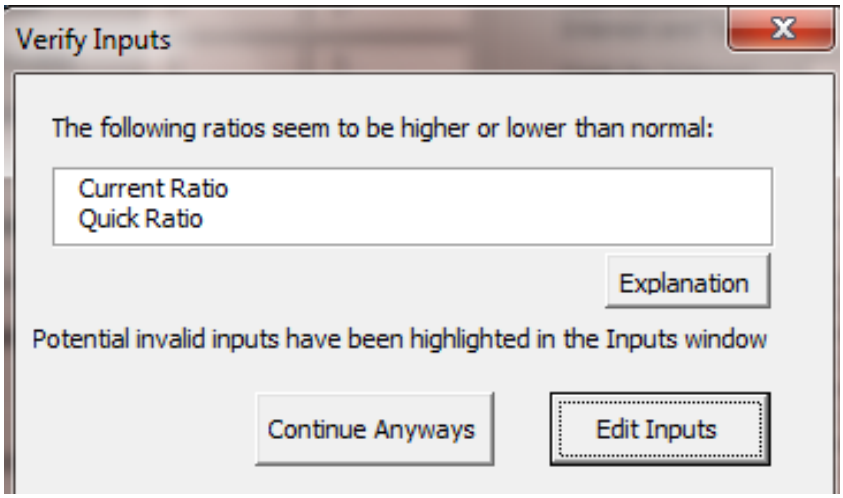
Next, all ratios selected by the user are calculated and stored as "single" type public variables.

VERIFY INPUTS FORM

After the ratios have been calculated, the ratios are checked to make sure they 1) do not contain an error, such as division by zero, and 2) fall within a reasonable range for that ratio. This could happen due to the user entering in the wrong numbers for any given input or it is also possible that the "Autofill" may have found the wrong value. The names of ratios considered outside of a normal range as well as the corresponding explanation are stored in two single dimension arrays. If any given ratio does not pass this reasonableness test, the text box inputs that correspond to the "unreasonable" ratio are highlighted yellow in the inputs form and the following form is displayed.



When this form loads, the list box and the form itself will resize according to how many ratios are to be displayed as seen in the following screenshot.

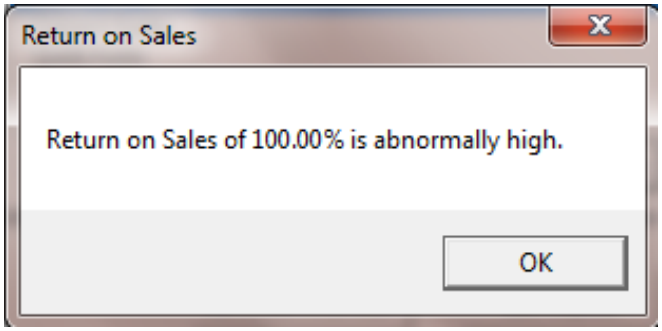


"X" IN THE TOP-RIGHT CORNER

In this case, the "X" in the top-right corner has been disabled. The user will be prompted to use only the buttons on the form.

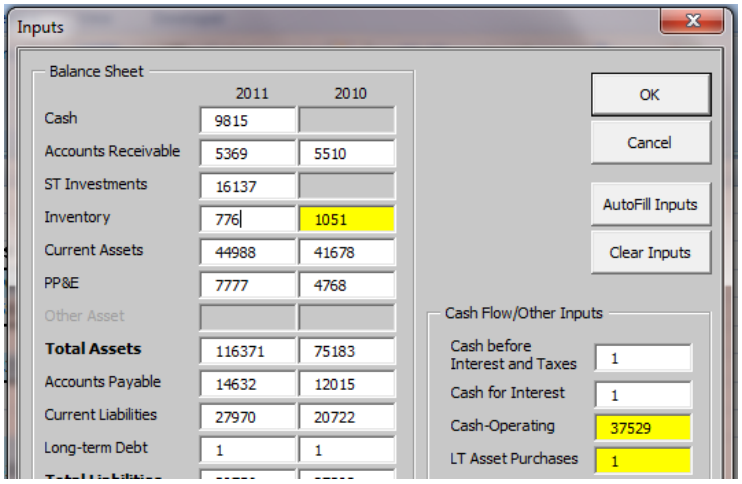
EXPLANATION BUTTON

The user can click on any ratio to get an explanation of why the input is considered invalid. This explanation could be that the ratio is abnormally high, abnormally low, or that the ratio could not be calculated because the formula results in division by zero.



EDIT INPUTS BUTTON

The user can click on the "Edit Inputs" button to return to the Inputs form. When the user changes one of the text boxes now highlighted in yellow, the text box turns white, indicating that it has been fixed. When the user presses "OK" again, all of the text boxes are restored to the default white. The application again checks the new inputs for validity and checks the reasonableness of each ratio.

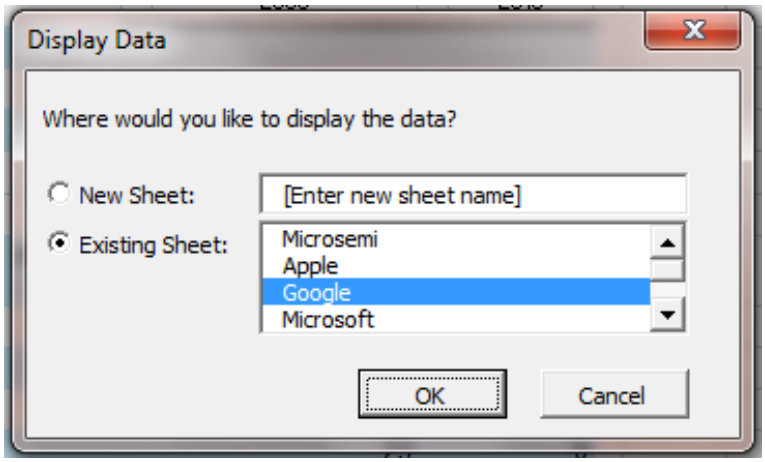


CONTINUE ANYWAYS BUTTON

If the user selects the "Continue Anyways" button, the program will ignore the fact that ratios appear to be unreasonable or result in an error. This button is provided in case 1) a company really does have an abnormally high or low value for one of the ratios or 2) the user simply does not want to take the time to go back and fix the error. After this button is clicked or after all of the inputs have been fixed and the user presses the "OK" button on the Inputs form, the "Display Data" form will be displayed.

DISPLAY DATA FORM

This is the last of the user forms in the application, and as the name suggests allows a user to determine where he or she wants the data output to be displayed.



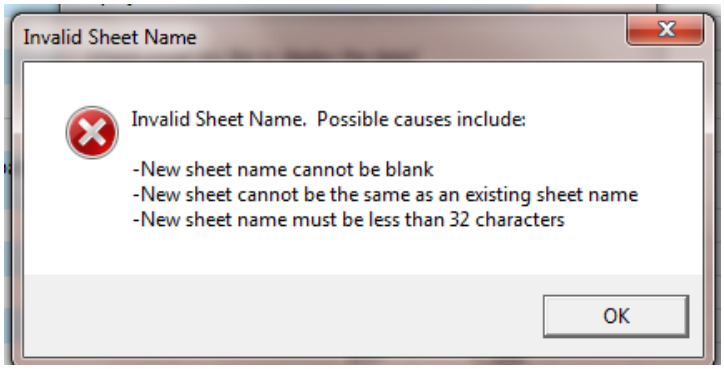
When this form loads, by default, the "Existing Sheet" option is selected, and the active worksheet is selected in the list box. If no workbooks are open, a new workbook will be created. If the user clicks on the "[Enter new sheet name]" text box, the contents are cleared, the "New Sheet" option is selected, and the user can then enter a new sheet name.

CANCEL BUTTON

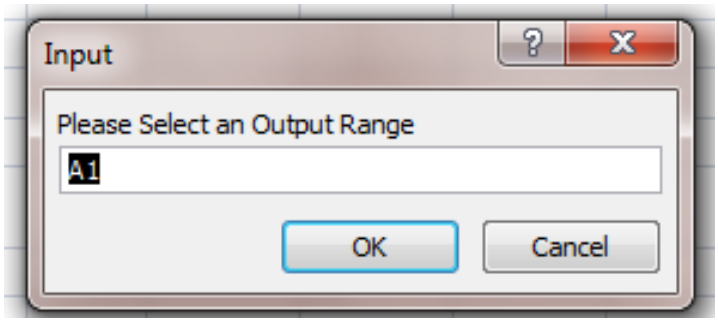
The "Cancel" button and the "X" in the top right corner of the user form will simply return the user to the "Inputs" form.

OK BUTTON

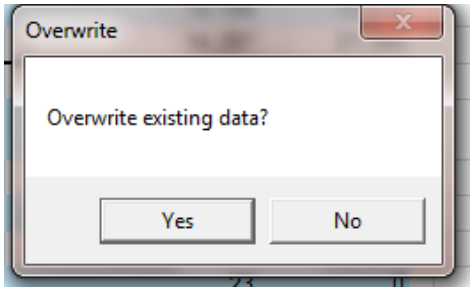
The OK button first checks to makes sure the user has entered a valid sheet name. If the user tries to enter an invalid new sheet name, the new sheet created is deleted. The user will not see the creation or deletion of the new sheet, but rather, the following error message will be displayed.



After choosing which sheet to display the data on, the selected sheet will become the active sheet. The user is then given the following input box to select where on the worksheet the output will be displayed. The default is "A1," but the user can click anywhere on the sheet and the cell address will populate in the input box.



After this, the program will check the chosen output range to make sure the user is not overwriting existing data. If the output would overwrite existing spreadsheet data, the user is given the option to either overwrite the data, or return to the input box to choose a different output location.



The final result is displayed in a table, as shown below.

Ratio	Value
Altman z-score	8.2733
Bankruptcy likelihood	0.00%
A/R Turnover	19.9005
Other Current Assets mix	3.89%
Assets to Equity	1.5397
Asset Turnover	1.1302
Ave collection Period	18.3412
A/P Turnover	4.8153
Beneish Y-score	-2.5361
Prob of Manipulation	0.56%
Book to Market	0.2125
Cash Adequacy	8.8096
Cash Flow to Net Income	1.4478

Return on Equity	=	Return on Sales	x	Asset Turnover	x	Assets-Equity
41.67%		23.95%		1.1302		153.97%

CONCEPTUAL DIFFICULTIES AND LEARNING

Throughout the course of making this project, I encountered many different conceptual difficulties and learning opportunities. One of the first issues I came across was the persistence of public variables. All of my inputs and ratio calculations are stored as public variables so that multiple forms can have access to the data. I soon found out, however, that I was getting erroneous data after running the application a few times. I then remembered that public variables do not automatically reset themselves at the end of a sub procedure or at the end of my project. My first solution to this was to create a sub procedure that set each individual variable back to zero or a zero-length string. After searching on Google, I encountered the "End" statement, which ends all code abruptly, effectively unloading all user forms and resetting all public variables. Now, at any point where the user could exit the application, the "End" statement is used.

The next challenge was with the Inputs form. I wanted the user to be able to use the open workbook to perform calculations or search for data inputs. However, under normal conditions a user form does not allow the user to click anywhere except the user form. I then discovered that if the user form property "ShowModal" is set to false, the user is allowed to use the spreadsheet and then return to the user form. However, this created problems with the way my code was written to run the program. I had initially written something like frmRatioOptions.show followed by frmInputs.show. Since these were both "modeless" forms however, both forms would be displayed rather than seeing them one at a time. The solution was to move the frmInputs.show command to the "OK" button of the first form. The sub procedure used to run the application now only contains one line of code, "frmRatioOptions.show".

Another conceptual difficulty resulted from the fact that user form text box values are by default stored as text, even if they are numerical inputs. My calculations were thus off, because 2 + 2 = 22 in this case instead of 4. I realized that I could use a line of code like Val(frminputs.txtCash) to turn this into a numerical value. However, this ultimately made my "Calculations" sub procedure difficult to read and edit. So, I ended up storing all of the textbox values in currency variables to change the data type and facilitate an easier to edit Calculations sub procedure.

Though the Autofill feature successfully fills in many of the inputs as I planned for, it still is unable to consistently find certain inputs, especially the cash flow inputs. This is mainly due to the wide variation in the way different companies present their financial statement data. The custom "Location" function is able to find a lot of the inputs that Excel's find function was not able to find accurately, but I am still working on improving the Auto fill functionality.

Another learning experience for me was trying to allow the user to select a cell location for the output. I realized quickly that the standard input box method only allowed a user to enter information in the given box. It does not allow a user to select a cell and then have that cell address populate in the text box. I learned however, that there are two versions of the input box in VBA. The "Application.inputbox" version allows you to select which type of input box, and type 8 is "cell reference." The final code to allow the user to select a cell was as follows.

```
Set myRange = Application.InputBox("Please Select an Output Range", Default:="A1", Type:=8)
```

I later learned that the "RefEdit" user form tool would perform the same function, but I decided to leave the "application.inputbox" as part of the code.

APPENDIX- Code for custom Location function

```
Function Location(SearchTerm1 As String, Optional Exclusion1 As String, Optional Exclusion2 As String,
Optional SearchTerm2 As String, Optional SearchTerm3 As String, Optional ExactMatch As Boolean) As
String

Dim row As Long, col As Long

'Find column where financial statement line item labels are located
col = Cells.Find(What:="assets", After:=Range("A1"), LookIn:= _
        xlFormulas, LookAt:=xlPart, SearchOrder:=xlByColumns, SearchDirection:= _
        xlNext, MatchCase:=False, SearchFormat:=False).Column

'If exclusion terms not specified set exclusions to random characters so they will not affect the search
If Exclusion1 = "" Then Exclusion1 = "~|!@"
If Exclusion2 = "" Then Exclusion2 = "~|!@"

'Add wildcards to search terms and exclusions
  If Not ExactMatch Then
     SearchTerm1 = "*" & SearchTerm1 & "*"
     SearchTerm2 = "*" & SearchTerm2 & "*"
     SearchTerm3 = "*" & SearchTerm3 & "*"
   End If

   'If optional arguments are left out, then * will match any item in the search
   If SearchTerm2 = "" Then SearchTerm2 = "*"
   If SearchTerm3 = "" Then SearchTerm3 = "*"

   Exclusion1 = "*" & Exclusion1 & "*"
   Exclusion2 = "*" & Exclusion2 & "*"

For row = 1 To Cells(ActiveSheet.Rows.Count, col).End(xlUp).row
   If LCase(Cells(row, col)) Like LCase(SearchTerm1) And _
   LCase(Cells(row, col)) Like LCase(SearchTerm2) And _
   LCase(Cells(row, col)) Like LCase(SearchTerm3) And _
   Not LCase(Cells(row, col)) Like LCase(Exclusion1) And _
   Not LCase(Cells(row, col)) Like LCase(Exclusion2) And _
   Cells(row, col).End(xlToRight) Like "*#*" Then
      Location = Cells(row, col).Address
      Exit Function
   End If
Next row

End Function
```