2009

An Automation Algorithm for Harvesting Capital Market Information from the Web

Pankaj Agrrawal
University of Maine - Main, pankaj.agrrawal@maine.edu

An automation algorithm for harvesting capital market information from the web

Pankaj Agrrawal
Department of Finance, University of Maine, Orono, Maine, USA

Abstract

Purpose – The purpose of this paper is to develop an algorithm to harvest user specified information on finance portals and compile it into machine-readable datasets for quantitative analysis.

Design/methodology/approach – The Visual Basic macro language in Microsoft Excel is applied to develop code that is not constrained by the single-query function of Excel. The core of the algorithm is built around the splitting of the URL connector line and the placement of a continuously updating variable into which are looped as many tickers as there are in the input list. The output is then written to non-overlapping cells.

Findings – Numerical information placed on major finance websites can be harvested into structured machine-readable datasets by applying this algorithm.

Research limitations/implications – One significant change in Microsoft Excel 2007 is that the worksheet is expanded from 2^{24} to 2^{34} cells, or to be more specific, from 256 (IV) columns × 65,536 rows ($2^8 \times 2^{16}$) to 16,384 (XFD) × 1,048,576 ($2^{14} \times 2^{20}$). These new limits while allowing for a larger number of tickers, still constrain a single worksheet to 16,384 columns. For five fields per ticker that translates into roughly 3,200 ticker symbols.

Practical implications – The algorithm extends user accessibility to websites that do not provide the facility of simultaneous downloading of information on multiple stock tickers. Furthermore, the procedure automates the downloading of multiple pieces of information (fields) and entire tables per ticker (record).

Originality/value – An exhaustive literature search did not find any paper that discusses a multiple ticker algorithm for web harvesting.

Keywords Information retrieval, Worldwide web, Programming and algorithm theory, Capital markets

Paper type Technical paper

The internet has altered production, consumption, transportation, communication, research and a whole range of other aspects of human activity. Physical proximity to the floors of the New York or London Stock Exchanges is no longer necessary for successful portfolio management and financial research. The speed and timeliness with which capital market information is delivered over the internet has effectively removed any logistical advantages that accrue to investors located in the proximity of the marketplace.

Some finance websites openly promote public access while others require premium access, thus making the access to embedded data either costly or inefficient. To the quantitatively inclined student of finance who wants to research a portfolio of securities and requires aggregated data sets, the usefulness of ticker-by-ticker data on a webpage is rather limited. Valuation metrics such as price to earnings, price to sales, price to cash flow, sector exposures and rolled-up risk measures such as aggregate beta or total volatility at the portfolio level, fundamental ratios and betas for exchange

The author wishes to thank an anonymous referee and the editor for suggesting numerous improvements that have benefited the paper. The usual disclaimer applies.

Managerial Finance
Vol. 35 No. 5, 2009
pp. 427-438
© Emerald Group Publishing Limited
0307-4358
DOI 10.1108/03074350910949790

traded funds (Agrrawal and Clark, 2007) or individual securities are necessary for learning about and effectively managing investment portfolios.

The paper provides an algorithm and a procedure that uses readily available resources to harvest web information[1] and to compile it into datasets for financial analysis. Specifically, the single ticker web-query feature in Microsoft Excel 2007 (and earlier versions) is extended to read multiple ticker symbols from an input list, to extract only a targeted subset of web information, and to arrange it in a predictable pattern on a worksheet. The goal is to quickly harvest data from a web page for many firms. This is done using Excel 2007s Visual Basic for Applications (VBA) macro programming language. Apart from set-up time, there is virtually no direct cost involved.

Literature

Financial data harvested from the internet is beginning to be referenced in peer-reviewed journals. A study of intra-day price formation in US equity markets by Hasbrouck (2003), published in the *Journal of Finance*, uses data from the Chicago Mercantile Exchange website to extract the volume-price files [www.cme.com/trading/dta/hist/]. The Chicago Board of Options Exchange (CBOE) website [http://cboe.com/micro/vix/] provides option-implied volatilities for the S&P 100 and NASDAQ 100 indices. Corrado and Miller (2005), utilized this data for their study published in the *Journal of Futures Markets*. The open, high, low and close data for the S&P 500 index obtained from Yahoo! Finance [http://finance.yahoo.com/], was used by Corrado and Miller (2006) for their study on estimating expected excess returns using historical and implied versions of volatility. The study was published in the *Journal of Financial Research*. Sites such as those of the Federal Reserve Bank in St. Louis [http://research.stlouisfed.org/fred2/] or the Federal Reserve Board in Washington, DC. [www.federalreserve.gov/datadownload/] have been widely used for quite some time by economists for downloading macro-economic data. Waggle and Johnson (2004) deployed the data from the Federal Reserve Board to estimate 30-year, conventional mortgage interest rates in their study published in the *Journal of Real Estate Portfolio Management*. Woodroof (1999) shows how to download stock prices into a Microsoft Excel spreadsheet while Rose and Rose (2001) show how to track market activity using Excel spreadsheets. Benninga (2008) discusses the how to create a VBA web-query; however, each run of the macro is limited to collecting data for a single firm. The macro developed here allows each run of the macro to collect data for multiple firms, without the manual entry of successive tickers in a windows dialog box.

The type of data that can be transferred from the web to a local computer using Excel can be broadly classified as one of the following four types:

(1) Data type 1: Web sites that have data in an Excel format. The user simply downloads the Excel file. An example is the web site: http://cboe.com/micro/vix/historical.aspx. The page has a link to an Excel file with historic prices. The user simply downloads the file.

(2) Data type 2: The web page has links that "automatically" transfer the data to an Excel worksheet. Examples are historical prices at Yahoo! Finance and Moneycentral.com. The user selects the firm, frequency of price data (daily, weekly or monthly), and period for the data; the data appears on the screen; there is a link that transfers the data to a local Excel worksheet.

(3) Data type 3: The web page does not have links that "automatically" transfers the data to an Excel worksheet. However, the data is in tables and Excel's Web Query feature can be used to bring in the data.

(4) Data type 4: The web page does not have links that "automatically" transfer the data to an Excel worksheet and it is not in the form of tables. Some data at the St Louis Fed is in this format. The Web Query function can be used but the data, when it appears on an Excel worksheet, is usually not in the desired format. Then, the user needs to use Excel's Text to Columns feature (Data tab, Data Tools group, Text to Columns icon).

The purpose is to describe a way to transfer data of Data type 3. Specifically, to automate the process using different ticker symbols for different firms and thus harvest large amounts of machine-readable data from the web to a local user.

The algorithm described has a generalized framework that applies Visual Basic in Excel to develop a downloading solution that is not constrained by the single-query function of Excel. It extends user accessibility to websites that do not provide the facility of simultaneous downloading of information on multiple stock tickers[2]. Furthermore, the procedure created and described here automates the downloading of multiple pieces of information (fields) and entire tables per ticker (record). This algorithm can be put to use by students and faculty in an academic setting who, for various reasons, do not have professional information delivery platforms that can often be prohibitively expensive, besides being overkill for instructional and learning purposes.

The two-part harvesting process

In this section the single ticker web-query functionality is extended to reading multiple tickers from an input list and thereafter extracting only a subset of the information on a webpage. That information is then transferred and arranged in a predictable pattern on a local Excel sheet, and an automated repetition of the process results in the generation of a machine-readable dataset of large dimensions. Creating the one item "web query" and subsequently automating it to apply the modified query to an input list are the two parts that comprise this procedure. For illustration purposes, we will access a premier provider of processed financial analytics and data, Morningstar.Com, to download fundamental valuation and sector exposure information on ETFs into an Excel spreadsheet.

Setting up a one ticker web query

Start by opening a new Excel 2007 workbook and naming it web-query-ETF-ratios-2008.xlsx. After creating the macro the file name extension needs to be changed to .xlxm, reflecting the fact that a macro is resident within the file. Excel has a feature to transfer numerical as well as non-numerical data from the web[3]. As shown in Figure 1, this option is on the "Data" ribbon, in the "Get External Data" group and is "From Web".

Clicking on "From Web" the button brings up the "New Web Query" dialog box, which is utilized to set up the web-retrieval for a single ticker. The target web address (URL) is entered into Box 1 of the "New Web Query" dialog box. This can be accomplished by opening up a browser, loading the target website, copying and then pasting the URL into the "Address" bar of the "New Web Query" dialog box (see

Figure 2). In this case the target webpage is: http://quicktake.morningstar.com/etf/Portfolio.asp?Symbol=DIA

DIA is an actively traded security on the American Stock Exchange and tracks the Dow Jones Industrial Average; it is commonly referred to as "the Diamonds".

A view and description of the target webpage[4] gives an idea of the composition of the page and how the data tables are embedded and layered on the page. A web page, such as this, has many layers of background graphics, text and also embedded tables that contain numerical data. The embedded tables are indicated by the presence of small arrows next to the tables. Any or all of these tables can be imported from the web page onto an Excel worksheet. Clicking on the arrow box selects the specific table of interest. Once selected, the box changes to a check mark and the table is ready for transfer to an Excel sheet, which is done by clicking on the "Import" button. However, the usefulness of this is limited to data transfer for a single query. In order to repeatedly access the same webpage, but with a different ticker each time, we need to know the table number that is referenced in the webpage. This number is available in something called a query file. The query can be saved by clicking on the small icon to the left of the Options item on the menu bar. By default this file has an .iqy extension. Double clicking on the saved".iqy" file will open a browser program, get the data, open

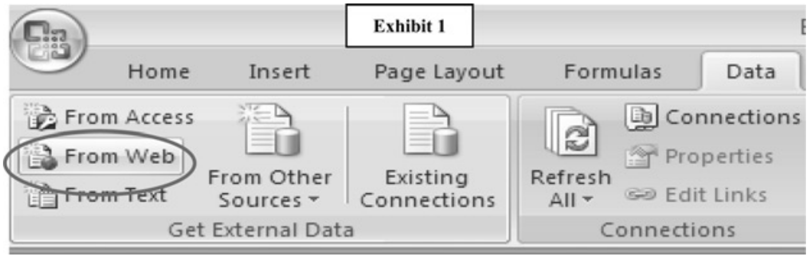
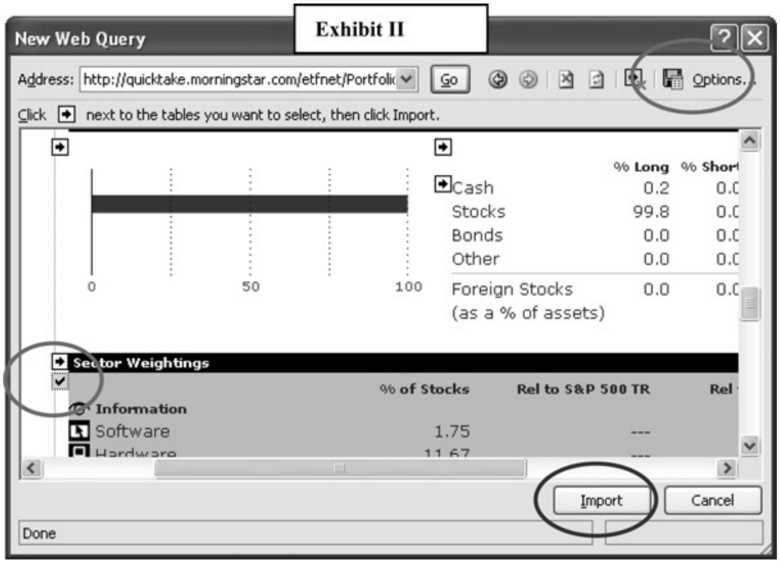


Figure 1.
Data ribbon, get external group, and from web icon



New Web Query

Address: http://quicktake.morningstar.com/etfnet/Portfolio

Click next to the tables you want to select, then click Import.

	% Long	% Short
Cash	0.2	0.0
Stocks	99.8	0.0
Bonds	0.0	0.0
Other	0.0	0.0
Foreign Stocks (as a % of assets)	0.0	0.0

Sector Weightings

	% of Stocks	Rel to S&P 500 TR	Rel
Information			
Software	1.75	---	
Hardware	11.67	---	

Figure 2.
New WebQuery dialog box

Excel, and place the data on an Excel worksheet. Right clicking it and opening it with a text editor such as “Notepad” reveals the code within the query and the table numbers selected (Benninga, 2008). Uncovering these table numbers is important to the automation process, as they are subsequently hard-coded into a VBA routine (see Figure 3).

The table numbers that are on a target webpage can also be directly recorded into the VBA code. This feature will prove to be very useful later on, when some VBA code is written to work around the Excel limitations on its single item web-query feature.

Setting up a multiple ticker web query using VBA code

The steps outlined in Section A above are the building blocks for the automation algorithm that we generalize to process a list of user-determined tickers. This section describes the construction and execution of the VBA code required for the automation. Rose and Rose (2001) illustrate the use of VBA code to automate the trigger of an e-mail when a security in the Excel worksheet reaches a certain price target. There are two parts required to develop the VBA code and to get it to read a list of tickers.

Translation of clicks to command line code. If the various steps outlined in section A above are executed when in the “Record Macro” mode, which is located under the Developer tab, then Excel’s VBA compiler generates the code equivalent to the mouse-clicks[5] (see Figure 4). Upon clicking on the “Visual Basic” icon on the Developer ribbon, the VBA editor window opens. This editor window contains the “module” linked to the spreadsheet in which the code is resident. This VBA module can now be edited to introduce commands that have to be manually hard-coded and cannot be recorded. After the editing is completed, the file is saved in the .xlsm macro enabled format (retain the same filename but with the new extension). A security warning appears each time a file with macros is opened and the suggested security setting is “Disable all macros with notification” for the macros to execute at the discretion of the user. This option is available in the Developer ribbon (see Figure 5).

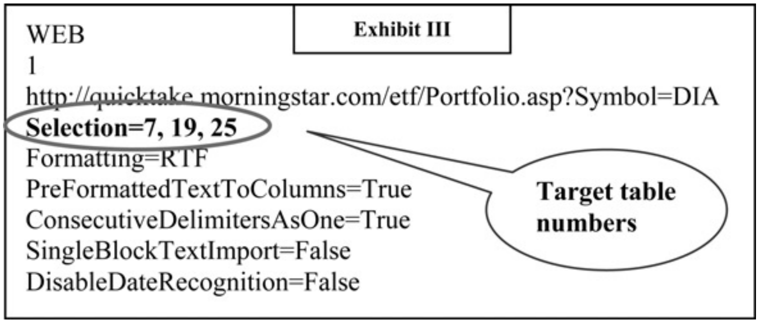


Exhibit III

```
WEB
1
http://quicktake.morningstar.com/etf/Portfolio.asp?Symbol=DIA
Selection=7, 19, 25
Formatting=RTF
PreFormattedTextToColumns=True
ConsecutiveDelimitersAsOne=True
SingleBlockTextImport=False
DisableDateRecognition=False
```

Target table numbers

Figure 3.
Table numbers from query file

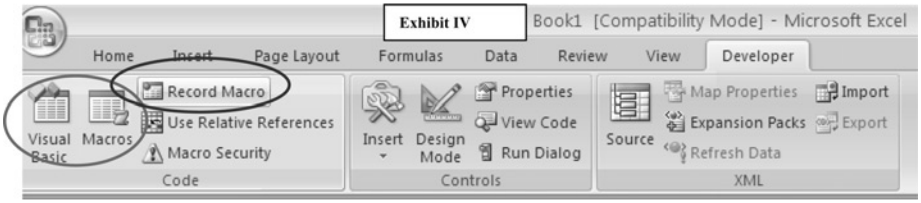


Figure 4.
Macro menu view

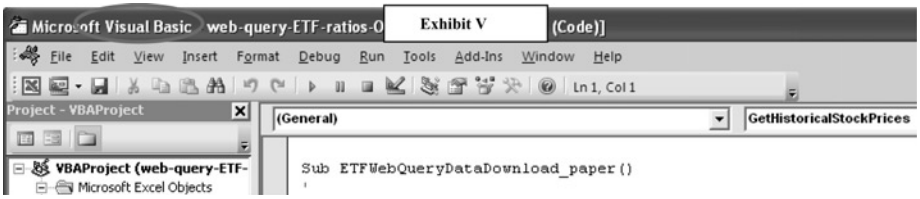
Figure 6 shows the Excel-URL connector line that needs to be split and modified to develop the solution.

A verbal translation of the line of code above is that a Query Table is added to a worksheet called "Sheet1" and then populated with material from a particular webpage; subsequently the information is pasted in cell G5 of "Sheet1". If we could fragment the URL line and introduce a continuously updating variable in place of DIA then we could loop this line through as many tickers as there are in our list. Also, the output location G5 could be dynamic so that the information retrieved for each additional ticker is not over-written. However, to create that workaround, we have to first set up our ticker list on an input sheet and link it with the modified URL connector via a macro, so that the VBA compiler can read the list.

Introducing a variable in the URL. In "Sheet1", the input sheet, copy and paste the portion of the URL code without the specific ticker in the column titled "1/2 URL". The specific tickers of interest are typed into the cells adjacent to the "1/2 URL" column[6], which we can call the "myTicker" column. Utilizing an Excel command to link two text strings (the 1/2 URL and myTicker), the two cells can be combined into one by using the "&" key. As an example, typing in "=B3&C3" in cell A3 (without the quotes) will result in a concatenated URL that can be passed into the VBA routine, and has the ticker appended to it. Now the problem has been simplified. All that needs to be done is to loop around the "Concatenated URL" column rows by cycling through as many cells as we have tickers in the column "myTicker" (see Figure 7).

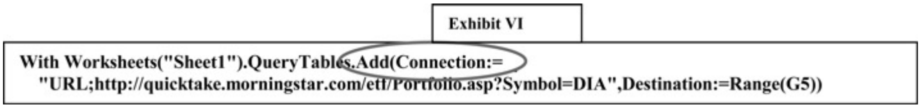
The structure of the program is shown in the following two flow diagrams in Figure 8, each with a different level of detail.

The "For . . . Next" loop is executed in the "inputSheet", which has the full list of tickers for which data is to be retrieved. Until the last element of the list is reached, the loop continues on the core code where the concatenated URL is passed on to the external server for data retrieval. Within the same block of code the retrieved


Microsoft Visual Basic - web-query-ETF-ratios-O
Exhibit V
(Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 1, Col 1
Project - VBAProject
(General)
GetHistoricalStockPrices
VBAProject (web-query-ETF-
Microsoft Excel Objects

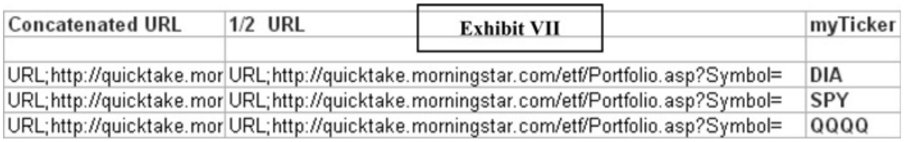
```
Sub ETFWebQueryDataDownload_paper()
'
```

Figure 5.
Excel visual basic editor view



```
With Worksheets("Sheet1").QueryTables.Add(Connection:=
  "URL;http://quicktake.morningstar.com/etf/Portfolio.asp?Symbol=DIA",Destination:=Range(G5))
```

Figure 6.
URL connector line



Concatenated URL	1/2 URL	myTicker
URL;http://quicktake.mor	URL;http://quicktake.morningstar.com/etf/Portfolio.asp?Symbol=	DIA
URL;http://quicktake.mor	URL;http://quicktake.morningstar.com/etf/Portfolio.asp?Symbol=	SPY
URL;http://quicktake.mor	URL;http://quicktake.morningstar.com/etf/Portfolio.asp?Symbol=	QQQQ

Figure 7.
Passing in tickers

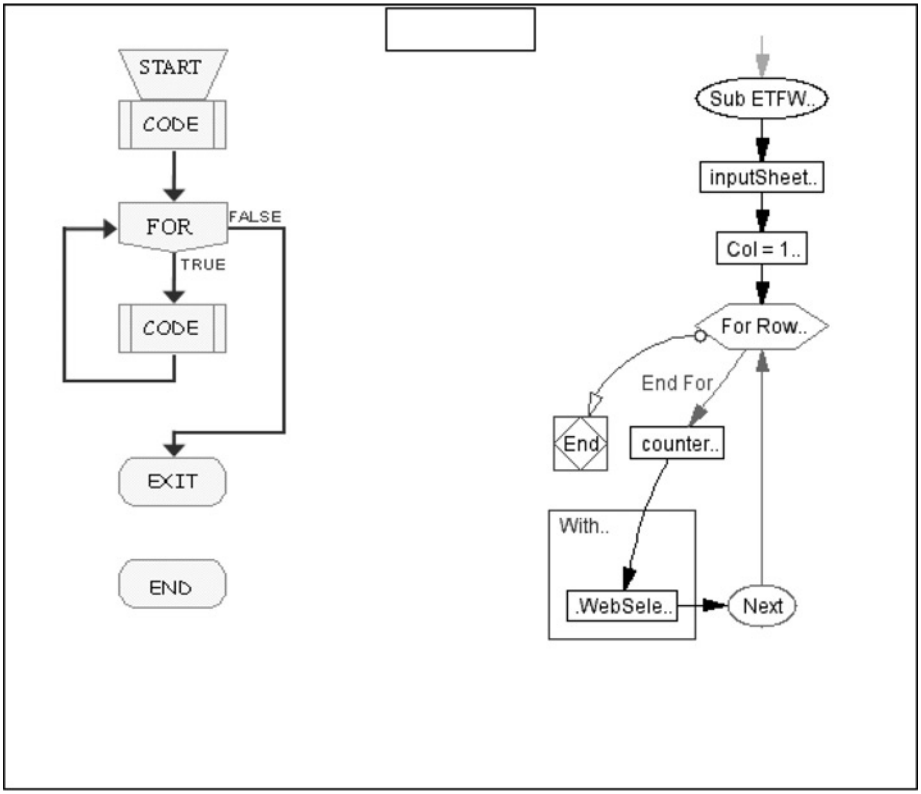
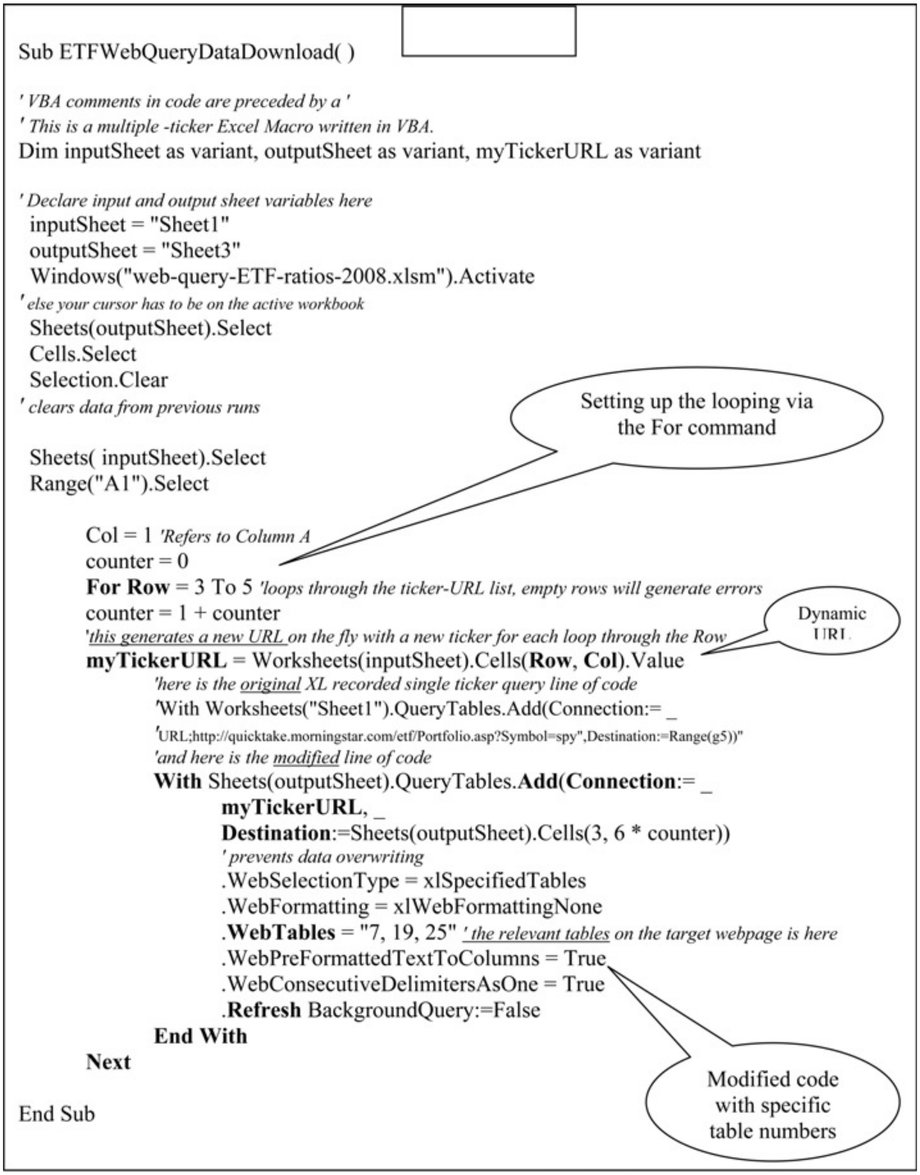


Figure 8.
Flow diagrams of the algorithm

information is stored in non-overlapping cells through the use of a "counter" which is a part of the "Destination" property, as shown in the full program (see Figure 9).

With most of the elements of the program in place, we can now look at the full code to establish how to modify and extend the built-in Excel capability to create an automated algorithm. Explanatory comments are preceded by a single quote in Figure 9, which details all of the necessary code for a multiple-ticker web query. A brief explanation of some of the main VBA properties and methods utilized in the code shown in Figure 9 is listed below:

- The Add method creates the web query and adds it to the specified worksheet.
- The Connection property accepts and passes a URL to the web to complete the query.
- The Destination property determines where the output is to be placed on the output sheet.
- The WebTables property identifies items/tables within a webpage that are to be extracted.
- The Refresh method imports the data onto the worksheet; setting the BackgroundQuery: = False turns off the automatic updating of information (useful for real-time quotes, but not for this scenario).



```
Sub ETFWebQueryDataDownload( )

' VBA comments in code are preceded by a '
' This is a multiple -ticker Excel Macro written in VBA.
Dim inputSheet as variant, outputSheet as variant, myTickerURL as variant

' Declare input and output sheet variables here
  inputSheet = "Sheet1"
  outputSheet = "Sheet3"
  Windows("web-query-ETF-ratios-2008.xlsm").Activate
' else your cursor has to be on the active workbook
  Sheets(outputSheet).Select
  Cells.Select
  Selection.Clear
' clears data from previous runs

  Sheets( inputSheet).Select
  Range("A1").Select

        Col = 1 'Refers to Column A
        counter = 0
        For Row = 3 To 5 'loops through the ticker-URL list, empty rows will generate errors
        counter = 1 + counter
        'this generates a new URL on the fly with a new ticker for each loop through the Row
        myTickerURL = Worksheets(inputSheet).Cells(Row, Col).Value
                'here is the original XL recorded single ticker query line of code
                'With Worksheets("Sheet1").QueryTables.Add(Connection:= _
                'URL;http://quicktake.morningstar.com/etf/Portfolio.asp?Symbol=spy",Destination:=Range(g5))"
                'and here is the modified line of code
                With Sheets(outputSheet).QueryTables.Add(Connection:= _
                        myTickerURL, _
                        Destination:=Sheets(outputSheet).Cells(3, 6 * counter))
                        ' prevents data overwriting
                        .WebSelectionType = xlSpecifiedTables
                        .WebFormatting = xlWebFormattingNone
                        .WebTables = "7, 19, 25" ' the relevant tables on the target webpage is here
                        .WebPreFormattedTextToColumns = True
                        .WebConsecutiveDelimitersAsOne = True
                        .Refresh BackgroundQuery:=False
                End With
        Next

End Sub
```

Setting up the looping via the For command
Dynamic URL
Modified code with specific table numbers

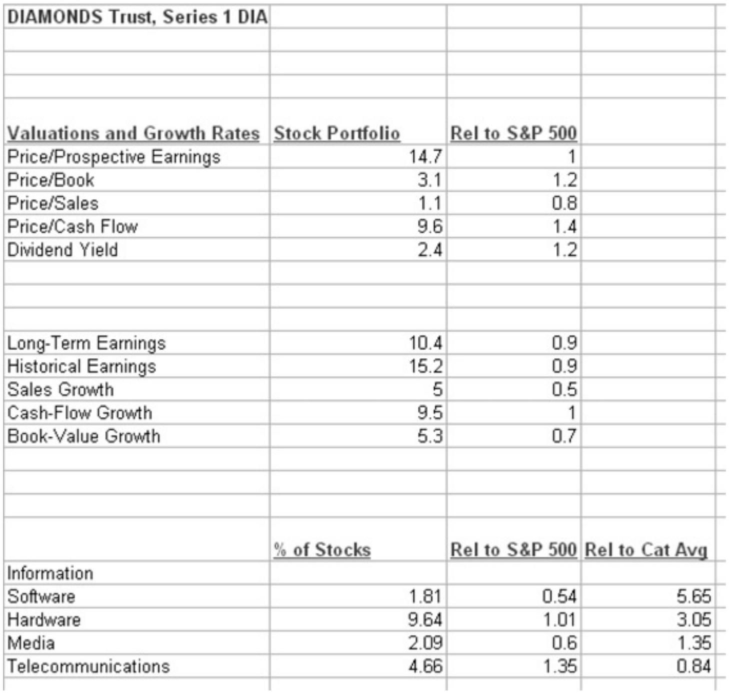
Figure 9.
The full program

As the code loops through each successive row down the same column, a new URL is picked up from the inputSheet, with a new ticker getting appended to the end of the URL as a result of the "&" operation in the inputSheet. The key to this approach is altering Excel's cell reference of say A1 to cells(1, 1) and generalizing it to cells(*i*, *j*) for each row *i* and each column *j*. In the simplified case shown in the VBA code, the column is fixed at 1 and the Row cycles from 3 through 5. The WebQuery operation is performed on cell cell (3, 1), cell (4, 1) and cell (5, 1). These are equivalent to cell references A3, A4 and A5. We pass this new URL to our variable myTickerURL, and with each increase in the "counter" value from 1 through 3 (initial seed set at 0), a QueryTable with new data is added to the outputSheet. The "Destination" of the output is Sheets(outputSheet). Cells (3, 6* counter)). For counter values of 1, 2 and 3 the destination cell values are thus cell (3, 6), cell (3, 12) and cell (3, 18), respectively. This ensures that the output is started at row#3 and skips six columns for each successive ticker. Figures 10 and 11 show how the information tables pertaining to tickers DIA, SPY and QQQQ appear on an Excel worksheet. Figure 10 is a magnified sub-section of the downloaded data, and shows some financial characteristics for the DIA ticker.

Harvested output

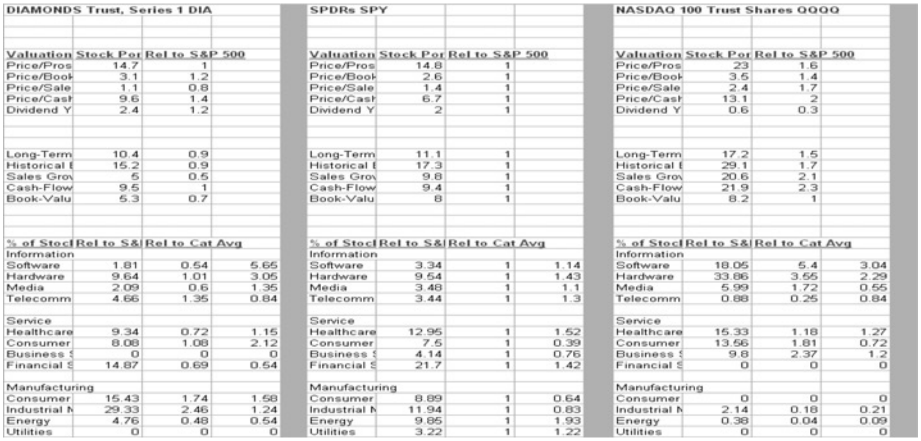
Figures 10 and 11 show how seemingly scattered and haphazardly placed data on a webpage (see Appendix) can be extracted and organized in an orderly manner that makes subsequent analysis easier. A standard web-retrieval function, such as the one built into Excel would populate the cells in Figure 10, for one ticker only, necessitating the use of multiple queries for a list of tickers. With the algorithm described above, the full set of desired items for each ticker is accessed, downloaded and stored in one round, within a few seconds, and with no repeated clicking.

The continuation of the loop produces a sheet, as shown in Figure 11, in which data for the same fields is stored for successive tickers in our list.



DIAMONDS Trust, Series 1 DIA			
Valuations and Growth Rates	Stock Portfolio	Rel to S&P 500	
Price/Prospective Earnings	14.7	1	
Price/Book	3.1	1.2	
Price/Sales	1.1	0.8	
Price/Cash Flow	9.6	1.4	
Dividend Yield	2.4	1.2	
Long-Term Earnings	10.4	0.9	
Historical Earnings	15.2	0.9	
Sales Growth	5	0.5	
Cash-Flow Growth	9.5	1	
Book-Value Growth	5.3	0.7	
	% of Stocks	Rel to S&P 500	Rel to Cat Avg
Information			
Software	1.81	0.54	5.65
Hardware	9.64	1.01	3.05
Media	2.09	0.6	1.35
Telecommunications	4.66	1.35	0.84

Figure 10.
A section of the downloaded data



DIAMONDS Trust, Series 1 DIA				SPDRs SPY				NASDAQ 100 Trust Shares QQQQ			
Valuation	Stock Por	Rel to S&P 500		Valuation	Stock Por	Rel to S&P 500		Valuation	Stock Por	Rel to S&P 500	
Price/Pros	14.7	1		Price/Pros	14.8	1		Price/Pros	23	1.6	
Price/Book	3.1	1.2		Price/Book	2.6	1		Price/Book	3.5	1.4	
Price/Sale	1.1	0.8		Price/Sale	1.4	1		Price/Sale	2.4	1.7	
Price/Cash	9.6	1.4		Price/Cash	6.7	1		Price/Cash	13.1	2	
Dividend Y	2.4	1.2		Dividend Y	2	1		Dividend Y	0.6	0.3	
Long-Term	10.4	0.9		Long-Term	11.1	1		Long-Term	17.2	1.5	
Historical E	15.2	0.9		Historical E	17.3	1		Historical E	29.1	1.7	
Sales Grov	5	0.5		Sales Grov	9.8	1		Sales Grov	20.6	2.1	
Cash-Flow	9.5	1		Cash-Flow	9.4	1		Cash-Flow	21.9	2.3	
Book-Valu	5.3	0.7		Book-Valu	8	1		Book-Valu	8.2	1	
% of Stocl	Rel to S&I	Rel to Cat Avg		% of Stocl	Rel to S&I	Rel to Cat Avg		% of Stocl	Rel to S&I	Rel to Cat Avg	
Information				Information				Information			
Software	1.81	0.54	5.65	Software	3.34	1	1.14	Software	18.05	5.4	3.04
Hardware	9.64	1.01	3.05	Hardware	9.54	1	1.43	Hardware	33.86	3.55	2.29
Media	2.09	0.6	1.35	Media	3.48	1	1.1	Media	5.99	1.72	0.55
Telecomm	4.66	1.35	0.84	Telecomm	3.44	1	1.3	Telecomm	0.88	0.25	0.84
Service				Service				Service			
Healthcare	9.34	0.72	1.15	Healthcare	12.95	1	1.52	Healthcare	15.33	1.18	1.27
Consumer	8.08	1.08	2.12	Consumer	7.5	1	0.39	Consumer	13.56	1.81	0.72
Business S	0	0	0	Business S	4.14	1	0.76	Business S	9.8	2.37	1.2
Financial S	14.87	0.69	0.54	Financial S	21.7	1	1.42	Financial S	0	0	0
Manufacturing				Manufacturing				Manufacturing			
Consumer	15.43	1.74	1.58	Consumer	8.89	1	0.64	Consumer	0	0	0
Industrial N	29.33	2.46	1.24	Industrial N	11.94	1	0.83	Industrial N	2.14	0.18	0.21
Energy	4.76	0.48	0.54	Energy	9.85	1	1.93	Energy	0.38	0.04	0.09
Utilities	0	0	0	Utilities	3.22	1	1.22	Utilities	0	0	0

Figure 11.
Downloaded data for multiple tickers

Conclusion

The design, construction and output of an automation device for large-scale harvesting of financial information resident on public access web-servers is discussed. The paper extends the single item web-query functionality in Excel, to reading multiple tickers from an input list and extracting only a targeted subset of information on a web page, with the final objective of compiling a structured machine-readable dataset. The algorithm is not limited in its applicability to any particular website and with minor modifications can be deployed to harvest information from any public web page. The full program and the algorithm are shown in the paper and no additional component, software or subscription is required. Such an algorithm can be devised with readily available tools on most personal computers, and at no direct cost to the student or researcher who requires the usage of multi-security datasets for financial study and research[7].

Notes

1. Numerical financial information fields that are embedded on a target webpage; see the Appendix for an illustration.
2. Note that all information being harvested is made available by the websites for public viewing and usage. This algorithm only accesses information that resides in the public domain, in a systematic and fast process, and is not a "hack" of any type.
3. If the desired web page is opened in Microsoft's Internet Explorer one can place the pointer on the data, right click, then from the pop up menu choose Export to Microsoft Excel. If the data is in table form it is transferred directly to Excel (starting in cell A1 of the first worksheet of a new workbook). If the data is not in table form the command opens the New Web Query dialog box (however, this method does not work with the Firefox browser).
4. See the Appendix for an illustration.
5. Excel 2007s default list of tabs does not include Developer. To display it: click the Microsoft Office Button ●, and then click Excel Options. In the Popular category, under Top options for working with Excel, select the Show Developer tab in the Ribbon check box, and then click OK (from MS Excel 2007 Help).

6. This three-ticker illustration can easily be extended to a larger list of tickers. For a fully compiled file with a list of some of the most active ETF's please send an e-mail to the author.
7. For a fully compiled Excel file that illustrates this algorithm, please e-mail the author at pankaj.agrrawal@maine.edu

References

Agrrawal, P. and Clark, J.M. (2007), "ETF betas: a study of their estimation sensitivity to varying time intervals", *Institutional Investor Journals*, Annual ETF Issue, Fall 2007, pp. 96-103.

Benninga, S. (2008), *Financial Modeling*, The MIT Press, Cambridge, MA.

Corrado, C. and Miller, T. (2005), "The forecast quality of CBOE volatility indexes", *Journal of Futures Markets*, Vol. 25 No. 4, pp. 339-73.

Corrado, C.J. and Miller, T.W. (2006), "Estimating expected excess returns using historical and option-implied volatility", *Journal of Financial Research*, Vol. 29 No. 1, pp. 95-112.

Hasbrouck, J. (2003), "Intraday price formation in US equity index markets", *Journal of Finance*, Vol. 58 No. 6, pp. 2375-99.

Rose, A.M. and Rose, J.M. (2001), "The automated spreadsheet", *Journal of Accountancy*, Vol. 191 No. 4, pp. 33-41.

Waggle, D. and Johnson, D.T. (2004), "Home ownership and the decision to invest in REITs", *Journal of Real Estate Portfolio Management*, Vol. 10 No. 2, pp. 129-44.

Woodroof, J. (1999), "How to link to web data", *Journal of Accountancy*, Vol. 187 No. 3, pp. 55-8.

Further reading

Davis, C.E., Clements, C. and Keuer, W.P. (2003), "Web-based reporting: a vision for the future", *Strategic Finance*, Vol. 85 No. 3, pp. 44-9.

Kim, S.H. (2000), "An architecture for advanced services in cyberspace through data mining: a framework with case studies in finance and engineering", *Journal of Organizational Computing & Electronic Commerce*, Vol. 10 No. 4, pp. 257-70.

Phelan, S. (2001), "The internet as an investment tool", *Journal of Accountancy*, Vol. 192 No. 2, pp. 27-31.

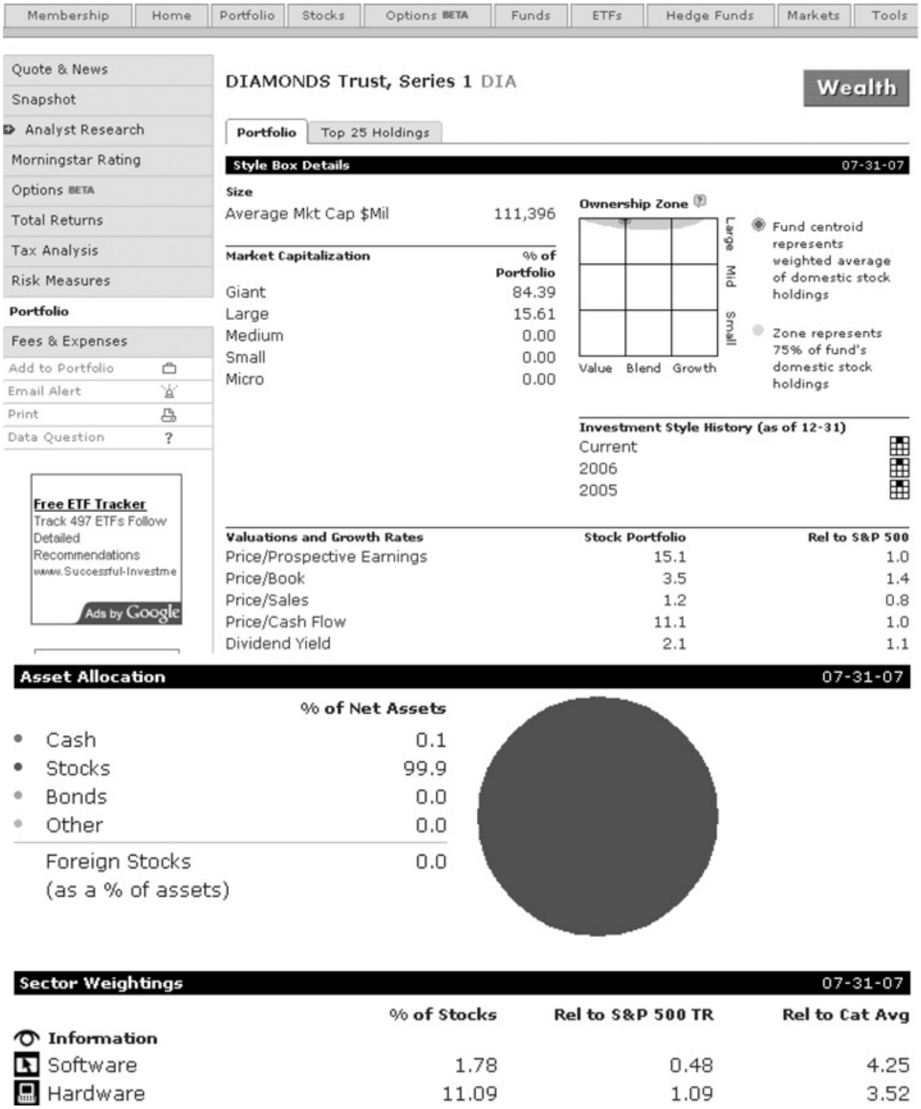
Thau, A. (2005), "Muni trading gets real: current prices on the internet", *AAII Journal*, Vol. 27 No. 4, pp. 5-10.

Vakil, F. and Lu, F.L. (2005), "The effect of the internet on stock market volume and volatility", *Review of Business*, Vol. 26 No. 3, pp. 26-30.

Zarowin, S. (2003), "A Napster for financial data", *Journal of Accountancy*, Vol. 195 No. 1, pp. 66-70.

Appendix

Figure A1 is a screen capture of a webpage, on which the information that we are interested in compiling into a structured dataset is resident. A page, such as this, has many layers of graphics and embedded tables that contain text and numerical data. The procedure described in the paper targets only specific tables and extracts them onto a spreadsheet leaving behind various images, text and unwanted tables. In that sense the algorithm "harvests" information from the web.



DIAMONDS Trust, Series 1 DIA

Style Box Details 07-31-07

Size

Average Mkt Cap $Mil	111,396

Market Capitalization	% of Portfolio
Giant	84.39
Large	15.61
Medium	0.00
Small	0.00
Micro	0.00

● Fund centroid represents weighted average of domestic stock holdings

● Zone represents 75% of fund's domestic stock holdings

Investment Style History (as of 12-31)

Current
2006
2005

Valuations and Growth Rates	Stock Portfolio	Rel to S&P 500
Price/Prospective Earnings	15.1	1.0
Price/Book	3.5	1.4
Price/Sales	1.2	0.8
Price/Cash Flow	11.1	1.0
Dividend Yield	2.1	1.1

Asset Allocation 07-31-07

	% of Net Assets
● Cash	0.1
● Stocks	99.9
● Bonds	0.0
● Other	0.0
Foreign Stocks (as a % of assets)	0.0

Sector Weightings 07-31-07

	% of Stocks	Rel to S&P 500 TR	Rel to Cat Avg
Information			
Software	1.78	0.48	4.25
Hardware	11.09	1.09	3.52

Figure A1.
Screen capture of webpage

Corresponding author
Pankaj Agrrawal can be contacted at: pankaj.agrrawal@maine.edu