Fairfield UNIVERSITY

International Journal of Computer and Systems Engineering

Volume 1 | Issue 1 Article 2

July 2020

The Vanguard Fund Report

Doug Lyon
Fairfield University, lyon@docjava.com

International Journal of Computer and Systems Engineering
Online at https://digitalcommons.fairfield.edu/ijcase/ Published by Fairfield University, **DigitalCommons@Fairfield © 2020**
Vol. 1, No. 1, August 2020

The Vanguard Fund Report

Deriving the PEGY ratio from the Vanguard Fund metrics with a Web Data Miner

by Douglas Lyon and Predrag Bokšić

Abstract

This paper presents a framework for data mining the Vanguard family of funds in order to extract a cap-weighted PEGY ratio. *Vanguard* offers 128 mutual funds and 59 exchange-traded funds. Our framework enables the construction of a wide variety of screeners that can potentially be used on a variety of different funds.

1. Problem Statement

Given several Vanguard funds, we seek to find the PEGY of each fund and report a confidence in our finding. We are subject to the constraint that the fund include elements with unknown PEGY ratios. The PEGY ratio relies on the **P/E ratio** - the financial metric used for the evaluation of a stock valuation. We obtain the P/E ratio by dividing the **stock price P** with the **earnings-per-share** (E or EPS).

To find the PEGY ratio, we use:

$$PEGY = \frac{\frac{Price}{EPS}}{\left[EPS_{growth} + Yield\right]}$$

Thus, we compute the PEG ratio by dividing the P/E ratio by the ***expected*** **earnings-per-share + the growth rate for the next** year, a projected value denoted as *EPS_growth* expressed as a percentage [Lynch]. The dividend yield is defined as the cash paid per share over the prior year divided by the current stock price times 100 (to obtain a percentage). The Vanguard web pages that display fund data use JavaScript to help generate dynamic content. Thus, we must use a JavaScript-enabled browser to render and save the pages.

2. Saving a Webpage Generated via JavaScript

Java SE 8 controls the overall execution, parses the downloaded pages and generates an HTML page that contains a table. Java 8 compatible libraries are used in this project. The libraries used include:

```
commons-codec:commons-codec:1.11

net.sourceforge.htmlunit:htmlunit:2.33 – HTML data
acquision library, org.freemarker:freemarker:2.3.28 – HTML
publisher library and org.jsoup:jsoup:1.11.3 – HTML data
parser library
```

In addition, the resulting HTML page uses *JQuery* and *Tablesorter* libraries (written in JavaScript) to format the data table. Tools we used include Kubuntu 18.04.2, *xdotool*, *wmctrl*, Python 2, and Firefox. These dependencies can be satisfied with the following unit command:

```
sudo apt install xdotool wmctrl python firefox openjdk-11-
jdk python-dev
```

Firefox Settings

Next, we need to prepare the Firefox browser for data mining. We start the Firefox browser for the first time and clear any welcome or configuration message. The browser can start with an empty tab each time you load it, without restoring the session, so select this option first in the “Options” page. Disable the use of history and cache. Enable the option to delete the cookies and site data when Firefox is closed. Open a new tab and visit this address: *about:config.* To allow a clean browser startup use:

```
browser.sessionstore.resume_from_crash=false

toolkit.startup.max_resumed_crashes=-1
```

Running the Program

Delete the old pages downloaded in the previous round of web scraping:

```
rm -rf old
rm -rf etf/*
rm -rf mutualfund/*
rm -rf [0-9]*
```

Run the program manually using:

```
<\java -jar vanguard.jar\
```

Or run it automatically using:

```
<\java -jar vanguard.jar auto\
```

This script enables autonomous execution. The script ends with a system “shutdown 30” command.

```
<\./run.sh\
```

Program Concept

The program downloads 59 exchange-traded funds and 128 mutual funds. The page download task requires a full-fledged browser, like Firefox to load, render, and save the pages. A Bash script automates the "Save Page As" operation. The delicate portions of the script are the time constants that impact the web page loading time. The script inputs a time variable in seconds, representing the time to open and render the page:

```
load_wait_time=20
```

The save-as dialog box requires time to remain displayed. We use:

```
sleep 12
```

To extend the download time use:

```
save_wait_time=8
```

The mining operation takes 2 hours to complete depending on the network conditions and anti-robot technology. The save-as dialog is modal so that the thread is blocked until the save completes. New tabs pile up during the save-as form processing. Automation relies on GUI automation from *xdotool* [Man]. *Xdotool* opens and closes Firefox windows and fills out the save-as dialogue. For example, it sends the Ctrl+F4 to the given window to close it. The automation script uses a different command to close the last remaining Firefox window(s) using: wmctrl -c "Mozilla Firefox". After the fund lists have downloaded, the pages are loaded into a Java program for processing by the *WebClient* class using the *HTMLUnit* library:

```
static WebClient webClient = new
WebClient(BrowserVersion.FIREFOX_60);

webClient.getOptions().setJavaScriptEnabled(false);

WebRequest request = new WebRequest(new URL(url));
HtmlPage page = webClient.getPage(request);
```

The URL for the local file can be produced from a string named “url”, which is retrieved from the local File object that represents the downloaded page:

```
url = file.toURI().toURL().toString().
```

In the following code snippet, we show a range of URLs grabbed from the page along with some extra data. These URLs represent the links to the individual fund pages.

```
List<DomElement> links =
page.getByXPath("//div[contains(@class,'productNameDataCell
productName fixedCol')]/a");

for (DomElement link : links) {
String href = link.getAttribute("href");
HtmlTableDataCell parentCell = (HtmlTableDataCell)
link.getByXPath("../../../..").get(0);
HtmlTableDataCell nextSibling = (HtmlTableDataCell)
parentCell.getByXPath("following-sibling::td").get(0);
HtmlTableDataCell secYieldCell = (HtmlTableDataCell)
parentCell.getByXPath("following-
sibling::td[contains(@class,'secYield')]").get(0);

String ticker = nextSibling.asText();
String yield =
CommonUtils.getFloatAsString(secYieldCell.asText());
String id = getIdFromHref(href);
String name = link.asText();
String individualFundUrl = PROFILE_URL.replace("@", id);

...
```

HtmlUnit offers the method *getByXPath* used to extract a number from a table embedded in the web page:

```
page.getByXPath("//div[contains(@class,'productNameDataCell
productName fixedCol')]/a");
```

Extracts the “div” element or elements in the web page’s HTML code that contains a class “productNameDataCell productName fixedCol”. The HTML code sample follows:

```
<div class="productNameDataCell productName fixedCol">
    <a
href="https://personal.vanguard.com/us/funds/snapshot?FundI
ntExt=INT&amp;FundId=0927">
        <span>Long-Term Bond ETF</span>
    </a>
</div>
```

The program downloads all the fund pages, for off-line processing. The Java program generates the *links.sh* script, sets its run permissions (/bin/chmod +x links.sh), and runs the script. Three Java classes

are used to control the execution process, namely the *ExecutorService*, *ProcessBuilder*, and *Process*. The *links.sh* script also allows us to see the explicit page download requests. For example:

```
#!/usr/bin/env bash
mkdir -p 5023
./save_page_as -b firefox --load-wait-time 20 -d 5023
https://advisors.vanguard.com/web/c1/fas-
investmentproducts/5023/portfolio
```

Two fund portfolio pages can be examined at https://investor.vanguard.com/mutual-funds/profile/vmmsx and https://investor.vanguard.com/etf/profile/portfolio/vglt [Investor]. A user looking at the fund page, should click on the Portfolio & Management tab to see the Price/earnings ratio and Earnings growth rate at https://investor.vanguard.com/mutual-funds/profile/portfolio/vmmsx [Investor]. Growth and P/E ratio are extracted by using a String key that locates these numbers in the table of data. The HTML is downloaded sans JavaScript:

```
// find growth
String growth = "";
String key = "<span data-ng-bind-
html=\"stockTable.nav.earningsGrowthRt | percentage:1 |
vuiEmDash\" class=\"ng-binding\">";
if (pageSource != null && !pageSource.isEmpty() &&
pageSource.contains(key)) {
    int begin_index = pageSource.indexOf(key) +
key.length();
    growth = (String) pageSource.subSequence(begin_index,
pageSource.length());
    if (growth.charAt(0) == '—' && growth.charAt(1) == '<')
{
        growth = "";
        System.out.println("  Failed to get growth.");
    } else {
        int end_index = growth.indexOf("%");
        growth = (String) growth.subSequence(0, end_index);
        System.out.println("  growth: " + growth);
    }
} else {
    System.out.println("  Failed to get growth.");
}

// find pe_ratio
String pe_ratio = "";
key = "<span data-ng-bind-
html=\"stockTable.nav.priceEarningsRatio | ratio |
vuiEmDash\" class=\"ng-binding\">";
if (pageSource != null && !pageSource.isEmpty() &&
```

```
pageSource.contains(key)) {
    int begin_index = pageSource.indexOf(key) +
key.length();
    pe_ratio = (String) pageSource.subSequence(begin_index,
pageSource.length());
    int end_index = pe_ratio.indexOf("x");
    pe_ratio = (String) pe_ratio.subSequence(0, end_index);
    System.out.println("  pe_ratio: " + pe_ratio);
} else {
    System.out.println("  Failed to get p/e ratio.");
}
```

In the case of the *VMMSX* fund, the SEC yield is given with "-", which implies it is zero. Price/earnings ratio is 12.5x. Earnings growth rate is 10.3%. From these values, we derive the PEGY ratio of 1.214.

A table of data may contain a number, a character, or a missing character further complicating the processing. After processing is completed, Freemarker generates the HTML.

3. The Fund PEGY Table

This section shows a portion of the *funds* dataset:

PEGY	P/E	EPS Growth	Yield	Beta (3y)	Ticker	Name	401k qualified
0.709	10	14.1	0	0.86	VNQI	Global ex-U.S. Real Estate ETF	No
0.709	10	14.1	0	0.87	VGRLX	Global ex-U.S. Real Estate Index Admiral Shares	No
0.802	21.1	25.3	1	0.84	VOX	Communication Services ETF	No
0.9	21.6	24	0	1.2	VWIGX	International Growth	No
0.915	14	13.3	2	1.04	VWNDX	Windsor	No
0.941	24.5	24.8	1.24	1.07	MGK	Mega Cap Growth ETF	No
0.985	13.4	13.6	0	1	VEMAX	Emerging Markets Stock Index Admiral Shares	No
0.985	13.4	13.6	0	1	VWO	FTSE Emerging Markets ETF	No
0.988	17	15.9	1.31	1.13	VPMAX	PRIMECAP Admiral Shares	No
1.006	18.2	17.3	0.79	1.24	VHCAX	Capital Opportunity Admiral Shares	No
1.05	12.7	12.1	0	0.91	VPL	FTSE Pacific ETF	No
1.05	12.7	12.1	0	0.91	VPADX	Pacific Stock Index Admiral Shares	No
1.085	16.6	15.3	0	1.17	VINEX	International Explorer	No
1.088	14.4	10.9	2.33	1.11	VCVLX	Capital Value	No
1.093	15.7	12.9	1.47	1.13	VEVFX	Explorer Value	No
1.148	14	12.2	0	1.03	VFSAX	FTSE All-World ex-US Small-Cap Index Admiral Shares	No
1.148	14	12.2	0	1.03	VSS	FTSE All-World ex-US Small-Cap ETF	No
1.158	15.9	12.3	1.43	1.24	VSEQX	Strategic Equity	No
1.191	13.2	8.9	2.18	1.06	VFH	Financials ETF	No
1.202	14.3	11.9	0	N/A	VSGX	ESG International Stock ETF	No
1.203	30.7	25.1	0.41	1.12	VWUSX	U.S. Growth	No
1.214	12.5	10.3	0	1.06	VMMSX	Emerging Markets Select Stock	No
1.28	17.2	12.1	1.34	1.13	VPCCX	PRIMECAP Core	No
1.294	14.1	10.9	0	0.99	VXUS	Total International Stock ETF	No
1.294	14.1	10.9	0	0.99	VTIAX	Total International Stock Index Admiral Shares	Yes
1.324	22.9	16.1	1.2	1.19	VCR	Consumer Discretionary ETF	No
1.336	14.5	9	1.85	1.14	VASVX	Selected Value	No
1.339	27.1	19.5	0.74	1.26	VBK	Small-Cap Growth ETF	No
1.339	27.1	19.5	0.74	1.26	VSGAX	Small-Cap Growth Index Admiral Shares	No
1.343	14.1	10.5	0	0.98	VFWAX	FTSE All-World ex-US Index Admiral Shares	No
1.343	14.1	10.5	0	0.98	VEU	FTSE All-World ex-US ETF	No
1.357	25.3	17.5	1.14	1.08	VUG	Growth ETF	No
1.358	25.3	17.5	1.13	1.08	VIGAX	Growth Index Admiral Shares	No
1.374	19.3	12.4	1.65	1	VFTAX	FTSE Social Index Admiral Shares	No
1.383	18.5	11.8	1.58	1.29	VIS	Industrials ETF	No
1.433	18.5	11.4	1.51	1.2	VSMAX	Small-Cap Index Admiral Shares	No
1.433	18.5	11.4	1.51	1.2	VB	Small-Cap ETF	No
1.435	20.1	12.4	1.61	N/A	ESGV	ESG U.S. Stock ETF	No
1.444	14.3	9.9	0	0.98	VTMGX	Developed Markets Index Admiral Shares	No
1.444	14.3	9.9	0	0.98	VEA	FTSE Developed Markets ETF	No
1.445	13.7	8.3	1.18	1.29	VSTCX	Strategic Small-Cap Equity	No
1.48	22.5	14.9	0.3	1.24	VEXPX	Explorer	No
1.511	18.8	11.1	1.34	1.18	VXF	Extended Market ETF	No
1.511	18.8	11.1	1.34	1.18	VEXAX	Extended Market Index Admiral Shares	Yes
1.56	19.7	10.4	2.23	1.05	VBIAX	Balanced Index Admiral Shares	No
1.575	16.7	10.6	0	0.87	VTWAX	Total World Stock Index Admiral Shares	No
1.575	16.7	10.6	0	0.87	VT	Total World Stock ETF	No
1.595	20	10.9	1.64	1.01	VTCLX	Tax-Managed Capital Appreciation Admiral Shares	No
1.611	19.9	10.4	1.95	0.84	VTMFX	Tax-Managed Balanced Admiral Shares	No
1.616	19.5	10.5	1.57	1.01	VQNPX	Growth and Income	No
1.618	21.7	12.2	1.21	1.1	VGT	Information Technology ETF	No

4. Querying Zacks.com

The second approach to Vanguard data mining uses *FinViz* and *Zacks* focusing on companies and funds, respectively. The data miner queries the *FinViz* website first, in order to obtain P/E ratio, EPS growth for the next 5 years and dividend yield. We download this data via special URLs, for example: https://finvinz.com/screener.ashx?v=150&r=1&c=0,1,2,7,9,14,20,48,65. The URL is changed to switch the range of companies in focus, by altering the "r=1" portion of the URL: https://finviz.com/screener.ashx?v=150&r=1&c=0,1,2,7,9,14,20,48,65

FinViz reveals a total of 7586 listed companies listed in the stock markets. The page does not require JavaScript execution. This works well on most funds, but there are exceptions, for example: "VOX", fund Vanguard Communication Services has a URL:

https://www.zacks.com/funds/mutual-fund/quote/VOX

Which redirects to:

https://www.zacks.com/funds/etf/VOX/profile

The VOX fund is an ETF fund that does not belong to the mutual fund pages group, hence the redirection. On the other hand, a mutual fund page could be read from:

https://www.zacks.com/funds/mutual-fund/quote/VWEHX

A fund profile page offers some fund data, such as the fund "SEC yield". Additionally, the "portfolio holdings" page lists the company stocks in the funds' portfolio. For example:

https://www.zacks.com/funds/etf/VIS/holding.

However, the same is not applicable to all funds. In some cases, we see a list that includes bonds, or foreign market stocks, while some are listed as "000OTHR":

https://www.zacks.com/funds/mutual-fund/quote/VWEHX/holding

The elements of the fund profile page can be processed with a cssQuery selector that extracts an HTML table element. Alternatively, the *Yahoo Finance* website could serve us with similar data if the web queries (once a day) in accordance with the fair use policy. For example:

```
https://finance.yahoo.com/quote/ + ticker + ?p= + ticker
```

The following code fragment extracts the 3-year beta risk factor from a data table at Yahoo Finance:

```
// parse HTML with JSoup
Document doc = null;
try {
    doc = Jsoup.parse(url, 30000);
} catch (Exception e) {
    e.printStackTrace();
```

```
}
// parse the table
// tutorial:
https://jsoup.org/apidocs/org/jsoup/select/Selector.html
Elements table = null;
if (doc != null) {
    table = doc.select("table[class=\"W(100%) M(0)
Bdcl(c)\"]");
}
Elements rows = null;
if (table != null) {
    rows = table.select("tr");
}
// table data storage
Float beta = Float.NaN;
String beta_string = "";

if (rows != null) {
    for (Element e : rows) {
        beta_string = e.select("td[data-test=\"BETA_3Y-
value\"]").text();

…
```

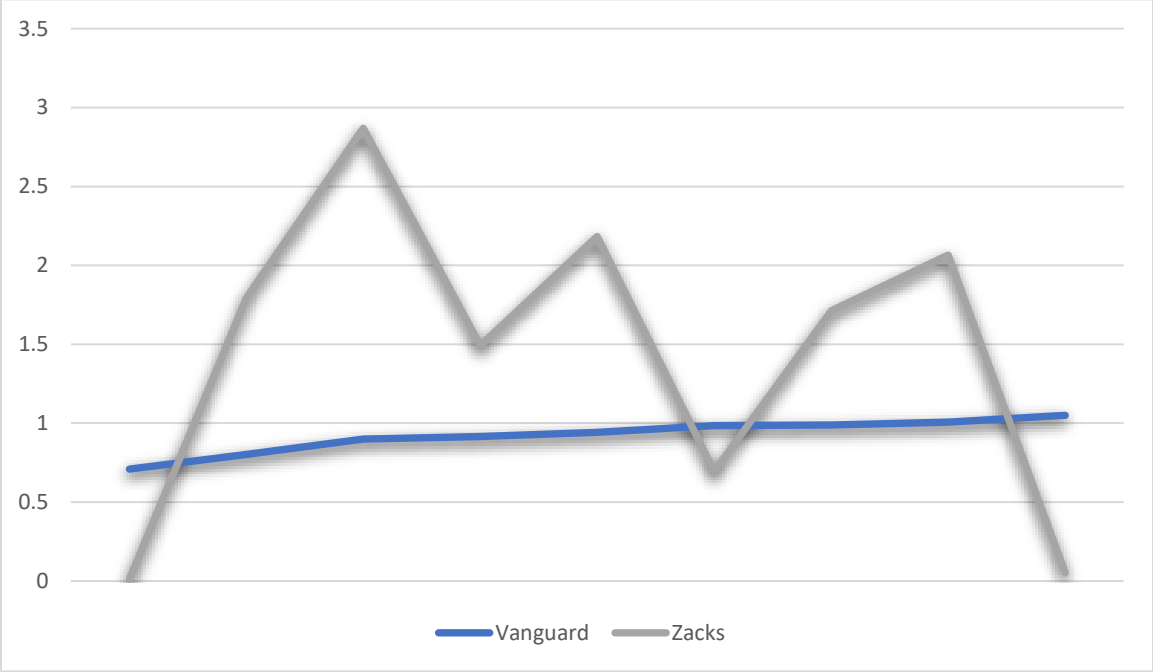
However, the holdings list page at *Zacks.com* appears in a long text string (abbreviated below). Notice the resemblance between the JSON data format and the string:

```
document.table_data   =  [  [ "<a
href=\"\/\/www.zacks.com\/funds\/mutual-fund\/quote\/ABBV\"
rel=\"ABBV\" class=\" hoverquote-container-od \"
alt=\"ABBV  Quote\" title=\"ABBV  Quote\" show-add-
portfolio=\"true\" ><span class=\"hoverquote-
symbol\">ABBV<\/span><\/a>", "2/28/2018", "4,670,093",
"540,936,863.00", "6.05 %", "ABBVIE INC", "COMMON" ] ,…  ,
[ "", "2/28/2018", "5,634,091", "11,155,500.00", "0.12 %",
"TRANSLATE BIO", "CVT PREF" ] ,  [ "", "2/28/2018",
"655,700", "17,697,343.00", "0.20 %", "ODONATE THERAPEUTICS
LLC", "COMMON" ] ] ;
```

If the fund portfolio data is available, we use the PEGY ratios of company stocks that are included in the fund portfolio in order to form a weighted sum of PEGY ratios. Each individual PEGY value is multiplied by the stock's percentage weight (denoted in the table), which represents a stock's portion of the fund net assets. In many cases, this list of companies is available in the holdings list page.

Vanguard PEGY		Zacks PEGY		Ticker
0.709		0.011		VGRLX
0.802		1.793		VOX
0.9		2.869		VWIGX
0.915		1.491		VWNDX
0.941		2.182		MGK
0.985		0.695		VEMAX
0.988		1.712		VPMAX
1.006		2.065		VHCAX
1.05		0.051		VPADX

The PEGY values derived from Zacks and FinViz mining vary significantly when compared with the fund data obtained from Vanguard directly, a variability discussed in [Lyon]. The highest company stock data coverage is the VDADX fund at 97 %.



5. Summary

We developed a framework for financial data mining of the Vanguard family of funds so that we could extract simple financial metrics. Many of the pages that we have encountered had dynamic content rendered with JavaScript. There is no simple tool for data mining when JavaScript is encountered and the most robust and reliable tool that we have found was a live browser (*Firefox)*. Manipulating the browser with automatic tools presented its own set of problems and we could only wish for a more predictable JavaScript rendering engine to ease the burden of datamining. This is a great opportunity for future work which, if accomplished would obviate the need for gui manipulation tools like *xdotool* and the live browser paradigm. While a number of JavaScript engines exist, few are written in Java, one that has great promise is *GraalVM* (https://en.wikipedia.org/wiki/GraalVM) and could be a next generation approach to

the data-mining problem. Another issue we had with our operation was the timing. We used open-loop control of the time-critical rendering and download features because we lacked good call-back operation from our browser manipulation, thus screening time took up to 2 hours to complete. For a cron job that runs once per day, this might be considered ok, by some, but it is not an easy amount of time to wait during debug cycles. You may view our work at: https://tinyurl.com/docjavaVanguard.

6. References

[Investor] https://investor.vanguard.com/mutual-funds/profile/vmmsx

[Investor] https://investor.vanguard.com/etf/profile/portfolio/vglt

[Lynch] “Once up on wall Street” by Peter Lynch, Simon & Shuster, 2nd Edition, 2000.

[Lyon 18B] “The Truth Behind PEG” by Douglas A. Lyon, Stocks and Commodities, V.36:09(22-24), Sept, 2018.

[Man] http://manpages.ubuntu.com/manpages/trusty/man1/xdotool.1.html

[Mitchell] “Instant Web Scraping with Java”, Ryan Mitchell, 2013 Packt Publishing

Author information

Douglas A. Lyon (M'89-SM'00) received the Ph.D., M.S. and B.S. degrees in computer and systems engineering from Rensselaer Polytechnic Institute (1991, 1985 and 1983). Dr. Lyon has worked at AT&T Bell Laboratories at Murray Hill, NJ and the Jet Propulsion Laboratory at the California Institute of Technology, Pasadena, CA. He was Chairman of the Electrical and Computer Engineering Department and presently serves as a professor of Computer Engineering at Fairfield University, in Fairfield CT, a senior member of the IEEE and President of DocJava, Inc., a consulting firm in Connecticut. Dr. Lyon has authored or co-authored four books and 50 journal publications. Email: lyon@docjava.com. Web: http://www.DocJava.com.

Predrag L. Bokšić has a degree in physics (PMF Novi Sad, Serbia in 2008). He is a Java programming enthusiast with particular interests in computer graphics, chaos theory, and AI. He works in the localization and web services testing areas.