

Research Note: Scraping Financial Data from the Web Using R Language

Vlad Krotov
Matthew Tennyson
Murray State University

ABSTRACT: The main goal of this research note is to educate business researchers on how to automatically scrape financial data from the World Wide Web using the R programming language. This paper is organized into the following main parts. The first part provides a conceptual overview of the web scraping process. The second part educates the reader about the Rvest package—a popular tool for browsing and downloading web data in R. The third part educates the reader about the main functions of the XBRL package. The XBRL package was developed specifically for working with financial data distributed using the XBRL format in the R environment. The fourth part of this paper presents an example of a relatively complex web scraping task implemented using the R language. This complex web scraping task involves using both the Rvest and XBRL packages for the purposes of retrieving, preprocessing, and organizing financial and nonfinancial data related to a company from various sources and using different data forms. The paper ends with some concluding remarks on how the web scraping approach presented in this paper can be useful in other research projects involving financial and nonfinancial data.

Keywords: web scraping; HTML; CSS; XML; XBRL; R; RStudio.

INTRODUCTION

Financial data are increasingly distributed over the World Wide Web (the web) using HTML, CSS, XML, XBRL, and other digital formats. For example, the Securities and Exchange Commission (SEC) has been mandating filing of financial statements in plain text or HTML formats and publishing these data via the Electronic Data Gathering, Analysis, and Retrieval System (EDGAR) database since 1993 (Shin 2003). In 2009 the SEC mandated all publicly listed companies to start filing certain financial reports in XBRL format by 2011 (Rashty 2013). Other countries have followed suit in mandating that companies make their financial reports accessible to the public via the web. All these regulations, together with growing use of information and communication technologies (ICTs) to power business operations, have resulted in an exponential growth in the volume of financial data available on the web. These financial data, when combined with other relevant qualitative and quantitative data on the web, can be used by academic and industry researchers to answer a number of old and new research questions with more rigor, precision, and timeliness.

Harnessing these vast volumes of financial data requires a variety of skills (Munzert, Rubba, Meißner, and Nyhuis 2015). Besides vast volume, there are three other common issues associated with using financial and other types of data available on the web: variety, velocity, and veracity (Goes 2014). First, web data come in a variety of formats and rely on various technological and regulatory standards (Basoglu and White 2015). Second, these data are characterized by extreme velocity: they are generated in real time and are continuously updated and modified. Third, these data are also characterized by veracity. Due to the voluntary, democratic, and often anonymous patterns of interactions on the web, there is an inherent uncertainty associated with availability and quality of web data. A researcher can never be completely sure whether the needed data are available on the web and whether they will be available in the future. Moreover, the quality and completeness of these data are usually uncertain too.

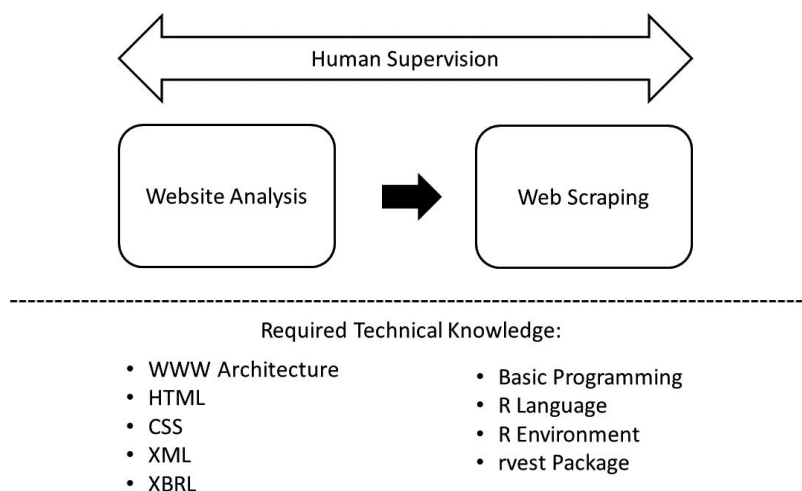
Given these challenges associated with using these “big financial data” from the web, harnessing these data requires a highly customizable and programmatic approach. Unfortunately, ready-made tools for accessing web data do not always work.

XXX.

Editor's note: Accepted by Miklos A. Vasarhelyi.

Submitted: March 2017
Accepted: February 2018
Published Online: February 2018

FIGURE 1
Web Scraping in R: An Overview



Because of that, many research projects relying on big datasets from the web require creating custom tools for automatically retrieving these data.

One popular and easily customizable platform for accessing and analyzing web data is R, which is one of the most widely used programming languages in data science (Lander 2014). R can be used not only to collect financial and other data from the web, but also to subject the collected data to myriad analysis techniques once the data are collected (or “scraped”). Thus, the main goal of this research note is to educate researchers and practitioners about the basic web scraping infrastructure in R and how this infrastructure can be used for extracting financial and nonfinancial data from the web. This research note contains enough information so that a reader with a basic understanding of programming and a familiarity with a few fundamental web technologies (e.g., HTML, CSS, and XML) can start using R for developing tools for automated collection of financial and nonfinancial data from various web sources.

WEB SCRAPING IN R: AN OVERVIEW

Web scraping is defined here as programmatic extraction, preprocessing, and organization of data from the web for further quantitative or qualitative analysis of these data. Web scraping consists of two intertwined phases: website analysis and web scraping (see Figure 1). Due to the velocity, variety, and veracity of web data, both phases cannot be fully automated and require at least some degree of human involvement.

During the website analysis phase, a researcher should analyze the structure of a website (or several websites—depending on how complex the data-collection task is) and inspect its underlying code. The main goal of the analysis phase is to determine what webpage elements contain the needed data. This analysis requires a basic understanding of the World Wide Web (WWW) architecture and some of the most commonly used technologies for storing, displaying, and transmitting information on the web: HTML, CSS, XML, and XBRL. A basic understanding of the HTML and XBRL mark-up languages is especially important since the EDGAR database distributes financial data of publicly listed corporations in these formats. Knowledge of CSS is also useful in cases where financial data are retrieved from a company’s website directly (as opposed to going to EDGAR) or from another web resource used to supplement financial data (e.g., letters to shareholders or news reports). The reason for that is that companies often use HTML and CSS to format their financial statements or other data that they distribute through their corporate websites. One can use specialized website analysis tools (e.g., the SelectorGadget add-on or extension for the Google Chrome browser) to determine what CSS elements contain the needed data. A good knowledge of XML and XML-related technologies is useful for understanding XBRL, since XBRL uses XML as its meta-language.

Once a website is analyzed and the location of the needed data in the underlying code of a webpage is determined, one can proceed to the second phase: web scraping. On the technical level, web scraping consists of a variety of tasks related to downloading, cleaning, organizing, and preprocessing data from the web for the purposes of obtaining a clean dataset. Web scraping can be done using a variety of programming languages (e.g., R, Python, Java) and ready-made software tools (e.g., the Web Scraper extension for Google Chrome or the import.io web tool). This research note discusses web scraping using the R

language for several reasons. The main reason is that R language is increasingly becoming a *de facto* standard in data science (Lander 2014). With thousands of free analysis packages available in R, R can be used for carrying virtually all known forms of basic and advanced analysis of quantitative and qualitative data (Comprehensive R Archive Network [CRAN] 2017). Moreover, R can be used to formally describe every step of data collection and analysis. This contributes to the reproducibility of research—one of the hallmarks of science (Peng 2011). Finally, the R environment already contains several useful, stable, and highly customizable packages that can be used for retrieving data from the web in various formats (e.g., HTML, CSS, XML, and XBRL). A reader wishing to learn more about any of the technologies used in web scraping (including R language) should refer to Appendix A, where we list the recommended resources.

Web scraping using R requires a basic understanding of programming in general, R syntax, the RStudio environment, as well as the basic functions of the Rvest and XBRL packages. The Rvest package was developed for harvesting HTML, CSS, and XML ~~as~~ data from the web (Wickham 2016). XBRL is a specialized package for retrieving and analyzing financial data in XBRL format (Bertolusso and Kimmel 2016). The functionalities of the two packages are covered in detail in the subsequent two sections.

RVEST PACKAGE

This section is devoted to the functionality of the Rvest package—a popular R package used for accessing and parsing web data in various formats. The first part provides an overview of the package. The second part provides a simple illustration of how the package can be used for retrieving financial data from the web.

Package Overview

Rvest is a popular web scraping package developed by Hadley Wickham (2016). Rvest contains functions that can be used for simulating sessions of a web browser or web crawler (it comes in handy when there is a need to browse through many webpages one at a time), as well as numerous functions for retrieving data from documents that use the HTML, XML, CSS, and JSON formats. Some of the most essential functions are listed in Table 1. Many other functions are available as a part of the Rvest package (see Wickham 2016).

Rvest Usage Example

The R script shown in Figure 2 provides a simple example of how the Rvest package can be used for accessing financial data. The script uses a third-quarter 10-Q statement by Berkshire Hathaway Inc. available from EDGAR. Balance sheet data are retrieved from the HTML page available through EDGAR that contains the Form 10-Q report for the third quarter.

Once balance sheet data are saved into a data frame (named “balance_sheet_table” in this example), individual values from the balance sheet can be accessed and used in calculations. For example, certain accounting ratios can be calculated. Alternatively, one can match the balance sheet with other data related to the company and available on the web. The Rvest package can be used in a similar fashion to obtain quantitative and qualitative data from the same HTML document or other sources on the web.

XBRL PACKAGE

This section is devoted to the functionality of the XBRL package—an R package developed specifically for accessing financial information in XBRL format. Just like the section devoted to Rvest, it consists of two main parts. The first part provides an overview of the XBRL package. The second part provides a simple illustration of how the package can be used for retrieving financial data from the web.

XBRL Package Overview

XBRL is a free package developed by Roberto Bertolusso and Marek Kimmel (Bertolusso and Kimmel 2016), two statisticians from Rice University. The package contains a number of functions that can be used for extracting financial information from an XBRL instance file and the associated taxonomies. Extracting and processing XBRL files can take some time. For example, an XBRL instance file for a large corporation can contain several megabytes worth of data—and this does not include related taxonomies. To save time and computing resources, the XBRL package can be used in a “piecemeal” fashion, where each element of an XBRL instance file is extracted and processed separately, depending on the needs of a researcher. For that, one needs to use a variety of low-level functions described in the manual supplied with the XBRL package (see Bertolusso and Kimmel 2016). Each of the low-level functions uses one of the arguments described in Table 2. The functions themselves are described in Table 3.

TABLE 1
Some Functions of the Rvest Package^a

Function Usage and Purpose	Arguments
Usage: <code>read_html(x, ..., encoding = "")</code>	<ul style="list-style-type: none"> • x: A URL, a local path, or a string containing HTML code that needs to be read
Purpose: This function reads the HTML code of a web page from which data are to be retrieved	<ul style="list-style-type: none"> • ...: Additional arguments can be passed to a URL using the GET() method • encoding: specify encoding of the web page being read
Usage: <code>html_nodes(x, css, xpath)</code>	<ul style="list-style-type: none"> • x: A document, a node, or a set of nodes from which data are selected • css, xpath: A name of a CSS element or an XPath 1.0 link can be used to select a node
Purpose: This function is used to select specific elements of a web document. To select these specific elements, one can use CSS elements that contain the needed data or use XPath language to specify the "address" of an element of a web page	
Usage: <code>html_session(url, ...)</code>	<ul style="list-style-type: none"> • URL: Address of a web page where browsing starts
Purpose: This function allows starting a web browsing session to browse HTML pages for the purpose of collecting data from them	<ul style="list-style-type: none"> • ...: Any additional httr config commands to use throughout session
Usage: <code>html_table(x, header = NA, trim = TRUE, fill = FALSE, dec = ".")</code>	<ul style="list-style-type: none"> • x: A node, node set or document
Purpose: This function can be used to read HTML tables into data frames (a commonly used data structure in R) and it can be especially useful for reading HTML tables containing financial data	<ul style="list-style-type: none"> • header: If NA, then the first row contains data and not column labels • trim: If TRUE, the function will remove leading and trailing whitespace within each cell • fill: If TRUE, automatically fill rows with fewer than the maximum number of columns with NAs • dec: The character used as decimal mark for numbers

^a Source: Wickham (2016).

FIGURE 2
A Simple Example of Using the Rvest Package

```
#This script requires Rvest package to be installed and activated
library(Rvest)

#The variable url contains a URL to the 10-Q document published via EDGAR
url <- "https://www.sec.gov/Archives/edgar/data/1067983/000119312516760194/d268144d10q.htm"

#read_html() function from Rvest package used to read HTML code from page
tenqreport_html <- read_html(url)

#xpath used to retrieve HTML code specifically for balance sheet table
balance_sheet_html <- html_nodes(tenqreport_html, xpath='/html/body/document/type/sequence/filename/description/text/table[7]')

#HTML code related to the balance sheet table is removed so that only balance sheet data is obtained
balance_sheet_list <- html_table(balance_sheet_html)

#Balance sheet data is saved from a list (a data structure that html_table function returns) into a data frame structure
balance_sheet_table <- balance_sheet_list[[1]]
```

TABLE 2
Arguments Used by XBRL Package Functions^a

Argument	Description
file	A local file name or a URL for an XBRL document.
doc	A pointer to an external memory source where XBRL data are stored. This pointer is returned by the <code>xbrlParse</code> function.
arcType	Represents type of XBRL data. Can be “presentation,” “calculation,” or “definition.”

^a Source: Bertolusso and Kimmel (2016).

The comprehensive function `xbrlDoAll()` combines all of the functionality of the low-level functions into one single function. This function performs all the steps necessary to extract a collection of various data frames, starting with an XBRL instance file (Bertolusso and Kimmel 2016). These data frames, together with relationships among them, are visualized in Figure 3.

The function `xbrlDoAll()` is used as follows (Bertolusso and Kimmel 2016):

```
xbrlDoAll(file.inst, cache.dir = “xbrl.Cache,” prefix.out = NULL, verbose = FALSE, delete.cached.inst = TRUE)
```

The arguments used by the function `xbrlDoAll()` are explained in Table 4.

XBRL Usage Example

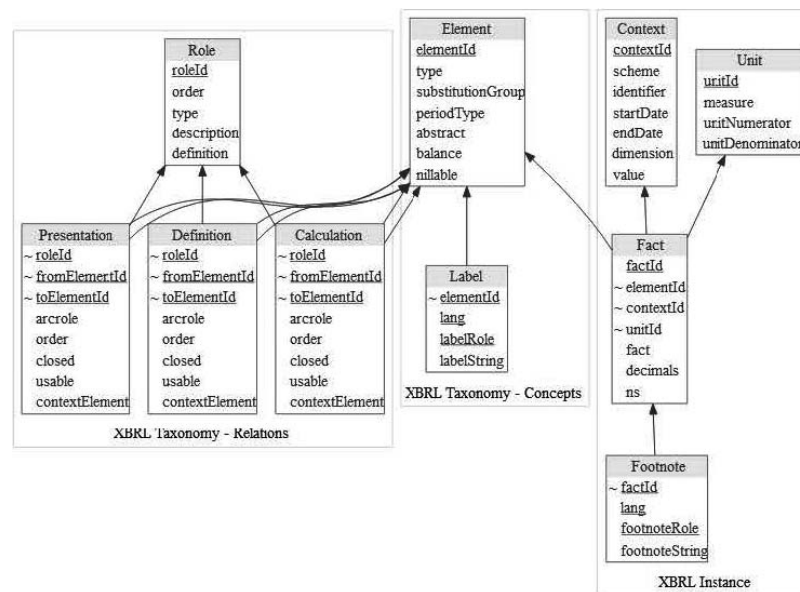
The simple example provided here uses a holistic approach to XBRL instance document discovery. All the information related to an XBRL instance file is extracted with the help of the `xbrlDoAll()` function. Just like the previous example involving Rvest, the script shown in Figure 4 uses a third-quarter Form 10-Q statement submitted by Berkshire Hathaway Inc. The difference is that the data are now retrieved from an XBRL report submitted in XMRL format, which is also retrieved through the EDGAR web database.

Note that the resulting list named “`xbrl.vars`” contains ten data frames with information related to the elements read from the instance document (one data frame for each entity shown in Figure 3). The data frame related to facts (that is specific financial data being distributed via the instance document) is of particular interest, since researchers may want to do calculations with actual financial data. Thus, this data frame contains a complete set of attribute values for each fact, together with its numeric value stored in the data frame’s column with the name “fact.”

TABLE 3
Low-Level Functions in the XBRL Package

Function	Used on	Purpose	Returns
<code>xbrlParse(file)</code>	Any XBRL file	XBRL file	A pointer to external memory used by the low-level functions. This can be thought of as making a local copy of the file for further analysis.
<code>xbrlFree(doc)</code>	Any XBRL file	Releases the external memory used for storing the contents of an XBRL file locally.	NULL
<code>xbrlGetSchemaName(doc)</code>	XBRL instance document	Extracts the name of the XBRL schema associated with a file.	A character vector
<code>xbrlGetLinkbaseNames(doc)</code> <code>xbrlGetImportNames(doc)</code>	XBRL taxonomy document	Extracts the names of documents linked to the taxonomy during the process of discovery.	A character vector
<code>xbrlProcessContexts(doc)</code> <code>xbrlProcessFacts(doc)</code> <code>xbrlProcessUnits(doc)</code> <code>xbrlProcessFootnotes(doc)</code>	XBRL instance documents or taxonomies	Extracts information from taxonomy and instance files in the form of data frames.	A data frame

FIGURE 3
XBRL Data Frames and Their Relationships



Source: Reprinted from Bergant (2015).

TABLE 4
Arguments of the xbrlDoAll Function

Argument	Description
file.inst	XBRL instance file name. Can be a URL or a path to a local file.
cache.dir	The name of a local directory that should be used for storing discovered taxonomies. If nothing is supplied (i.e., the value of the parameter is NULL), no caching (or storing) of discovered taxonomies is done.
prefix.out	Prefix to be used for creating CSV files with the content of the data frames produced.
verbose	Should be set to TRUE if one wants to see the details of the process used for discovery and analysis of XBRL documents.
delete.cached.inst	Should be set to TRUE if one wants to delete downloaded XBRL instance, schema, and linkbase files.

FIGURE 4
A Simple Example of Using XBRL Package

```
# Setting stringsAsFactors = FALSE forces R to treat string or text data as text and
# not as values or levels of a particular variable (something that is done by default in R)
options(stringsAsFactors = FALSE)

# XBRL package library is loaded
library(XBRL)

# The variable url contains the link to the XBRL instance file (3d Quarter Form 10-Q
# filed by Berkshire Hathaway via EDGAR) to be analyzed
url <- "https://www.sec.gov/Archives/edgar/data/1067983/000119312516760194/brka-20160930.xml"

# xbrlDoAll() function creates a list of data frames (the list is named "xbrl.vars")
# containing all the information extracted from the instance file.
xbrl.vars <- xbrlDoAll(url, verbose=TRUE)
```

210

FIGURE 5
Collected Data after Formatting

	A	B	C	D	E	F	G	H	I	J
	Year	Net Income	Shareholders Equity	Total Assets	ROE	ROA	Sentiment Syuzhet	Sentiment Bing	Sentiment Afinn	Sentiment NRC
1										
2	2009	\$8,055,000,000	\$131,102,000,000	\$297,119,000,000	0.06	NA	94.10	53.00	100.00	119.00
3	2010	\$12,967,000,000	\$157,318,000,000	\$372,229,000,000	0.08	0.04	95.80	28.00	105.00	156.00
4	2011	\$10,254,000,000	\$164,850,000,000	\$392,647,000,000	0.06	0.03	108.80	38.00	106.00	119.00
5	2012	\$14,824,000,000	\$187,647,000,000	\$427,452,000,000	0.08	0.04	111.15	52.00	114.00	142.00
6	2013	\$19,476,000,000	\$221,890,000,000	\$484,931,000,000	0.09	0.04	122.55	67.00	144.00	158.00
7	2014	\$19,872,000,000	\$240,170,000,000	\$526,186,000,000	0.08	0.04	137.55	70.00	119.00	169.00
8	2015	\$24,083,000,000	\$255,550,000,000	\$552,257,000,000	0.09	0.04	121.05	52.00	122.00	142.00
9	2016	\$24,074,000,000	\$283,001,000,000	\$620,854,000,000	0.09	0.04	120.55	50.00	132.00	171.00
10										

Source: XXX.

A COMPLEX WEB SCRAPING EXAMPLE

This section contains an integrated example of using both the Rvest and XBRL packages to automate a relatively complex web scraping task. The first part of this section provides an overview of the web scraping task. The second part provides an annotated R script used for automating the data-collection task.

Web Scraping Task Overview

The web scraping task is initiated to collect data that can potentially answer the following research question: Is there a relationship between annual performance of a company and the sentiment expressed in an annual report? Among other things, the web scraping task involves the following:

- Retrieving financial data for Berkshire Hathaway Inc. from two different web sources: (1) EDGAR, and (2) the company's corporate website (<http://www.berkshirehathaway.com>).
- The two data sources provide two different units of observation. In the case of EDGAR, financial data are retrieved in the form of XBRL instance documents (10-K reports). Berkshire's website is used to retrieve annual reports in PDF format for the period corresponding to the instance documents.
- Simple calculations (ROE and ROA) involving basic logic (e.g., whether data from the previous period are available) are performed based on the data obtained from the 10-K reports.
- Annual reports are subjected to sentiment analysis to calculate a sentiment score for each of the annual reports using four different scoring algorithms: Syuzhet, Bing, AFINN, and NRC. These algorithms are implemented as a part of the Syuzhet R package (see Jockers 2017).
- All the data are saved into an Excel table (see Figure 5). The Excel format is quite popular among researchers; it can be used for both programmatic and manual data manipulation and analysis.

FIGURE 6

An Integrated Example of Using Rvest and XBRL for Automating a Complex Web Scraping Task in R

Panel A: XXX

```

#Load required packages
require(Rvest)
require(XBRL)
require(pdftools)
require(syuzhet)

#Set working directory
setwd("C:/Users/[name removed]/Google Drive/Research/JETA/Code")

#Store strings as strings, rather than factors, inside data frames
options(stringsAsFactors=FALSE)

#Set filters
type <- "10-K"      #read annual statements
numStmts <- 8       #number of statements to be read (the most-recent statements will be read)

#Build the URL to get the list of reports
baseQueryUrl <- "https://www.sec.gov/cgi-bin/browse-edgar?action=getcompany&CIK=0001067983&type="
queryUrl <- paste(baseQueryUrl, type, sep="")

#Get and parse the HTML for the list of reports
reportListHtml <- read_html(html_session(queryUrl))
documentLinks <- html_nodes(reportListHtml, "a[id=\"documentsbutton\"]")
documentLinkUrls <- data.frame(xml_attrs(documentLinks))

#Create frame in which data results will be stored
all_data <- data.frame(Year=character(0),
                      NetIncome=character(0),
                      ShareholdersEquity=character(0),
                      TotalAssets=character(0),
                      ROE=character(0),
                      ROA=character(0),
                      SentimentSyuzhet=character(0),
                      SentimentBing=character(0),
                      SentimentAfinn=character(0),
                      SentimentNRC=character(0))

#Create columns for the data frame
colnames(all_data) <- c("Year",
                      "NetIncome",
                      "ShareholdersEquity",
                      "TotalAssets",
                      "ROE",
                      "ROA",
                      "SentimentSyuzhet",
                      "SentimentBing",
                      "SentimentAfinn",
                      "SentimentNRC")

#This will store the "Total Assets" value for the previous year (used in ROA calculation)
prevTA <- -1

```

(continued on next page)

FIGURE 6 (continued)

Panel B: XXX

```

#Read each statement one at a time
for(stmt in numStmts:1)
{
  #Read HTML for the current statement
  filingDetails <- toString(documentLinkUrls[1,stmt])
  filingDetailsUrl <- paste("https://www.sec.gov", filingDetails, sep="")
  filingDetailsHtml <- read_html(html_session(filingDetailsUrl))

  #Read the Data Files html table which should contain the XBRL document
  dataFilesTable <- html_nodes(filingDetailsHtml, "table[summary=\\"Data Files
    \"]")
  tableRows <- html_nodes(dataFilesTable, "tr")

  for(row in 1:length(tableRows))
  {
    #Specifically find the XBRL document from among the Data Files
    if(grepl("XBRL INSTANCE DOCUMENT", tableRows[row]))
    {
      #Get the URL of the XBRL document itself
      link <- html_node(tableRows[row], "a")
      linkUrl <- paste("https://www.sec.gov", xml_attr(link, "href"), sep="")
      writeLines(linkUrl) #print the url of the XBRL document

      #Read the data from the XBRL instance file
      xbrl.vars <- xbrlDoAll(linkUrl, cache.dir="XBRLcache", prefix.out="out" , verbose=FALSE)

      #Determine the date of the annual statement based on the name of the XBRL file
      year <- substr(linkUrl, nchar(linkUrl)-11, nchar(linkUrl)-8)
      month <- substr(linkUrl, nchar(linkUrl)-7, nchar(linkUrl)-6)
      day <- substr(linkUrl, nchar(linkUrl)-5, nchar(linkUrl)-4)

      #Retrieve the "facts" from the XBRL
      values <- xbrl.vars$fact

      #Find the positions of Net Income (NI), Shareholders Equity (SE), and Total
      Assets (TA) in the facts data frame
      regex <- paste(year,"-",month,"-",day,"(_0)?","$",sep="")
      indexNI <- which(values$elementId=="us-gaap_NetIncomeLoss" & grepl(regex, values$contextId))
      indexSE <- which(values$elementId=="us-gaap_StockholdersEquity" & grepl(regex, values$contextId))
      indexTA <- which(values$elementId=="us-gaap_Assets" & grepl(regex, values$contextId))

      #Obtaining NI, SE, and TA values
      NI <- as.numeric(values$fact[indexNI]) #NI contains income for both the year &
      the 4th quarter (we only use the annual value)
      SE <- as.numeric(values$fact[indexSE])
      TA <- as.numeric(values$fact[indexTA])

      #print key values to output
      writeLines(paste("Net Income: ", NI[1]))
      writeLines(paste("Shareholders Equity: ", SE))
    }
  }
}

```

(continued on next page)

FIGURE 6 (continued)

Panel C: XXX

```

writelines(paste("Total Assets: ", TA))
writelines("")

#Download a corresponding Letter to Shareholders and read into a character vector
txt <- pdf_text(paste("http://www.berkshirehathaway.com/letters/",year, "ltr.pdf",sep=""))
all_txt <- paste(txt, sep="", collapse="")

#Calculate sentiment score using four different methods
sentimentSyuzhet <- get_sentiment(all_txt, method="syuzhet")
sentimentBing <- get_sentiment(all_txt, method="bing")
sentimentAfinn <- get_sentiment(all_txt, method="afinn")
sentimentNRC <- get_sentiment(all_txt, method="nrc")

#If available, use the previous Total Assets to calculate ROA
if(prevTA!=1)
  ROA <- 2*NI[1]/(TA+prevTA)
else
  ROA <- "NA"

#Consolidate all the information about this statement into a single dataframe
stmt_info <- data.frame("Year"=year,
                        "NetIncome"=NI[1],
                        "ShareholdersEquity"=SE,
                        "TotalAssets"=TA,
                        "ROE"=NI[1]/SE,
                        "ROA"=ROA,
                        "SentimentSyuzhet"=sentimentSyuzhet,
                        "SentimentBing"=sentimentBing,
                        "SentimentAfinn"=sentimentAfinn,
                        "SentimentNRC"=sentimentNRC)

#Write the current statement info to the frame where all data is stored
all_data <- rbind(all_data, stmt_info)

#this year's TA is stored as the previous year's TA to use in the next loop iteration
prevTA <- TA
}
}
}

#Finally, write all the collected data to an output file
write.table(all_data, file="output_data.csv", sep=";", col.names=TRUE, row.names=FALSE)

```

The data-collection task is deliberately small. The final dataset contains only eight observations for one single company, which is Berkshire Hathaway Inc. (see Figure 5). A small data-collection task is selected to minimize the execution time of the script. Still, even a short example like this one will run for almost one minute on an average office computer. Naturally, this example can be extended to collect additional data from other companies to produce a large dataset. The fact that this small data-collection task takes almost one minute for a computer to complete shows how much time will be saved by automating this task with the help of the R language.

Annotated R Code Used for Web Scraping

The annotated R code used for collecting the data shown in Figure 5 is provided in Figure 6. The code is also available in the form of an R script file from one of the author's GitHub accounts upon request.

IMPLICATIONS FOR RESEARCHERS

There is nothing new about obtaining financial data from the web. Numerous web databases (e.g., EDGAR) have been available for years (Debreceeny et al. 2005). Researchers have been downloading structured datasets containing financial data and analyzing them using computer tools for years. The real value of the web scraping toolset presented in this paper comes from several areas. First, in the era of Big Data, even simple manipulations with accounting numbers can be very tedious and time consuming if done manually. The R environment can be used to automate a number of simple and also complex data collection and data transformation processes and techniques involving a complex heuristic. Second, the approach outlined in this paper allows researchers to go beyond accounting data (which are already available in structured format from the numerous databases containing accounting data). Accounting facts can be combined with other sources of quantitative and even qualitative data for better understanding of a particular phenomenon. Third, using R for web scraping and subsequent analysis ensures reproducibility of accounting research (Peng 2011). Any manual manipulation of data may involve subjective choices and interpretations in relation to how data are retrieved, saved, cleaned, formatted, etc. Oftentimes, even research involving simple data manipulations cannot be replicated by other researchers due to the errors committed at various stages of data gathering and analysis (Dennis and Valacich 2014). Fourth, once financial and nonfinancial data are retrieved using R, they can be subjected to virtually all known forms of analysis implemented via thousands of user-generated packages available from

77 CRAN.

CONCLUSION

The World Wide Web is a vast repository of data. Many old and new research questions can be addressed by retrieving and analyzing these data. Unfortunately, web data are often unstructured or semi-structured, based on somewhat loose standards, and generated or updated in real time. As demonstrated in this research note, the R environment can be used as a platform for creating and modifying an automated tool for retrieving financial and nonfinancial data from the web. A “one size fits all” web scraping solution can hardly work due to the diverse nature of business research and the constantly and rapidly evolving World Wide Web. We are confident that the approach to web scraping outlined in this research note should serve as a good starting point for any accounting or general business research requiring web scraping.

REFERENCES

- Basoglu, K. A., and C. E. White, Jr. 2015. Inline XBRL versus XBRL for SEC reporting. *Journal of Emerging Technologies in Accounting* 12 (1): 189–199. <https://doi.org/10.2308/jeta-51254>
- Bergant, D. 2015. *Explore XBRL with R*. Available at: <https://github.com/bergant/XBRLFiles>
- Bertolusso, R., and M. Kimmel. 2016. *Package “XBRL.”* Available at: <https://cran.r-project.org/web/packages/XBRL/XBRL.pdf>
- Comprehensive R Archive Network (CRAN). 2017. *The Comprehensive R Archive Network*. Available at: <https://cran.r-project.org/>
- Debreceeny, R. S., A. Chandra, J. Cheh, D. Guithues-Amrhein, N. Hannon, P. Hutchison, D. Janvrin, R. Jones, B. Lamberton, A. Lymer, M. Mascha, R. Nehmer, S. Roohani, R. Srivastava, S. Trabelsi, T. Tribunella, G. Trites, and M. Vasarhelyi. 2005. Financial reporting in XBRL on the SEC’s EDGAR system: A critique and evaluation. *Journal of Information Systems* 19 (2): 191–210. <https://doi.org/10.2308/jis.2005.19.2.191>
- Dennis, A. R., and J. S. Valacich. 2014. A replication manifesto. *AIS Transactions on Replication Research* 1 (1): 1–4. <https://doi.org/10.17705/1attr.00001>
- Dynamic Web Solutions. 2017. *A Conceptual Explanation of the World Wide Web*. Available at: <http://www.dynamicwebs.com.au/tutorials/explain-web.htm>
- Goes, P. B. 2014. Editor’s comments: Big Data and IS research. *Management Information Systems Quarterly* 38 (3): iii–viii.
- Jockers, M. 2017. *Package “syuzhet.”* Available at: <https://cran.r-project.org/web/packages/syuzhet/syuzhet.pdf>
- Lander, J. P. 2014. *R for Everyone: Advanced Analytics and Graphics*. Boston, MA: Addison-Wesley.
- Munzert, S., C. Rubba, P. Meißner, and D. Nyhuis. 2015. *Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining*. Chichester, UK: John Wiley & Sons, Ltd.
- Peng, R. D. 2011. Reproducible research in computational science. *Science* 334 (6060): 1226–1227. <https://doi.org/10.1126/science.1213847>
- Rashty, J. 2013. An overview of XBRL compliance: Understanding the risks and liabilities. *The CPA Journal* 83 (8): 67–71.
- Tutorials Point. 2017. *HTTP Tutorial*. Available at: <https://www.tutorialspoint.com/http/>
- W3Schools. 2017. *HTML: The Language for Building Web Pages*. Available at: <https://www.w3schools.com/>
- Wickham, H. 2014. *Advanced R*. Boca Raton, FL: CRC Press.
- Wickham, H. 2016. *Package “rvest.”* Available at: <https://cran.r-project.org/web/packages/rvest/rvest.pdf>
- 78 World Wide Web Consortium (W3C). 2017. Available at: <https://www.w3.org/>
- XBRL International. 2017. *XBRL Essentials*. Available at: <https://specifications.xbrl.org/xbrl-essentials.html>

APPENDIX A

Additional Resources

Web Architecture

The following brief online tutorials are recommended for those who want to learn more about World Wide Web architecture:

- Dynamic Web Solutions 2017. *A Conceptual Explanation of the World Wide Web*. Available at: <http://www.dynamicwebs.com.au/tutorials/explain-web.htm>
- Tutorials Point. 2017. *HTTP Tutorial*. Available at: <https://www.tutorialspoint.com/http/>

World Wide Web Consortium is the ultimate source on the technologies and specifications related to the web:

- World Wide Web Consortium (W3C). 2017. Available at: <https://www.w3.org/>

HTML

A free online tutorial from w3schools.com on various aspects of HTML:

- Available at: <https://www.w3schools.com/html/default.asp>

CSS

A free online tutorial from w3schools.com on CSS:

- Available at: <https://www.w3schools.com/css/default.asp>

XML

A free online tutorial from w3schools.com on XML and a number of related technologies:

- Available at: https://www.w3schools.com/xml/xml_exam.asp

In addition to providing a rather thorough treatment of XML, the tutorial also has sections devoted to related technologies such as XML Namespaces, XML Schema, XML XPath, and XML XLink.

XBRL

A comprehensive collection of various articles and technical specifications documents describing XBRL and related issues is available from the XBRL International website:

- Available at: <https://www.xbrl.org/>

A good starting point for learning the technical aspects of XBRL is XBRL International's "how to" page for developers:

- Available at: <https://www.xbrl.org/the-standard/how/getting-started-for-developers/>

R and RStudio

We recommend the following book to people with a basic understanding of computer programming but no previous knowledge of R:

- Lander, J. P. 2014. *R for Everyone: Advanced Analytics and Graphics*. Boston, MA: Addison-Wesley.

The book contains an excellent introduction into various aspects of R language and contains a manual on installing and using RStudio. Many of the examples found in this note are based on this book.

For those already familiar with R, the following advanced text on R is recommended:

- Wickham, H. 2014. *Advanced R*. Boca Raton, FL: CRC Press.

Alternatively, one can access various articles and tutorials on R online:

- Available at: <https://www.r-bloggers.com/how-to-learn-r-2/>



For those wishing to get a practical introduction to data science in R and learning various related technologies and statistical techniques, the following online specialization from John Hopkins University is available in Coursera:

- Available at: <https://www.coursera.org/specializations/jhu-data-science>

Those who already have some knowledge of R, and wish to get straight into web scraping using R, should read the documentation for the following two R packages:

- Rvest: Available at: <https://cran.r-project.org/web/packages/rvest/rvest.pdf>
- XBRL: Available at: <https://cran.r-project.org/web/packages/XBRL/XBRL.pdf>

The following online tutorial by Darko Bergant contains a good overview and a practical demonstration on using the XBRL package:

- Available at: <https://github.com/bergant/XBRLFiles>

The Package “finreportr” can be used for retrieving financial statements from the SEC using R:

- Available at: <https://cran.r-project.org/web/packages/finreportr/finreportr.pdf>

Queries for jeta-15-01-03

1. Author: Please insert an acknowledgment, if desired, or mark to delete the XXX placeholder. Copyeditor
2. Author: Your copyright release form was not submitted at the time that the article page proofs were generated. If you have not already done so, then please send the copyright release form to nate@aaahq.org. If you need the form, then please contact Nate. Jan
3. Author: The in-text citation “Shin 2003” is not in the reference list. Please correct the citation, add the reference to the list, or delete the citation. Copyeditor
4. Author: The Rvest and Syuzhet are also shown as rvest and syuzhet throughout the article. Should the capitalization be consistent? If so, mark to revise as appropriate. Copyeditor
5. Author: For the sentence beginning “The difference is that the data are now ...” is XMRL correct here? Please review and revise if warranted. Copyeditor
6. Author: For the sentence beginning “The code is also available in the form of an R script file ...” I slightly modified the wording for clarity. Please approve as is, or mark to revise as preferred. Copyeditor
7. Author: For the sentence beginning “Fourth, once financial ...” I modified the wording for clarity and grammatical correctness. Please approve as is, or mark to revise as preferred. Copyeditor
8. Author: W3C 2017. Please provide an article title or an appropriate URL that lands on the desired page. Copyeditor
9. Author: AAA cannot accommodate color figures in the print journal. They have been rendered in greyscale. To appear as you intend, the original files may be posted as downloadable supplemental materials should you wish to do so. Please advise. Copyeditor
10. Author: Please provide a citation (and reference entry if needed) for the graphic used in Figure 5 at the XXX placeholder (or within text), or advise that the graphic is original to this article. Copyeditor
11. Author: Figure 6 has been split into panels to conform to AAA standard page length guidelines. Please note that the panel descriptions cannot be deleted due to current XML-related formatting constraints. Please review and provide panel titles at the XXX placeholders. Copyeditor
12. Author: re Table 4: Should “xbrlDoAll” in the title include () as in the rest of the document? Please review and revise as warranted. Copyeditor