Web Scraping Large Qualitative Datasets Using R: The Case of Dice.com

Abstract

This paper bridges the gap between the qualitative research tradition in Information Systems (IS) and the new, emerging paradigm of data science that relies on big datasets and technologically enabled inquiry. This is accomplished by proposing a process framework of technology-assisted data collection from the Web (or "web scraping"). This framework is designed to equip qualitative researchers with the technical and epistemological skills for harnessing the vast volumes of qualitative data available on the Web. From a technical standpoint, the paper educates IS researchers on how R, a popular programming language in data science, can be used for automatic retrieval and organization or large qualitative datasets. The approach to web scraping described in this paper is general enough to be customized to any research project relying on qualitative or quantitative data from the Web. Dice.com, a leading employment website for the IT industry, is used as a context for explaining web scraping with R. Thus, the methodology described in this paper contributes to the stream of research on IT workforce trends by allowing researchers to answer the traditional questions in relation to IT workforce trends with greater rigor, better precision, and more timeliness. Moreover, by utilizing the qualitative data collection technique explained in this paper, research on IT workforce trends can produce a number of new theoretical contributions and help researchers, educational institutions, companies, and recruiters gain a better understanding of various IT roles.

Keywords: *technology-enabled inquiry, web scraping, data science, R, qualitative research, IT workforce trends, dice.com*

Introduction

In the past, data used in social science research was scare and hard to collect (Munzert et al., 2015). The increasing digitalization and datification of social processes has resulted in zetabytes (billions of gigabytes) of data becoming available on the World Wide Web (the Web) (Cisco, 2016). Accordingly, social scientists are now facing the opposite problem: there is too much data available on the Web in differing formats (Munzert et al., 2015). These Web data often exceed information processing capacities of individual researchers or even large research teams (Indulska et al., 2012; Debortoli et al., 2016).

A substantial share of the data available on the Web is comprised of unstructured or semi-structured qualitative data stored in the form of Web pages, emails, tweets, blog posts, etc. (Watson, 2014). These data artifacts reflect numerous intricate processes, relationships, and interactions in the sociomaterial realm (Cecez-Kecmanovic et al., 2014). Thus, these vast volumes of data present qualitative researchers with a wealth of opportunities for gaining new insights about individuals, organizations or even geographical areas in real time (Constantiou and Kallinikos, 2015). Yet this "big data" can also become a big challenge for qualitative researchers who lack the technical skills or the epistemological foundations required to make use of these data (Müller et al., 2016). In fact, some scholars have expressed concerns that qualitative research may lose its relevance in the era of big data and technology-enabled inquiry. Lynne Markus (2014) explained this concern as follows:

> *The Big Data movement […] promises many benefits for IS research, for business, for government, etc.; however, I worry about IS research culture clashes between scholars who "have (big) data, will publish" versus those who do qualitative research and those who argue that "all IS research should make a contribution to 'theory'*.

In this paper, we show that in the era of big data and technology-enabled inquiry the traditional qualitative research skills are still relevant. Regardless of whether one uses big or small datasets, or whether the entire process of data collection and analysis is fully automated or manual, the old epistemological challenges of qualitative research remain and need to be dealt with. Among other things, these epistemological issues include: selecting and justifying a site for research, collecting and storing observational units, extracting units of analysis from the observational units, etc. Being thus equipped with the traditional qualitative research skills, a researcher can save time, money, and computing resources by avoiding collecting big data that is of poor quality, not aligned with the theory, or organized in a way that prevents further analysis.

These traditional qualitative research skills have to be supplemented with a variety of technology skills to handle the big, unstructured, and often "messy" qualitative datasets available on the Web (Munzert et al., 2015). This combination of qualitative and technical skills can enable qualitative researchers to answer old and new questions with more rigor, better precision, and in a more timely fashion. Thus, the integration of data science techniques potentially expands the depth and scope of current qualitative research in IS.

The focus of this paper is on the automated collection and organization of large qualitative datasets from the Web (further referred to as "web scraping"). Automating the process of data collection from the Web requires working knowledge of many Web technologies and practical experience with web scraping to find work-arounds for the numerous technical problems that may be encountered when trying to automatically retrieve data from the Web. Apart from the vast volume of data available, there are three other common issues associated with using these qualitative data from the Web: variety, velocity, and veracity (Goes, 2014). First, Web data come in a variety of formats. Each of these specific data formats is often governed by a variety of technical standards (e.g. HTML, CSS, XML, JSON, XBRL, MySQL, NoSQL, etc.) and context-specific conventions (e.g. how a particular Web designer actually uses these technologies for organizing data). Second, these data are characterized by extreme velocity: data are generated on the Web in real time and are continuously updated. Therefore, a dataset available from the Web may be evolving while a researcher is trying to collect it. Finally, veracity deals with the inherent uncertainty of data available on the Web. A researcher can never be completely sure about the extent to which a particular website will be accessible. Moreover, the quality of Web data (in terms of formatting, reliability, and validity) is frequently questionable. Because of these issues associated with data available on the Web, "ready-made" tools for web scraping do not always work. Virtually every research project relying on Web data requires a certain degree of tool modification and fine-tuning.

The process of retrieving and analyzing big, messy data becomes easier when a single, open source technological platform is used (Alharthi et al., 2017). In this paper, we explain how R, a popular programming language in data science, can be used for automatically retrieving and organizing large qualitative datasets from the Web. Our approach to web scraping is simple and general enough so that it can be customized for any research project relying on qualitative or quantitative data from the Web.

Dice.com, a leading employment website for IT professionals (Lee, 2007), is used as a context for explaining how web scraping can be done using R. We have selected this website in the hope that this paper can also contribute to the research on IT workforce trends (e.g. see Davis et al., 2003; Dattero and Galup, 2004; Abraham et al., 2006; Goles et al., Hirschheim et al., 2007; Sandvig, 2007; Kennan et al., 2008; Topi et al., 2010; Debortoli et al., 2014), which has been an important IS research stream since the late 1990's (Hirschheim and Klein, 2012). Using the approach outlined in this paper, researchers can retrieve large datasets comprised of job descriptions for various categories of IT professionals virtually in real time. These job descriptions can be a source of rich, current qualitative data on various workforce trends which are of great importance to organizations relying on IT and academic institutions offering educational programs for IT professionals. These data can be used to answer traditional research questions in this stream of research with more rigor, depth, and breadth, and also help researchers answer new, more forward-looking questions.

For example, these data can be used to answer one of the most important questions in IT workforce trends: what are the most-in demand IT skills? (e.g. Kennan et al., 2008). Due to large volumes of data available on the Web, this question can be answered for various IT roles and geographical areas. Alternatively, researchers can look into the

future needs of companies with respect to IT labor. This can be done by analyzing and comparing job descriptions for various roles to determine the need for a new set of IT skills or even a new IT role (e.g. Debortoli et al., 2014). Once a new IT role is derived from data, educational institutions can design a new curriculum that would equip students with these new, emerging skills and meet the current needs of employers. In fact, this is how Management Information Systems (MIS) was conceived as an educational and research field: by realizing that there was an increasing demand for technology professionals who understood the human and institutional aspects of implementing technology within an organization (George, 2017). Managers can incorporate these emerging IT skills requirements into their talent acquisition strategies and, thus, improve their future competitiveness.

Web Scraping: A Process Framework

In this section, we propose a process framework for web scraping. First, web scraping is defined and the main phases of web scraping are introduced. Second, we show how web scraping fits with the emerging data science paradigm. Third, we provide a summary of which technical and qualitative research skills need to be combined to enable automatic retrieval and organization of large qualitative datasets from the Web. The rest of this section is devoted to an in-depth overview of the phases comprising the web scraping framework proposed in this paper.

Web Scraping Defined

In this paper web scraping is defined as the technology-assisted collection and organization of data from the Web (Iacus, 2015; Munzert et al., 2015; Vargiu and Urru, 2013). Web scraping is viewed as a process Mohr, 1982; Soh and Markus, 1995) comprising several phases:

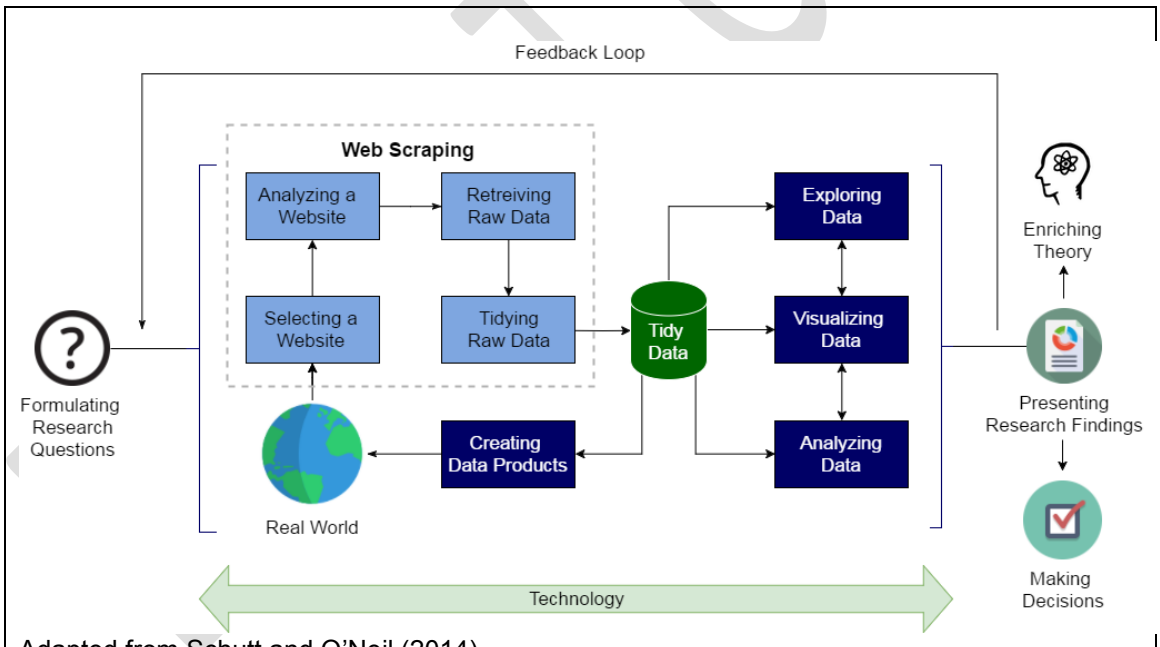
Pahse 1: Selecting a Website;
Pahse 2: Analyzing a Website;
Pahse 3: Retrieving Raw Data;
Pahse 4: Tidying Raw Data.

These phases are implemented to produce a “tidy” dataset that can be used for answering a research question (or a set of questions). These phases are listed in order - something that may suggest strict linearity of the web scraping process. Yet, this may not necessarily represent how web scraping works in reality. For most projects, scraping Web data requires moving back and forth between phases until “tidy data” is obtained. Furthermore, while the focus of this framework is on automatic collection and organization of qualitative (or textual) data available on the Web, this framework is also applicable to quantitative data collection as well.

Web Scraping and Data Science

The phases comprising web scraping can be viewed as early steps within a data science project (Schutt and O’Neil, 2014). A data science project (just like any other “normal science” research project) is often driven by a research question (see Figure 1). Naturally, the research question also determines the selection of a particular website

that can contain data to address the research question. The end result of web scraping is clean (or tidy) data (Wickham, 2014). Tidy data is “clean” (i.e., free from various “impurities”, such as formatting or mark-up symbols that are not needed for research) and organized in such a way that enables various forms of analysis. This tidy data can be visualized and subjected to exploratory or more formal confirmatory analysis. Depending on the nature of research (e.g. positivist or interpretive), there may or may not be a formal confirmation of one or more hypotheses formulated based on the research question. Therefore, hypotheses are not included in the process model. Moreover, an interpretivist research project may involve data interpretation while subsequently returning to the original research question (Walsham, 1993). Hence, a “feedback loop” is added to the model. The results of analysis are then presented in a form appropriate for the intended audience. The results, once presented, can either guide a decision maker within an organization (Watson, 2009) or be generalized to a theory in order to contribute to a theoretical understanding of a phenomenon (Markus and Robey, 1988). Tidy data may also be used for creating data products that are employed by the intended audience to gain a better understanding of the world around them (e.g. a website displaying the most in-demand IT competencies by state). Various technologies or a single integrated technology environment (e.g. R) can support virtually all steps of the data science process.



Adapted from Schutt and O’Neil (2014)

Figure 1. Web Scraping as a Part of the Data Science Process

Web Scarping Phases, Technical Knowledge, and Qualitative Research

Each of the phases in web scraping starts with addressing important epistemological issues associated with qualitative research design. These include:

- Justifying website selection.
- Reflecting on the organizational or social context within which the data is captured and managed.
- Defining the unit of observation and the units of analysis and how these units are related.
- Determining how data should be captured and stored so that it corresponds to the relevant theoretical concepts or units of analysis.

Furthermore, each web scraping phase requires some technical knowledge related to programming, Web technologies, and the legal framework governing the use of Web data (see Table 1). Thus, we view web scraping as a socio-technical work system requiring a broad range of skills in the technical and social domains (Alter, 2013). Some practical issues necessary to be addressed are also included in each of the phase.

Table 1. Web Scraping: Phases, Technical Knowledge, and Qualitative Research Skills				
Phase	Phase 1: Selecting a Website	Phase 2: Analyzing a Website	Phase 3: Retrieving Raw Data	Phase 4: Tidying Raw Data
Step Description	Selecting a website or a web repository (i.e. comprised of databases or files) that contains rich, quality data that can potentially answer a research question	Analyzing the underlying architecture of a web repository for the purpose of understanding how and by who the data is produced and managed, how data is stored, where the needed data is located, and how the needed data can be programmatically retrieved	Developing, debugging, and running a script for automatically retrieving the needed data from a web repository	Organizing clean data in a form where each dataset represents a particular unit of analysis, each concept of variable forms a column and each observation forms a row.
Qualitative Research Knowledge and Skills	Website selection should be driven by the initial research question and research design. The selection of the website should be theoretically justified (Yin, 1994; Benbasat et. al., 1987)	The researcher needs to identify the structure of the website in terms of its social hierarchies (e.g. who creates or who moderates data) and overall organization. This with the purpose of reconstructing the context of the data (Benbasat et. al., 1987; Denzin, 1994; Walsham, 1993)	The researcher needs to identify the different units of analysis in the website and their correspondence with the initial research design (Denzin and Lincoln, 1998; Yin, 1994).	Researchers should examine the data to make sure it is stored in a way that corresponds to theoretical categories (Eisenhardt, 1989; Yin, 1994; Corbin and Strauss, 2008).
Technical Knowledge and Skills	Basic understanding of the Web and sources of various types of data; legal framework around web crawling and web scraping	Web Architecture, HTML, CSS, XML, XBRL, JSON, MySQL, website-specific APIs, etc.	Programming skills (R, Python, etc.); knowledge of web scraping libraries (e.g. Beautiful Soup or rvest); basic understanding of networking (e.g. TCP/IP or HTTP)	Programming skills (R, Python, etc.); Popular file formats (e.g. Excel, JSON, CSV, etc.); Natural Language Processing (NLP) tools.

Phase 1: Selecting a Website

The first step in the automatic collection of qualitative data from the Web is selecting a website that contains data potentially capable of answering a research question (or questions). Note that the term "website" is used in a broad sense. For example, one can select several websites related to a theoretical phenomenon or phenomena and programmatically merge data together. A website can also be a collection of simple, HTML-based web pages, an online archive of files in various formats (e.g. in Excel, CSV, JSON, etc.), and a relational (e.g. MySQL) or non-relational database (e.g. NoSQL).

The justification of selecting a website is similar to the process of justifying a site when doing traditional qualitative research. For the purposes of web scraping, we suggest that websites are selected through a theoretical sample. This means they are selected either because they are extreme cases of a theoretical phenomenon of interest or because they exhibit characteristics that are aligned with the theoretical framework guiding the study (Benbasat et. al., 1987). For example, a theory that concerns conflict could be used to justify the selection of a website that focuses on disputes of any sort. In this case, researchers could claim that their study extends such a conflict to theory to explain online phenomena (e.g. Silva et al., 2009). The most important practical consideration is whether the website contains data that can potentially answer a research question.

When selecting a website, one can also adapt Yin's (1994) recommendations for selecting sites for case studies. Yin suggests three additional types of theoretical justifications. The first is selecting a critical site, which presents ideal circumstances for testing a theory. This is somewhat analogous to exposing a hypothesis to a "critical experiment" in natural sciences. The second approach to theoretical justifications of site selections corresponds the specific conditions of a website that make it worthy of investigation. These conditions should make the website a source of a "typical case" of a phenomenon of interest. The third reason is revelatory, that is when the site provides access to a phenomenon that has not been studied before. In this case, the site is selected simply because it is the only feasible source of data on the phenomenon of interest.

Selecting a website for web scraping does not require technical skills beyond a basic understanding of the Web architecture and the ability to use a Web browser for accessing information. Yet it is a very important step given the veracity issue associated with Web data. A researcher needs to select a website that not only contains data that can be potentially used for addressing a research question, but also has adequate data quality and availability. The data needs to have the level of detail or richness suitable for a research project. Moreover, there should be a certain degree of assurance that data will still be available by the time automated data retrieval commences.

When inspecting the website, one should also note if some data elements are missing in some observations, as this may create the situation where not all observations contain data about all needed theoretical concepts. This initial data inspection will help the researcher acquire a sense of the overall richness and quality of data. An important practical consideration here is whether any of the detected data issues can potentially

stand in the way of answering the research question. If that's the case, then the researcher may be better off finding another potential source of data.

During this step a researcher should also reflect on the legality and ethics of data collection from a particular website. Legality of web crawling and web scraping are still a somewhat "grey area" in the legal field (Snell and Care, 2013). Yet the current consensus among the analytics community and legal professionals is that web scraping of publicly available Web data is legal if (Snell and Care, 2013):

1. Web crawling and web scraping are not explicitly prohibited by the website's "Terms of Use" policy
2. Data is not explicitly copyrighted (though copyrighted data can still be used if the use complies with the "Fair Use" principle)
3. Web scraping is done for a legitimate, non-commercial, and non-fraudulent purpose (e.g. research)
4. Crawling and scraping a website does not cause physical damage to the website (e.g. by overloading a web server with HTTP requests)

Researchers should start their web scraping project by examining the "Terms of Use" for the website they wish to crawl and scrape. While some websites explicitly prohibit automated crawling and data scraping, many websites do not prohibit or explicitly allow these practices. Some websites view these practices as a way of engaging consumers, leveraging the "wisdom of crowds", and expanding their business. These websites (e.g., Twitter, Yelp, Dice, Glassdoor, etc.) offer Application Programming Interfaces (APIs) that make it easier for programmers to develop automatic data collection tools. The use of these APIs is often governed by specific rules outlined in the "Terms of Use". Large-scale or real-time data retrieval using these APIs often requires purchasing a premium API account from a website.

Legal conditions expressed in a website's user policies usually address rather obvious ethical issues associated with the use of data from the website. Unintended ethical consequences of using technology and data are, by definition, harder to predict (Light and McGrath, 2010). Because of that, researchers should always be mindful of the possible ethical consequences of their actions. For example, a finding derived from a research project relying on data collected from a website may unintentionally compromise privacy of individuals and organizations participating in the activities afforded by the website. It is also possible that some uses of data may reduce perceived value of the website in the eyes of the intended audience. It is hard (if not impossible) to foresee all the ethical issues arising from retrieval and use of data from the Web data and find a universally accepted way of addressing them. But being mindful of these potential unintended consequences is a good starting point.

To sum up, this phase requires addressing the following questions:

- Can this website be used for answering the research question (or questions) of the study?
- Can the website selection be justified from a theoretical standpoint?
- Is this website a quality source of rich data that can be used for answering the research question?

- Is it ethical or legal to crawl and scrape this website?

Phase 2: Analyzing a Website

After a particular website is selected as a potential source of qualitative data, it needs to be analyzed in detail to understand how data are organized, where the needed data elements are located, and how these data elements can be programmatically accessed using a particular technology platform. This step in automatic data collection also require some technical knowledge. For example, if the data intended for retrieval is stored in the form of webpages, then one needs to have a basic understanding of HTML and CSS. This is needed to examine the underlying source code of a webpage to determine which HTML or CSS elements contain the needed data. Often, data is stored and transmitted over the Web using XML. In this scenario, a good understanding of the structure of an XML document is required. A basic understanding of XML-related technologies (e.g. XPath for programmatic navigation through a document structure or XLink for understanding how XML documents are related) may also come in handy. JSON, a new light-weight data-interchanged format, is increasingly being used on the Web as well. If data is stored on the Web in a relational (e.g. MySQL) or non-relational (e.g. NoSQL) database, then one needs to have theoretical understanding of data modeling as well technical knowledge of the specific database technologies used for storing data on the Web. If one intends to use an API created by a website to facilitate data access by the user community, then the API manuals supplied by the website have to be examined. API's often use HTML, XML, and JSON for data interchange with user developed software applications.

Often, knowledge of several of the technologies mentioned above is required for scraping data from the Web. But this does not infer that one has to be an expert in any of these technologies (although that would in all likelihood help). One needs to have a basic understanding of how these technologies works to find where the needed data elements are located and how they can be accessed programmatically.

Researchers should also pay attention to the institutional or social hierarchies represented in the organization of the site. The main objective here is to identify the site's social organization. For example, who is it that decides what posts are included on a particular website? Is there any type of moderation on this website? This information is captured for reconstructing the context of where the data has been published. This in turn will be important for interpreting the meaning of the collected data (e.g. Benbasat et. al., 1987; Denzin, 1994; Walsham, 1993). Also, understanding the social hierarchies used to create and manage the data can shed further light on the quality of the data that the website contains.

In summary, this phase requires addressing the following questions:

- How is data organized from a technical standpoint? What markup languages are used?
- What mark-up elements contain data needed for answering the research question?
- Who, how, and when creates and manages the data? What are the social structures governing data creation and management?

Phase 3: Retrieving Raw Data

After a website or Web repository is analyzed, researchers should reflect on the fundamental issues inherent to qualitative data collection. Within the qualitative research tradition, data collection from the Web can be viewed as being analogous to unstructured interviews, in the sense that what is really relevant is to get information from the subjects of study without filters (Denzin and Lincoln, 1998). Prior to commencing the data retrieval stage, the researcher should reflect on how the units of observation are related to the theoretical concepts (or the units of analysis) needed for answering the research question in an analytical (or theoretical) way. This is necessary to make sure that all theoretically relevant data is retrieved and can be analyzed later.

Units of analysis should be distinguished from units of observation (Eisenhardt, 1989; Yin, 1994). A unit of observation is the data that is actually collected (e.g. an online review) by researchers. Units of analysis correspond to the main theoretical objects of a study, such as *sentiments* expressed in a review (theoretical concept 1) and how these sentiments relate to *purchasing intentions* (theoretical concept 2). Thus, a research study involving such units of analysis can be aimed at explaining the nature and consequences of customer reviews on a website. Ideally, each unit of observation should contain the relevant theoretical concepts. These concepts could be from an a-priory theoretical framework (Yin, 1994) or in an open-emerging fashion as recommended by grounded theory researchers (Corbin and Strauss, 2008). The units of observations can also be supplemented with various metadata elements that can help a researcher to remember the context of data collection and assist in further analysis of the theoretical objects.

After these issues are addressed, the researcher needs to develop a script (or program) for automatically crawling and downloading the needed data. While there are many "ready-made" tools for web scraping (e.g. import.io or Web Scraper plug-in for Google Chrome Browser), these tools do not always work. Again, this has to do with the velocity, variety, and veracity of Web data as highlighted previously. Therefore, a researcher should consider learning how to use a programming language for web scraping.

Two popular programming languages for developing web scraping applications are Python and R. Their popularity rests on two reasons. First, both languages already contain numerous libraries for web scraping (e.g. BeautifulSoup in Python or "rvest" package in R). Second, both languages contain libraries that can be used for various forms of data analysis once the data is scraped. However, R has one advantage over Python: a vast and constantly growing collection of user-produced data analysis packages that can be easily downloaded and used at no charge. Java is also often used sometimes for developing "enterprise-grade" scraping and mining applications.

In contrast to using Java, only a basic knowledge of programming is needed to understand how to use R or Python for web scraping. The syntax of R and Python is fairly intuitive. Moreover, the code for web scraping using these languages is often written in a script-like, procedural fashion. Thus, it is not necessary to have an advanced understanding of the object-oriented paradigm for creating and customizing such scripts.

Once a script is developed, it can be used for automatic collection of data from the Web. Crawling a large repository of Web data may take hours or even days (depending on the

size of the data, one's network bandwidth, computer processing capacity, and numerous other technical factors). A web scraping script left running on a machine overnight or over the weekend can be halted sometime into the process for a variety of technical reasons, resulting only a small fraction of the needed data being collected. Thus, some human supervision of the script's execution is often necessary, at least during the debugging stage.

Various issues may arise during a web scraping session. These may include a simple programmatic error (e.g. a loop used to go through all the URLs goes "out of bounds"). It can also be a network issue (e.g., a website is unresponsive or a local firewall is blocking the script). Or a website administrator can block the IP address from which HTTP requests are coming to prevent the website from being overloaded. In any case, it is important to "catch" and analyze these errors for the purposes of fine-tuning the script and making it more reliable. Analyzing these errors often requires a basic understanding of networking protocols (e.g. TCP/IP, HTTP).

To summarize, this phase requires achieving the following objectives:

- Developing a good understanding of how the units of observation that a website contains can be used to represent various theoretical concepts or units of analysis.
- Determining how can the data be automatically scraped in order to minimize the time involved in data collection.
- Selecting a programming language or a tool that can be used for scraping the needed data.
- Writing code or fine-tuning an existing script for the project.
- Running the developed script in a supervised fashion. This is done to make sure the actual web scraping tasks proceeds as planned. This requires fixing bugs if interruptions happen due to errors or readjusting the tool used for scraping if the script does not collect the needed data.

Phase 4: Tidying Raw Data

The first step in tidying raw data is finalizing the list of data and metadata elements to be saved and creating meaningful linguistic labels for these elements. These data elements need to be saved in a way that takes into account the relationship between the units of observation and the units of analysis. Once these relationships are clearly defined by the researchers, retrieved qualitative data needs to be cleaned and organized in a way that clearly delineates units of observation and allows for further extraction and analysis of units of analysis. This approach thus makes the data more easily accessible and analyzable by both humans and machines alike.

When we refer to "clean data" we mean data that is free from data elements that are not needed in analysis. We call these data elements "data impurities". Among other things, these "impurities" may include special characters such as HTML tags or text formatting symbols like new line, or extra white spaces. These impurities can be programmatically removed in R using the numerous functions that are a part of various NLP packages. But if a quality website is selected initially, then the chance and magnitude of "data

impurities" reduces considerably. For example, if all posts to the website are written by qualified people or subjected to moderation, then data of higher quality will be produced.

Since large qualitative datasets often require computer tools for analysis, it is advisable to use a popular file formats such as text, Excel, CSV, JSON, etc.to organize such datasets as tables. Please note however that some of these formats (e.g. text or Excel) do not exclude manual analysis of data.

Regardless of the format used for data organization, the data still needs to be "tidy". Drawing on Wickham (2014), we define tidy qualitative data as follows:

1. The data is clean
2. Data is organized according to the unit of observation. For example, one can use a table to store information about many units of observation. Alternatively, a separate file can be used for each observation (if units of observations are large).
3. Each unit of observation can be further subdivided into theoretical concepts or units of analysis (either grounded in data or from an a priori theoretical framework). For example, a text file storing information about one observation can be subdivided into separate sections related to different theoretical concepts. If the observations are in a table, then columns can be used to subdivide each observation into theoretical concepts related to a research question. Metadata elements can also be added as columns in the table or as headers (first lines) in each file.
4. Each observation should be clearly demarcated from other observations. For example, if data is tabulated than rows can be used to separate each observation. If observations are used as separate files, then the file structure provides the needed separation.
5. The data and metadata elements are labeled using intuitive linguistic labels.

Again, programmatic tools such as R can be used not only to tabulate qualitative data and store it in a machine-readable file format, but also to do preprocessing that eliminates certain data quality issues and makes the data suitable for different forms of analysis. For example, R can be used to eliminate excessive white spaces, formatting characters, "stop words", etc. Programmatic tools can also be used to read a particular qualitative dataset and convert it into a format useable by various text mining tools (e.g. see Feinerer et al., 2015).

To sum up, this phase requires achieving the following objectives:

- Reflecting on how the units of observation relate to the theoretical concepts or the units of analysis needed for answering the research question.
- Deciding on the structure for storing data that is aligned with the tidy data principles and theoretical considerations.
- Selecting a file format for data storage, depending on how the data will be analyzed in the future.
- Creating intuitive linguistic labels for various data and metadata elements
- Saving data using the structure and file format selected earlier programmatically, in order to save time and eliminate errors.

- If necessary, data needs to be cleaned from impurities (e.g. white spaces or unnecessary special characters can be removed) prior to saving.

R Environment

Virtually all of the data manipulations mentioned as a part of the description of the web scraping framework can be implemented using R. R is a programming language developed specifically for data analysis. The original intent of R was to implement various statistical analysis techniques. But in recent years R has grown substantially to accommodate qualitative data analysis as well (in addition to the countless other extensions of the language that have been developed). R is increasingly becoming a de facto standard platform for data science projects (Schutt and O'Neil, 2014). It is also gaining popularity in academic research. Although R language entered the academic mainstream only a few years ago, Google Scholar already returns 33,100 results for the search phrase "R language".

There are several reasons to explain the growing popularity of R in academic research and industry data science projects. First, unlike other popular data analysis tools, R is open-source and free to use. Second, R was not created with a particular industry or application area in mind. Unlike SPSS (designed primarily for social sciences) or SAS (aimed at large for-profit enterprises), R is used and developed by researchers in virtually all industries and academic fields (Lander, 2014). Yet there are many domain specific packages available for R. For example, package "ade4" is used for analyzing ecological data. Package "syuzhet" is used for analyzing plots in literary works. Finally, there are more than ten thousand packages available for free from the Comprehensive R Archive Network (CRAN) repository (a "peer-reviewed" online collection of various analysis tools or "add-ons" developed by the R user community). Thus, data collected with R can also be analyzed in R using a variety of packages.

Among other things, R packages include:

- pre-programmed functions for data manipulations and cleaning (e.g. "dplyer", "tidyext", "tidyxl");
- virtually all known forms of statistical analysis of quantitative data (anything from basic descriptive statistics to advanced techniques such as regression analysis and structural equation modeling);
- dozens of implementations of various machine learning algorithms (e.g. "fugeR", "mlearning", "mtools", etc.);
- advanced tools for organizing and visualizing quantitative and qualitative data (e.g. "gglot2");
- various types of text mining and natural language processing algorithms (e.g. "LSA", "tm", "syuzhet", "SnowballC" etc.)

With R containing functions to support virtually all known forms of data analysis, R can be used to support and unambiguously describe virtually every step of a research project. Failure to unambiguously capture and communicate all the steps of a research project can reduce the ability of other researchers to reproduce and, thus, validate the

results of the study (Dennis and Valacich, 2014). Hence, the scientific merit of such research may be diminished (Olbrich et al., 2015)

R can enhance reproducibility of research. For example, R includes many functions for downloading and managing various data files and data elements. Using these functions, a researcher can not only streamline data analysis, but document the entire research protocol from the time a dataset is loaded or obtained from the Web, to the time when a research report is produced. A built-in mark-up language called Markdown is specifically designed for documenting the entire research protocol in R and disseminating the results of research. Markdown implements the "literate programming" paradigm proposed by Donald Knuth (1984). Under "literate programming", researcher's comments, analysis code, and analysis outputs are meshed into one single file that can be later automatically converted into a Word document, PDF file, or even a deck of PowerPoints. R also includes packages for generating professional-looking tables for descriptive statistics or regression output in Latex or HTML formats (e.g. the "stargazer" package). Even the entire "environment" (which includes memory areas filled with variables and their values) can be saved and distributed in a project file. These R codes can be distributed together with the datasets used so that any researcher can run those files on his or her computer and validate research findings. All these tools can not only speed up and automate production of research, but also make one's research reproducible (Peng, 2011).

Before using R, one has to download and install an R distributive. R can be downloaded from the CRAN website (https://cran.r-project.org/). On this website on can find download links for various operating systems. Currently, R is available for Windows, OS X (Mac), and Linux. We recommend that the latest version of R distributive is downloaded. Moreover some recently developed data analysis packages may not be compatible with older versions of R. Once an appropriate R distributive is installed, one can test whether the installation is working by launching R.

Once downloaded and installed, R language can be used in a console fashion (see Figure 2). Figure 2 contains what could be an equivalent of "Hello, World!" in programming. Two variables (`x` and `y`) are assigned values (both are equal to 2). Then the system is asked to add the two numbers. The system automatically displays the output (which is 4).

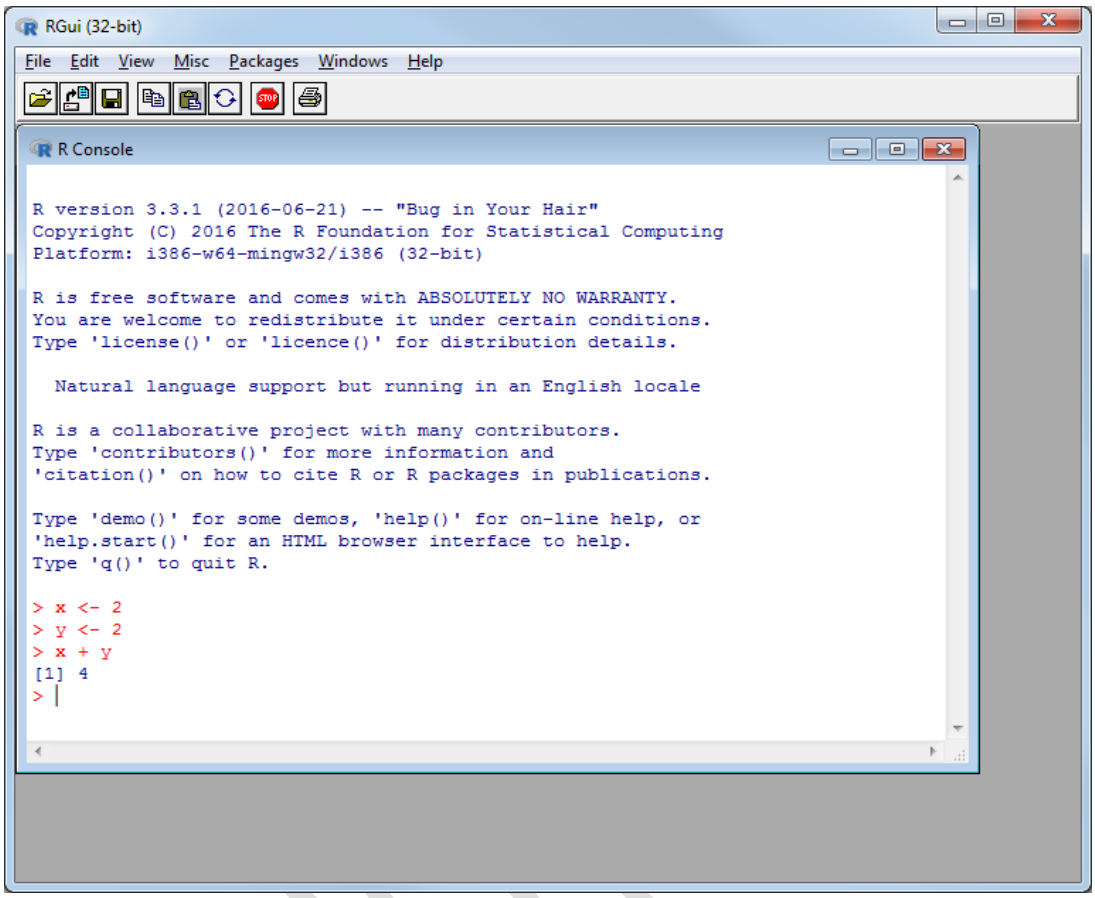


Figure 2. R Console

Yet most people prefer to install an Integrated Development Environment (IDE) on the top of the R base installation. While many IDEs have been developed for R, RStudio seems to be the most popular IDE for R. RStudio is available from https://www.rstudio.com. R Studio includes a free, desktop version together with several enterprise-grade versions (e.g. RStudio Commercial Desktop and RStudio Server Pro). Enterprise versions require a license fee.

The free version is fully functional and suitable for the needs of most academic researchers. RStudio makes coding in R easier and facilitates development of advanced R scripts. RStudio includes a code editor together with advanced debugging and visualization tools. RStudio has many other useful and free add-ons. For example, the so-called Shiny tool contains functionalities for building Web-based data products using R. No knowledge of HTML, CSS, or JavaScript is required to build these Web products using Shiny. RStudio also contains a graphical interface for the “knitr” package (available from CRAN). This package can be used in conjunction with R Markdown language to generate data analysis reports in HTML, PDF, Word, and even PowerPoint formats. These reports weave together R code, user comments, and output from R in one single document that can be later shared with the intended audience. These documents can subsequently be updated and generated easily.

Another reason for the popularity of RStudio is its highly customizable Graphical User Interface (GUI) that allows users to have different layouts of RStudio's GUI to suit their own unique needs and working style. A typical layout will look similar to the one depicted in Figure 3.

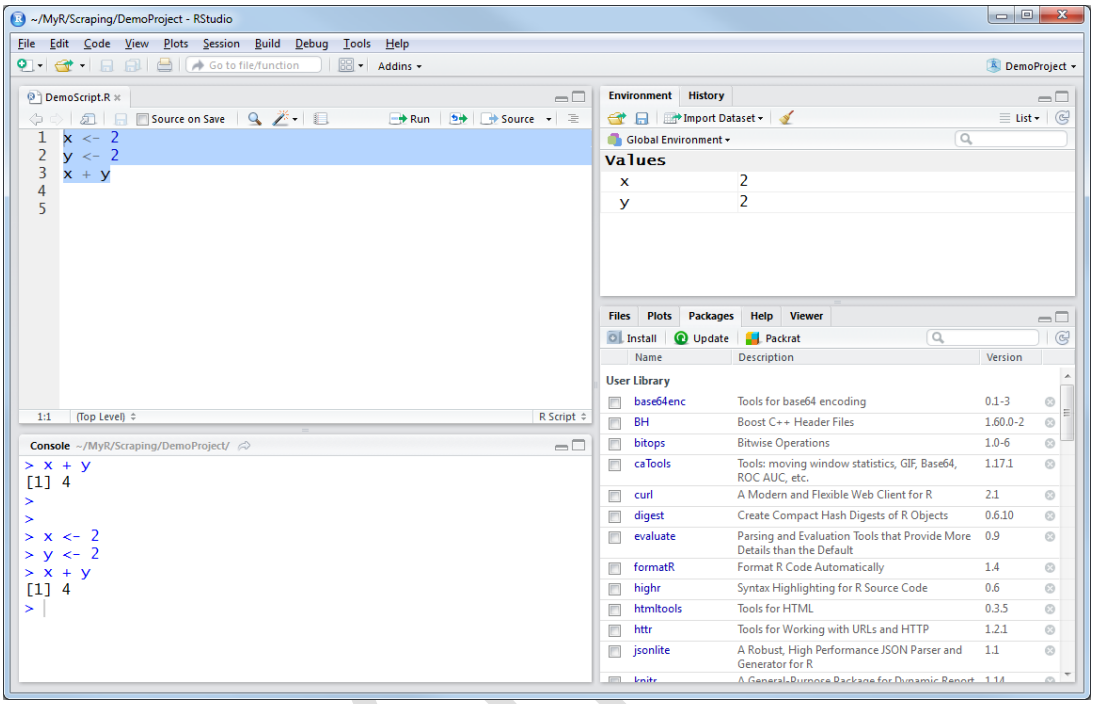


Figure 3. RStudio IDE

The Case of Dice.com

In this section, we illustrate application of the proposed web scraping framework to Dice (www.dice.com) - an employment website for IT and Engineering professionals (DHI Group, 2017). R language is chosen as a technology platform for data collection due to the growing popularity of the language in research and industry data science projects as we as availability of numerous packages for data collection, processing, and analysis. This includes R packages developed specifically for data collection from the Web.

This illustrative qualitative research project aims to answer the following research question:

> *RQ: What are the most in-demand skills or requirements for the role of a Systems Analyst?*

This research question is important from both practical and theoretical standpoints. From a practical standpoint, an answer to this research question can be used to fine-tune the curricula of MIS programs. Many MIS programs position themselves as programs preparing Systems or Business Analysts. Thus, an answer to this question can lead to a theoretical yet practically relevant contribution in the form of a competency framework for

the role of a Systems Analyst within an organization. This framework can be of practical value to educational institutions, recruiters, and organizations employing Systems Analysts.

This question can be potentially answered using the data available on Dice. Dice is one of the leading, specialized recruitment websites (Lee, 2007). Historically, the website has specialized in job search and recruitment solutions for IT and Engineering workforce (DHI Group, 2017). At a given moment, the website contains close to 80,000 active job listings that are publicly accessible, and has a collection of 2.1 million resumes (not in public access). The website also reports more than 2 million unique visitors to the website every month.

At its most basic level, Dice.com can be viewed as an online bulletin board, where potential job seekers can post their resumes and recruiters representing various companies can post their job ads. Job seekers can then apply to posted jobs, while recruiters can browse through the resumes of IT professionals using the website. Dice.com offers a number of other research and data solutions related to recruitment in IT for companies in Finance, Energy, Healthcare, and Hospitality (e.g. security clearance of job applicants). The company also publishes various types of research in relation to IT labor and recruitment.

The main outcomes of the web scraping process framework described in the previous section that is applied to Dice.com is provided in Table 2 below. Subsequent sections expand on the information provided in the table.

Table 2. Web Scraping Dice: Phases and Main Outcomes
Phase 1: Selecting a Website • "Typical case" theoretical justification is used (Yin's, 1994). Dice is a specialized website contains job listings that specifically reflect the current skills requirements for various IT rules. Moreover, this website was used by other IS studies attempting to answer similar research questions (Davis et al., 2003; Sandvig, 2007). • Collectively, this data can be potentially used to answer the research question. • Overall, based on the examination of the website and prior literature (e.g. Lee, 2007), the website appears to be a reliable source of rich data on job descriptions for various IT roles. • The website's "Terms of Use Agreement" allows scraping the website using the API provided by the company.
Phase 2: Analyzing a Website • The job listings are posted by companies and IT recruiters. The data is further audited and moderated by Dice staff, as per the "Terms of Use Agreement". • Information about job listings is transmitted via API using the following data interchange formats: HTML, XML, and JSON. JSON was chosen, since it is "light" (has less "tagging" than HTML and XML) and R has several packages for working with JSON. • API transmits mostly metadata about job listings. Job descriptions need to be retrieved by following the URLs supplied with job listings data via the API. Examination of web pages containing job descriptions reveals that HTML and CSS is used to format these complete job descriptions. The CSS element that contains the actual job description has id "jobdescSec".
Phase 3: Retrieving Raw Data • The units of analysis (job listings) contain the relevant theoretical concepts or units of analysis that can be programmatically retrieved from the units of observation. • R is chosen for developing a script for working with the API and scraping job descriptions. One reason R is chosen that the language has many "packages" for simulating web sessions and working with various data formats (HTML, CSS, XML, etc.). The second reason is that R contains numerous packages for further processing and analysis of scraped data. • RStudio Integrated Development Environment (IDE) is used to develop a script for scraping data. • The script is tested many times for small, medium, and large-size datasets to eliminate most of the errors. • A moderate-size (1104 job listings) and a large size (44,889 job listings) datasets containing jobs listings related to the "Systems Analyst" roles are downloaded using R.
Phase 4: Tidying Raw Data • A unit of observation (a job listing) can be further subdivided into the following concepts and metadata elements: job URL, job title, company that posted the job, job location, the date when job was posted, and job description. • To meet the tidy data principles the data is saved in a table, where each column represents concepts and metadata elements and each raw represents individual observations. • Excel file format is chosen to make the data analyzable by automated tools (e.g. R) and also manually (most people are used to inspecting data using Excel). • The data was programmatically saved in an Excel file using the tidy data principles

Understanding the technical details of the framework's application to Dice requires a basic understanding of browser functionality (e.g. how to view the underlying code of a webpage) and various Web and data transmission technologies (e.g. HTML, CSS, JSON, HTTP). One does not have to be familiar with R to understand the code

presented in this section. R syntax is fairly intuitive, so only a general understanding of programming is required. Nevertheless, we recommend referring to the resources listed in Appendix C if the reader wishes to delve deeper into some of the technologies and packages used in this section.

Phase 1: Selecting a Website

Selecting Dice as a potential source of data for answering our research question involves several steps. First, we provide initial reflections on the richness, quality, and overall suitability of data available on the website based on the initial inspection of the website. Second, selection of the website is justified theoretically and also using prior IS literature on IT workforce trends utilizing this website. Finally, we determine that the website's "Terms of Use" policy allows for automatic crawling and scraping of data using the website's API. The main outcome of this step is the decision to proceed with the website as a source data potentially capable of answering our research question.

Reflecting on Data Richness and Quality

The website contains a rich repository of qualitative data in relation to IT jobs and recruitment. Initial inspection of Dice's data suggests that its job listings are quite rich (i.e., that most job descriptions are quite long and detailed) and of high data quality (i.e., no missing or poorly written job descriptions were detected upon initial examination of job listings). Thus, we determine that the website is potentially suitable for answering the research question.

Justification of Website Selection

The data from Dice that is publically available includes active jobs ads posted by companies and recruiters to the website. These jobs reflect the "typical requirements" that companies and recruiters formulate in relation to various IT roles. Thus, Yin's (1994) "typical case" theoretical justification can be used. Overall, the data that Dice contains is very much reflective with the main objective of the study: uncovering skills or requirements for the Systems Analyst role as formulated by the "job market".

Moreover, Dice has been used in several ways by IS studies related to IT workforce trends. Table 3 below provides an illustration of some of these academic and practical uses. One use of the website has been specifically to answer the question about the in-demand skills for a particular IT role (e.g. Davis et al., 2003 and Sandvig, 2007).

Table 3. IS Research on IT Workforce Trends Using Dice.com	
Study	**Use**
Lee (2007)	Dice.com was suggested as a source for niche recruitment for IT jobs as a part of a holistic, e-recruitment system.
Granger et al. (2007)	Dice.com was used to illustrate the number of IT jobs available in the U.S. as a part of a broader discussion on IS enrollment trends
Sandvig (2007)	Job listings for Seattle, WA area were analyzed to discover the most in-demand skills and technologies related to server-side programming. The findings were used to evaluate the relevance of the E-Business curriculum at Western Washington University. Data was analyzed manually.
Patterson (2006)	Dice.com was used by the President of ACM, David Patterson, to show the growth in IT jobs in the US. This information was used in shaping the strategies of ACM in terms of boosting enrollment in IT programs across the country.
Dattero and Galup (2004)	Dice.com was used to disseminate a survey on programming language used by software developers. The goal of the study was to detect whether the use of programming languages differ by gender.
Davis et al. (2003)	Job listings from Dice were used to identify the current industry needs for E-Business skills and match these skills with the skills taught by E-Business programs in universities. Dice.com was used in combination with a number of other websites devoted to IT jobs. A total of 1000 job listings were retrieved and analyzed by two researchers.

Reflecting on Legality of Crawling and Scraping Dice.com

The “Terms of Use Agreement” explicitly allows data access using the API created by the company. This API is available to developers at no charge. The API from Dice can be used to programmatically search through jobs ads and, if necessary, download this information. Thus, the company encourages automatic crawling and scraping by the user community.

Main Outcomes

The main outcome of this step is the overall conclusion that Dice contains rich, quality data on Systems Analyst job descriptions. This data can potentially answer the research question of the study. Thus, the site’s selection can be justified theoretically. The website “Terms of Use Agreement” allows web scraping using the API. Thus, it was decided to proceed with this website as a source of data and subject it to further analysis.

Phase 2: Analyzing a Website

Further analysis of Dice involves several steps. First, we reflect on the social structures used to create data, and how these social structures impact data quality. Second, we explore the API manual supplied by the website for programmatic retrieval of the website’s data. Third, we analyze the structure of webpages containing complete job descriptions to determine which data element contains qualitative job descriptions. The main outcome of this step is developing an understanding of how the data needed to answer the research question can be automatically retrieved.

Reflecting on Data Quality and Social Structures Further

Further, detailed inspection of sample job listings on Dice reveals that the data is of high quality. This is not surprising. The main business of Dice involves various data products for the IT recruitment field. Thus, the data that the website contains is professionally managed. Moreover, job listings are created by professionals, such as managers and recruiters. This fact contributes to data quality. In general, Dice is considered to be a quality source of data among IS researchers (Davis et al. 2003).

Exploring the API

A technical analysis of Dice starts with studying the API manual provided by the website (see Appendix A). From this manual, one can see that the website allows for one domain to be used by the public for accessing job data: "service.dice.com" The data therein can be obtained using the API in three formats: HTML, XML, and JSON. The website's database of job listings can be queried by supplying the URL devoted to a certain interchange format with parameters (a complete list of these parameters is also provided in Appendix A). Among other things, these parameters allow to narrow down the search or pre-format the resulting output from the Web server. For example, one can use the following URL to search for systems analyst jobs in New York City Area (see Figure 4) using the API:

```
http://service.dice.com/api/rest/jobsearch/v1/simple.json?text=systems
%20analyst&city=New+York,%20NY
```

Figure 4. Using Dice API to Retrieve Systems Analyst Jobs in New York City

Since the query uses the JSON URL for sending the query, the output is returned by the server as a webpage in JSON format (viewable inside a browser). Please note that all jobs containing the two words ("systems" and "analyst") are retrieved (even if these two words do not occur together as a phrase). If one wants to search for the exact phrase ("systems analyst"), the two words need to be presented in quotes to show that it is a phrase. Please also note that one should use the symbol `%22` instead of quotes to ensure proper URL encoding. Thus, to search for all "systems analyst" jobs listed on Dice, one should use the following URL (Figure 5):

```
http://service.dice.com/api/rest/jobsearch/v1/simple.json?text=%22syst
ems%20analyst%22
```

Figure 5. Searching Systems Analyst Jobs in New York City Using a Phrase

The output from this query is provided in Appendix B. The output contains useful metadata for the entire dataset. For example, when the query was sent, Dice contained 1105 jobs with the phrase "systems analysts". The information is provided with 50 jobs being listed per page. To open the subsequent page, one can follow the "nextURL" (see Figure 6) listed at the top of the JSON output (see Appendix B):

```
"nextUrl":"/api/rest/jobsearch/v1/simple.json?areacode\u003d\u0026coun
try\u003d\u`0026state\u003d\u0026skill\u003d\u0026city\u003d\u0026text
\u003d%22systems+analyst%22\u0026ip\u003d\u0026diceid\u003d\u0026page\
u003d2"
```

Figure 6. Searching Systems Analyst Jobs in New York City Using a Phrase

As one can see from the output in JSON format (see Appendix B), the API does not allow for actual job descriptions to be retrieved. The API provides only for URLs pointing to actual job descriptions using the following attribute-value format (see Figure 7):

```
detailUrl":"http://www.dice.com/job/result/RTX16c2f4/it-systems-
analyst?src\u003d19"
```

Figure 7. Searching Systems Analyst Jobs in New York City Using a Phrase

Thus, for every job retrieved a crawler needs to load the corresponding URL with details and retrieve job descriptions from there.

Analyzing the Webpages Containing Job Descriptions

One can open any of these URLs in a browser (e.g. Google Chrome) for further analysis. In Google Chrome, the “Inspect” feature can be used analyze the underlying code of a job description page in order to determine which elements contain the needed job description. Upon examination one can see that the `id` of the `div` element containing the job description is `"jobdescSec"` (see Figure 8):

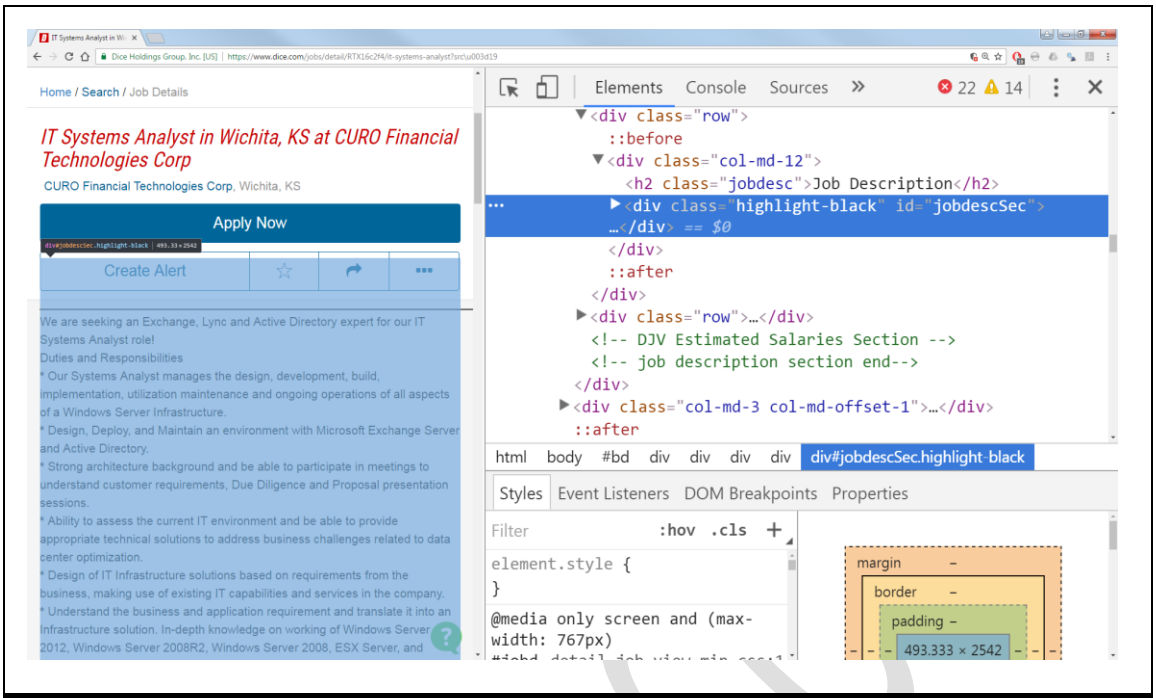


Figure 8. Analyzing Code in Google Chrome

One can also use the SelectorGadget add-on for Google Chrome to find the needed page element in a “point and click” fashion (see Figure 9). This is easier for those not very familiar with HTML, CSS, or XML, depending on which mark-up language is used to structure a website. It is also often difficult to acquire the right element without also “grabbing” neighboring ones using the SelectorGadget. Thus, analyzing the underlying code manually is probably a more reliable way of determining which mark-up element contains the needed data.

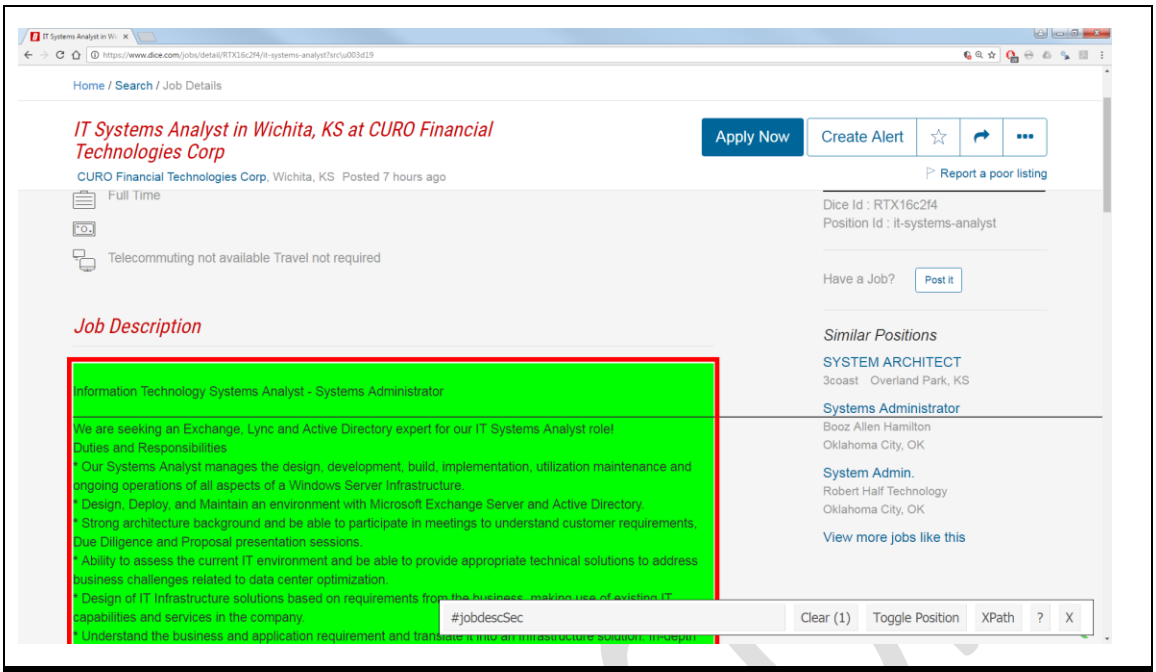


Figure 9. Finding CSS Element Using the SelectorGadget Add-on in Chrome

Main Outcomes

By analyzing the API supplied by Dice, we now know what data are available through the API and how this data is organized and managed. Unfortunately, job descriptions are not available directly via the API output. However, the API returns URLs for accessing detailed job descriptions. The Inspect feature of Google Chrome and the Selector Gadget "add-on" for Chrome are used to determine which HTML element contains the job descriptions. All this information is needed for developing an R script that automates data collection from Dice.

Phase 3: Retrieving Raw Data

Retrieving data on job listings related to Systems Analyst consists of three main steps. Step one involves finalizing the links between units of observation and units of analysis. Step two involves developing and debugging an R script for crawling the website and downloading the data related to Systems Analyst job listings in a way that is consistent with the study's units of analysis (or theoretical concepts). Before the script was used for retrieving data, it was "pre-tested" several times on smaller tasks (e.g. niche jobs within certain geographical areas known to contain very few job listings). Step three involves running the script with some degree of human supervision to retrieve the data. The main outcome of this phase is a data frame (a popular data structure in R similar to a table in Excel) filled with data and meta-data related to Systems Analyst job listings.

Linking Units of Observation and Units of Analysis

The unit of observation obtained from Dice is a job listing. A job listing can be further subdivided into the following theoretical concepts or units of analysis: the name of the position advertised, the name of the company advertising a position, the geographical

location for which the position is advertised, the description of the position, and job requirements. Additional theoretical concepts can be potentially derived from job descriptions using the text mining tools. All these units of analysis can also be supplemented with metadata, including the date when the job was posted to the website and the URL for each job posting. These units of analysis can potentially be combined to provide a theoretical answer to the research question formulated earlier.

Developing and Debugging an R Script

In this section, an R script is presented that downloads job descriptions and other data related to each job listing (or main unit of observation) for jobs that contain the phrase "Systems Analyst". The R script used for downloading data is deliberately short. It is applied to a moderately-sized data set of 1,104 observations first and then to a dataset of 44,889. Only the application to the moderately sized dataset is described in detail to make the illustration simpler. Moreover, some useful functionality (e.g., processing of error messages, automatic formation of URLs based on search parameters, etc.) is excluded from the script to reduce the total number of lines of code to the minimum necessary. Moreover, as shown in Figure 10, the algorithm behind the R script is rather linear and corresponds to the way a human downloads and saves data from the Web manually. All this makes the script easier both to understand and customize for a different research project.

The R code used specifically for data collection is presented in Figure 11, while the code used for tidying and saving data in as an Excel file is presented in Figure 13. We feel it is important to include the code in its entirety to provide a holistic, socio-technical description of the web scraping process. The code contains detailed comments explaining each section of the code. Comments are highlighted in green and start with a hashtag symbol (#).

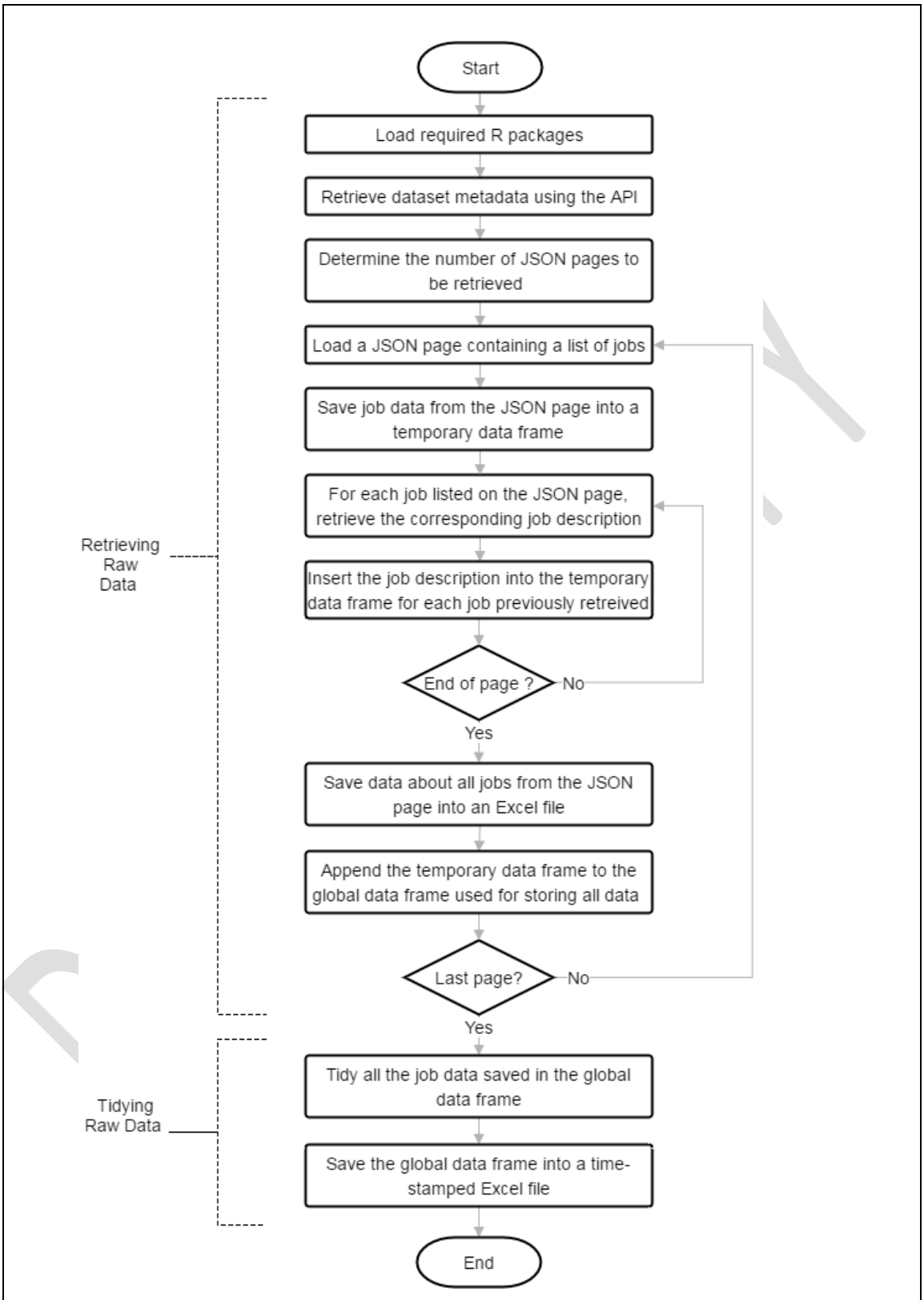


Figure 10. Web Scraping Algorithm

Note that the main goal of the script provided in Figure 11 is to process the units of observation (job listings) in a way where all relevant units of analysis (e.g. job title, job description, etc.) and relevant metadata elements (e.g. date and time when a job listing was posted to the website, the URL from which the job description was retrieved, the date when data was scraped, etc.) can be accessed and captured. Thus, a good understanding of how units of observation and units of analysis are related is already required at this stage.

```
#Loading the required packages

require(jsonlite)
require(xlsx)
require(rvest)
require(curl)
require(plyr)

#Set the directory where data will be saved
setwd("C:/Users/cob.user/Documents/Research/JAIS/R/Data")


#Forming an initial URL. This URL is used to retrieve scraping
parameters, like the number of job descriptions to be retrieved
initial_url <-
"http://service.dice.com/api/rest/jobsearch/v1/simple.json?text=%22sy
stems+analyst%22"

#Loading JSON data from the URL. Note that jdata is a list of 6
elements
jdata <- fromJSON(curl(initial_url, handle =
curl::new_handle("useragent" = "RStudio")))

#Element 5 from the jdata list is a data frame containing information
about jobs. So we save it in a separate data frame
job_data <- data.frame(jdata[5])

#We also save the total number of job descriptions to be retrieved.
This value is element 1 in the jdata list
job_count <- as.numeric(jdata[1])

#Number of total JSON pages is calculated by dividing the number of
total jobs by 50, which is the number of jobs listed on each JSON
page by default
page_count <- ceiling(job_count/50)

#Creating a data frame where all job descriptions will be stored
job_table <- data.frame()

#Creating a temporary data frame where job descriptions from each
page will be temporarily stored until they are appended to the
job_table data frame
job_temp_table <- data.frame()

#This outer for loop will load the pages with job listings in JSON
format from the first page to the last page as calculated earlier
for (i in 1:page_count){
```

```
#A page URL is formed. Page number is supplied as a parameter
page_url <-
paste("http://service.dice.com/api/rest/jobsearch/v1/simple.json?text
=%22systems+analyst%22&page=",i,sep = "")

#Reading JSON data into a list called jdata
jdata <- fromJSON(page_url)

#If it's not first or last page (where the list returned contains 6
elements), save the 6th element of the data frame into the temporary
data frame job_temp_table
if ((i != 1) && (i != page_count)) {

    job_temp_table <- data.frame(jdata[6])

#The first and the last page are saved into a list of only 5
elements, since previous URL element is absent on the first page and
next URL element is absent on the last page. Thus, the 5th element of
the jdata list contains the needed job description data
} else {
  page_url <-
paste("http://service.dice.com/api/rest/jobsearch/v1/simple.json?text
=%22systems+analyst%22&page=",i,sep = "")
  jdata <- fromJSON(page_url)
  job_temp_table <- data.frame(jdata[5])
}
 #Inserting a new column named Description into the temporary data
 frame. This is where job descriptions will be added for every job
 job_temp_table["Description"] <- ""

 #Calculating the total number of jobs to be loaded for each page
 using the firstDocument and lastDocument elements in the list. The
 number of jobs to be downloaded will be the 50 for all job poges
 except the last one.
 jobs_to_load <- as.integer(jdata[3]) - as.integer(jdata[2]) + 1

 #This is a part of a simple status bar. It will show the user which
 page is being scraped.
 cat(paste("Scraping ",jobs_to_load, " jobs from page ",i," out of
 ",page_count,": ", sep = ""))

  #This inner loop uses job URLs to download job descriptions for
  each of the jobs listed on a JSON page and saves these descriptions
  in the "Description" column of the temporary data frame
  job_temp_table.
  for (job_id in 1:jobs_to_load){

      #The try({})block is used for catching errors. If there is an
      error reading or scraping a particular URL, then move to next
      URL.
      try({
      #The lines below are used to load URLs containing job
      descriptions, read HTML codes of pages loaded, and save data
      contained in a CSS element containing a detailed job
      description into a data frame.
      job_html <-
```

```
      read_html(curl(job_temp_table$resultItemList.detailUrl[job_id],
      handle = curl::new_handle("useragent" = "RStudio")))
      job_node <- html_node(job_html,"#jobdescSec")
      job_text <- html_text(job_node)
      job_temp_table$Description[job_id] <- job_text

      #This is a part of a simple status bar that shows progress in
      scraping jobs from each page
      cat("*")

      })
  }

  #To prevent data loss, each page is saved into a separate Excel
  file
  page_file_name <- paste("page",i,".xlsx", sep = "")
  write.xlsx(job_temp_table,file = page_file_name)

  #Job data from the page is appended to the global data frame where
  all job descriptions are stored
  job_table <- rbind.data.frame(job_table, job_temp_table)

  #Output screen is cleared to avoid clutter
  cat("\014")
}
```

Figure 11. R Code for Scraping Job listings from Dice.com

Retrieving Data Using the R Script

The RStudio console window contains a simple progress bar that shows the crawler is on page 5 out of a total of 23 pages as shown in Figure 12:

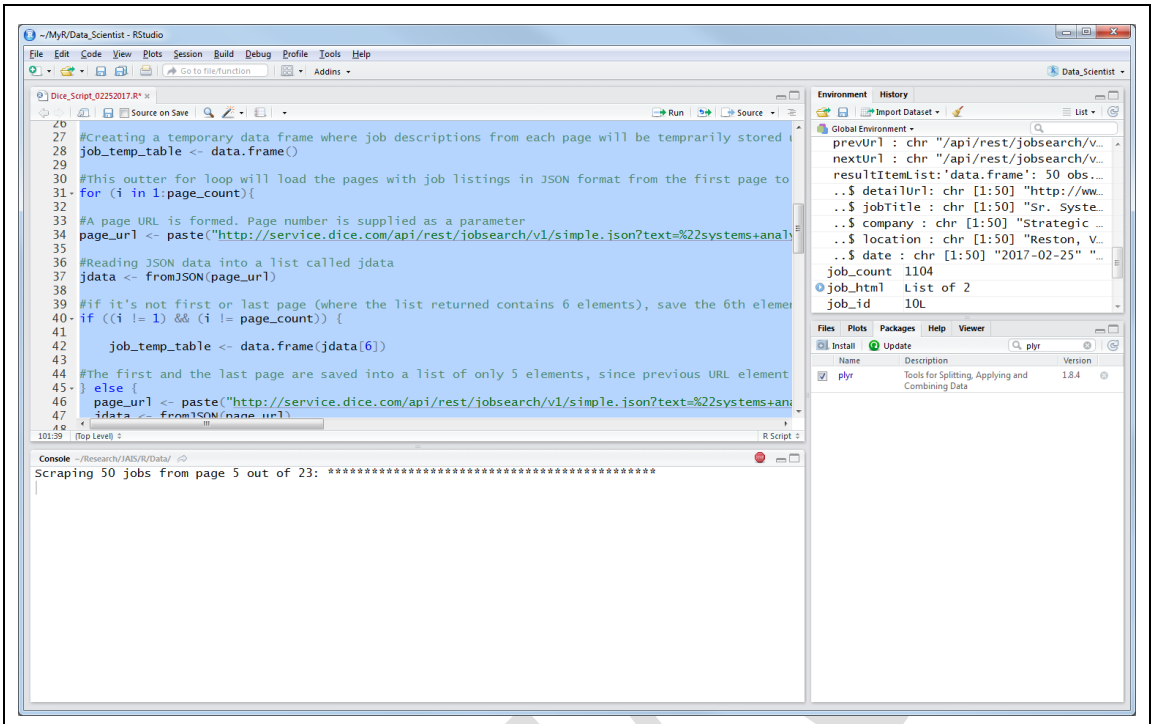


Figure 12. R Console During the Web Scraping Session

It took approximately 19 minutes to download data related to 1104 jobs on an office computer with the following configuration:

- Processor: Intel® Core™ i5-4590 CPU @ 3.30 GHz
- RAM: 16GB
- Upload/Download Speed: ~100 Mbps

Given that one spends one minute processing each job listing manually (that involves loading a particular job URL, copying the relevant data from the Web page, saving this data into an Excel sheet together with corresponding metadata), then it would take approximately 19 hours of non-stop work by one researcher to obtain all this data. This is a long time for a dataset of a rather modest (by big data standards) size. In reality, this data collection exercise is likely to take more time due to the monotonous nature of such work, and the mistakes that one is likely to make performing a repetitive and tedious task for many hours (or days).

It took approximately 12 hours to download data for 44,889 job listings that contained words “systems” and “analyst” (not necessarily as a single phrase) using the computer with the configuration provided above. This roughly corresponds to one month of “around the clock” work by a single researcher. Again, given repressiveness of the work and other commitments that a researcher may have, it would potentially take close to a year to collect this data.

Note that these 12 hours do not include a one hour interruption caused by the computer running out of memory (which happened 11 hours into the script execution). The computer was restarted to continue the scrapping task. No data was lost since the script

saves data from every page into a separate sheet. The script was restarted to load the remaining pages. Separate Excel sheets can be quickly assembled into one file using R functions for working with Excel files (contained within the “xlsx” package).

Main Outcomes

Also note that all the data is now stored in a `job_table` data frame. A data frame is probably the most common type of data structure in R. A data frame corresponds roughly to a\n Excel table, where each column represents a variable. Each variable has a name (or column heading) and can be of different data type. This data frame is saved to the R project environment. Thus, this data can be saved and distributed together with the project file and analyzed within the same project. Yet the goal of this demonstration is to save the file in a tidy format in an Excel file so that this data can be analyzed using other software tools (e.g. NVivo) or even manually. Then next section described how the data is “tidied up” further and saved in an Excel file containing some metadata.

Phase 4: Tidying Raw Data

Tidying raw data involves the following steps. First, meaningful linguistic labels for columns in the data frame are created. Second, the data is saved into an Excel file in a tidy format to allow for subsequent analysis of the data using other software tools or even manually. Metadata in the form of a time and date “stamp” is then added to the file name to make it easier for researchers to keep track of various scraping jobs performed. The main outcome of this step is the Excel-file structure produced by R containing data and metadata on Systems Analysts job listings. This data can be further analyzed using various software tools (including R) or even manually (e.g. Sandvi, 2007).

Saving the Data in an Excel File

Once all the data is saved in the `job_table` data frame (see the previous script in Figure 11), we can use R to further process and organize data in an Excel sheet (we call this process “data tidying”). The code in Figure 13 is used to save the content of the data frame into an Excel file. This code in Figure 13 can be appended to the code provided in Figure 11. Here it is separated into a separate Figure as it is related to a different phase under web scraping framework: tidying raw data.

```
#Tidying the previously created data frame by renaming the columns of
the global data frame in a way that will make it easier for a user to
understand data
colnames(job_table) <- c("JobURL", "JobTitle", "Company",
"JobLocation", "JobDate", "JobDescription")

#Generating a string containing current date and time
current_date <- as.character(date())
file_date <- gsub(" ", "_", current_date, fixed = TRUE)
file_date_clean <- gsub(":", "_",file_date, fixed = TRUE)

#Generating a meaningful file name that contains current data and
time
file_name <- paste("Jobs_",file_date_clean,".csv", sep= "")

#Writing all the data scraped and stored in the job_table data frame
(see previous script) into an Excel file.
write.xlsx(job_table,file = file_name)
```

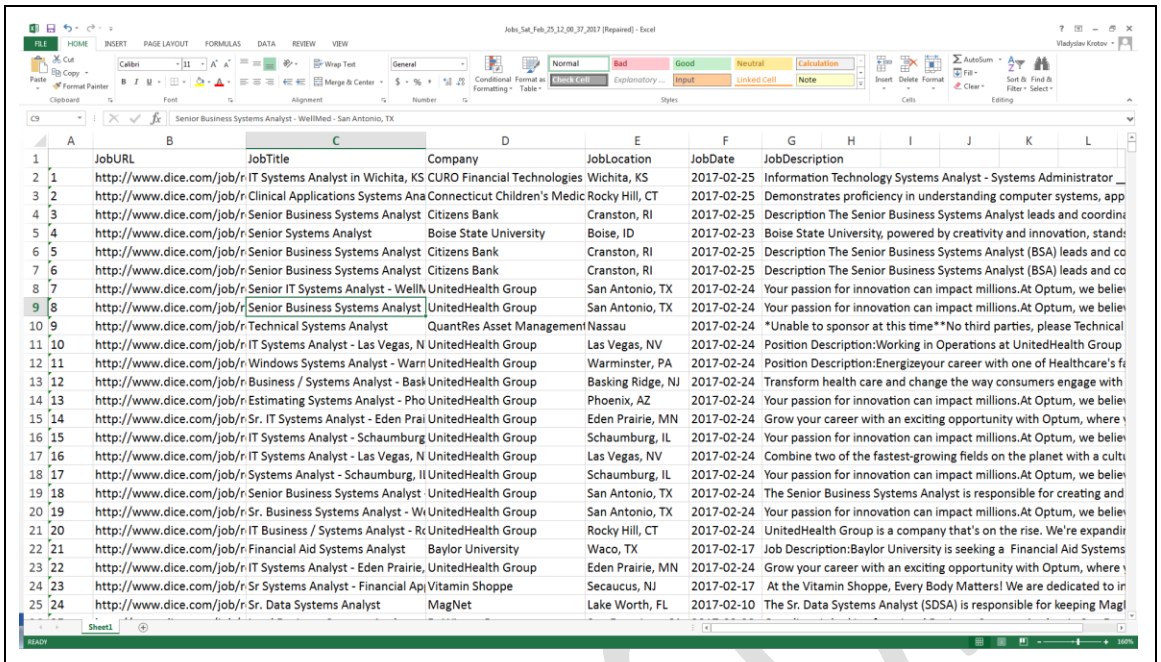
Figure 13. R Code Tidying Data and Saving it in Excel Format

The data is saved as an Excel file due to the universality of this format. Excel is often used for manual data analysis. Most people are quite comfortable viewing and analyzing data in Excel. Moreover, Excel files can be easily imported into virtually any type of software (including R and NVivo). Note that Microsoft Excel is just one of the file formats that can be generated. Virtually any file format and data organization can be produced using R. For example, if necessary, each job description can be saved as a text file.

The file name contains the additional metadata: date and time. This is done so that by simply looking at a file, a researcher can quickly determine when a particular scraping job was performed. Some research projects may require scraping data on multiple dates to come up with a substantially large data sample.

Reflecting on Data "Tidiness"

Note that the data stored in each file is "tidy" (see Figure 14). The file contains information about Systems Analyst jobs listed on Dice. Each row corresponds to a particular job listing, so units of observation are clearly delineated. Each column represents the units of analysis that each unit of observation represents together with metadata about each observation. The specific columns include the URL used to obtain a job description (JobURL), the title of the job listed (JobTitle), the name of the company that poste the job (Company), the date when the job was posted (JobDate), and the full description of the job listings (JobDescription)



	A	JobURL	JobTitle	Company	JobLocation	JobDate	JobDescription
2	1	http://www.dice.com/job/r	IT Systems Analyst in Wichita, KS	CURO Financial Technologies	Wichita, KS	2017-02-25	Information Technology Systems Analyst - Systems Administrator _
3	2	http://www.dice.com/job/r	Clinical Applications Systems Ana	Connecticut Children's Medic	Rocky Hill, CT	2017-02-25	Demonstrates proficiency in understanding computer systems, app
4	3	http://www.dice.com/job/r	Senior Business Systems Analyst	Citizens Bank	Cranston, RI	2017-02-25	Description The Senior Business Systems Analyst leads and coordina
5	4	http://www.dice.com/job/r	Senior Systems Analyst	Boise State University	Boise, ID	2017-02-23	Boise State University, powered by creativity and innovation, stands
6	5	http://www.dice.com/job/r	Senior Business Systems Analyst	Citizens Bank	Cranston, RI	2017-02-25	Description The Senior Business Systems Analyst (BSA) leads and co
7	6	http://www.dice.com/job/r	Senior Business Systems Analyst	Citizens Bank	Cranston, RI	2017-02-25	Description The Senior Business Systems Analyst (BSA) leads and co
8	7	http://www.dice.com/job/r	Senior IT Systems Analyst - Wellh	UnitedHealth Group	San Antonio, TX	2017-02-24	Your passion for innovation can impact millions.At Optum, we belie
9	8	http://www.dice.com/job/r	Senior Business Systems Analyst	UnitedHealth Group	San Antonio, TX	2017-02-24	Your passion for innovation can impact millions.At Optum, we belie
10	9	http://www.dice.com/job/r	Technical Systems Analyst	QuantRes Asset Management	Nassau	2017-02-24	*Unable to sponsor at this time**No third parties, please Technical
11	10	http://www.dice.com/job/r	IT Systems Analyst - Las Vegas, N	UnitedHealth Group	Las Vegas, NV	2017-02-24	Position Description:Working in Operations at UnitedHealth Group
12	11	http://www.dice.com/job/r	Windows Systems Analyst - Warr	UnitedHealth Group	Warminster, PA	2017-02-24	Position Description:Energizeyour career with one of Healthcare's fa
13	12	http://www.dice.com/job/r	Business / Systems Analyst - Bask	UnitedHealth Group	Basking Ridge, NJ	2017-02-24	Transform health care and change the way consumers engage with
14	13	http://www.dice.com/job/r	Estimating Systems Analyst - Pho	UnitedHealth Group	Phoenix, AZ	2017-02-24	Your passion for innovation can impact millions.At Optum, we belie
15	14	http://www.dice.com/job/r	Sr. IT Systems Analyst - Eden Prai	UnitedHealth Group	Eden Prairie, MN	2017-02-24	Grow your career with an exciting opportunity with Optum, where
16	15	http://www.dice.com/job/r	IT Systems Analyst - Schaumburg	UnitedHealth Group	Schaumburg, IL	2017-02-24	Your passion for innovation can impact millions.At Optum, we belie
17	16	http://www.dice.com/job/r	IT Systems Analyst - Las Vegas, N	UnitedHealth Group	Las Vegas, NV	2017-02-24	Combine two of the fastest-growing fields on the planet with a cult
18	17	http://www.dice.com/job/r	Systems Analyst - Schaumburg, Il	UnitedHealth Group	Schaumburg, IL	2017-02-24	Your passion for innovation can impact millions.At Optum, we belie
19	18	http://www.dice.com/job/r	Senior Business Systems Analyst	UnitedHealth Group	San Antonio, TX	2017-02-24	The Senior Business Systems Analyst is responsible for creating and
20	19	http://www.dice.com/job/r	Sr. Business Systems Analyst - W	UnitedHealth Group	San Antonio, TX	2017-02-24	Your passion for innovation can impact millions.At Optum, we belie
21	20	http://www.dice.com/job/r	IT Business / Systems Analyst - R	UnitedHealth Group	Rocky Hill, CT	2017-02-24	UnitedHealth Group is a company that's on the rise. We're expandi
22	21	http://www.dice.com/job/r	Financial Aid Systems Analyst	Baylor University	Waco, TX	2017-02-17	Job Description:Baylor University is seeking a Financial Aid Systems
23	22	http://www.dice.com/job/r	IT Systems Analyst - Eden Prairie,	UnitedHealth Group	Eden Prairie, MN	2017-02-24	Grow your career with an exciting opportunity with Optum, where
24	23	http://www.dice.com/job/r	Sr Systems Analyst - Financial Ap	Vitamin Shoppe	Secaucus, NJ	2017-02-17	At the Vitamin Shoppe, Every Body Matters! We are dedicated to in
25	24	http://www.dice.com/job/r	Sr. Data Systems Analyst	MagNet	Lake Worth, FL	2017-02-10	The Sr. Data Systems Analyst (SDSA) is responsible for keeping Magl

Figure 14. Excel File Content

Main Outcome

The main outcome of this phase is the file structure produced by R (see Figure 15). There is an Excel file for each JSON formatted page of job listings being scraped. By default, the API returns information about 50 job listings at a time. Thus, each of the 50 job listings are stored in a separate Excel file as well. These saved pages collectively duplicate the information saved in the main Excel sheet (Jobs_Sat_Feb_25_12_00_37_2017.xlsx). The duplication is done to avoid complete data loss. Obtaining a dataset that contains thousands of observations may take a computer hours or even days of non-stop work. If script execution halts (e.g. Web server becomes unavailable, computer runs out of memory or freezes, there is a power outage in the building, etc.) researchers still have access to the data collected prior to the failure point. These data will be saved as Excel files up to the point when the script execution was halted. If script execution is successful, the entire dataset is saved in the main, time-stamped Excel file (Jobs_Sat_Feb_25_12_00_37_2017.xlsx). The file name means that the data were saved on Feb 25, 2017 at 12:00:37pm.

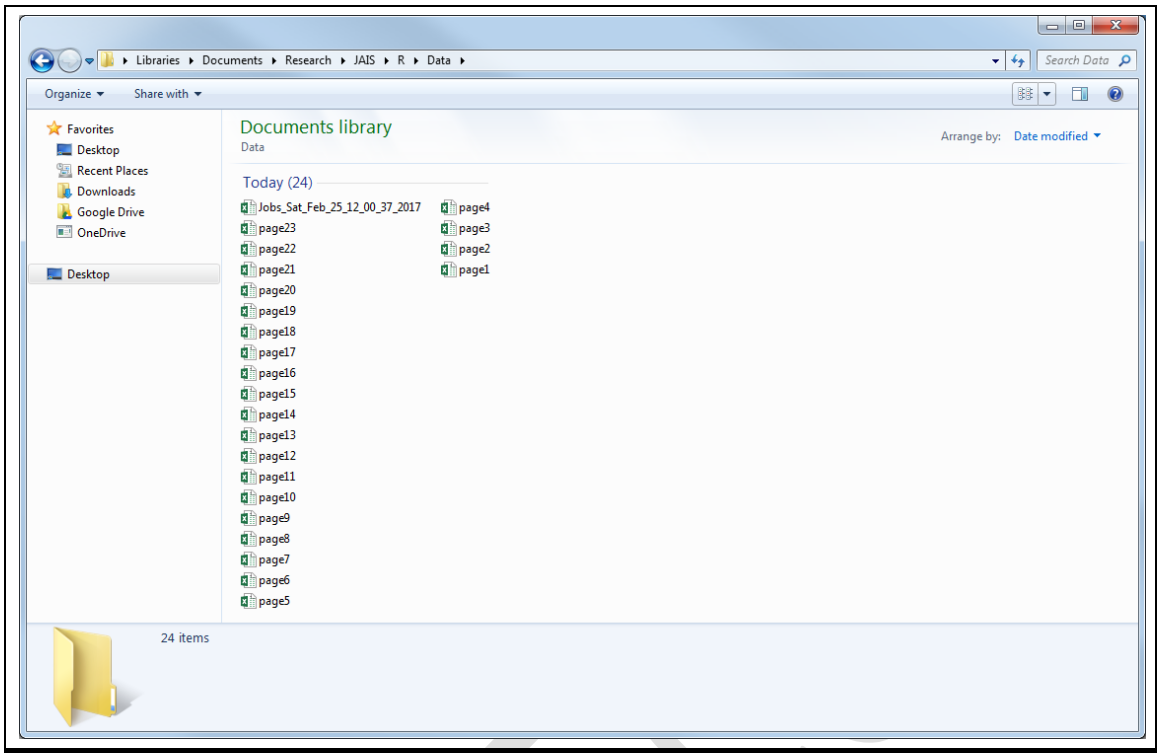


Figure 15. File Structure Created

Directions for Further Research

The data generated as a result of the web scraping session can then be imported into a software traditionally used for qualitative analysis (e.g. NVivo) or even analyzed manually (e.g. see Sandvig, 2007). The data can also be read into R (using the "xlsx" package for working with Excel files), preprocessed, and analyzed using one of the numerous text and topic mining packages available in R, such as "tm", "syuzhet", "SnowballC", "LSA", etc. Analysis of qualitative data with various text mining tools is outside the scope of this paper. Numerous tutorials and methodology papers on text mining are available (e.g. Debortoli et al., 2016). If one intends to analyze the data in R, then one can start with the text mining resources listed in Appendix C.

In general, the data collected from Dice can be used to answer a number of old and new research questions in relation to IT workforce trends, such as:

- What are the geographical patterns in IT recruitment?
- How do IT recruitment patterns change over time?
- Which geographical area has the most job openings for a particular IT role?
- What are the most in-demand technology skills in the IT labor market?
- What are the most sought after technology skills in various geographical areas?
- What are the most frequently mentioned skills for a particular IT role?
- What are the most common clusters of IT skills?
- How are various IT roles related? What are the most common clusters of various IT roles?

- How do the requirements for specific IT roles change over time?
- Are there any new clusters of IT skills originating from different IT roles and suggesting the need for a new IT role that is at the intersection of several existing roles?

If one is planning to analyze the data manually or in a semi-automated way (i.e. by using NVivo) or has limited computer processing capacity, then the data collected may exceed researchers' cognitive or computing processing capacities. For large text corpora, text mining algorithms often require a powerful computer with large amounts of RAM (Müller et al., 2016). Human mental capacity also has some obvious limitations when it comes to analyzing large volumes of text (Debortoli et al., 2016). Thus, the amount of text analyzed needs to be reduced using a formal approach.

Probably the simplest way for reducing data is to draw a simple random sample of observations from the collected data to reduce the size of the sample used for analysis. One can also use text mining algorithms to identify observations that correlate (based on word similarity) strongly with the theoretical phenomena that a research project is most concerned with (e.g. leadership skills). Alternatively, a researcher can train and then use machine-learning algorithms to score observations for the purposes of identifying those that are most relevant or those that seem to be of highest quality given the research question(s) of the study. Again, theoretical sampling techniques can be used to formally distinguish between the observations that are included and not included in the analysis. Machine learning algorithms can also be used to sort observations into groups (e.g. in accordance with the theoretical constructs used) so that each group of observations is analyzed separately by a separate researcher. Numerous other data reduction or filtering approaches are possible to make big data smaller and more manageable.

Conclusion

The World Wide Web presents researchers with vast opportunities for collecting qualitative data that reflect various theoretical phenomena in the sociomaterial realm in virtually real time. Harnessing these vast volumes of qualitative data requires a programmatic approach for automating data collection and organization. This rich resource of qualitative data is also volatile, as it is characterized by extreme velocity, variety, and veracity. Because of that, no "ready-made" web scraping tool can work all the time and fit all research purposes (just like any other tool used in qualitative research, such as NVivo). The approach presented in this paper can be extended or customized for any web scraping project. Moreover, the process framework of web scraping described in this paper can be used in conjunction with any current or future technology platform used in data science. Contrary to what some qualitative researchers believe, the framework shows that qualitative research skills are still important in the era of big data and technology-enabled enquiry. These traditional qualitative research skills can help a researcher save time and computing resources by avoiding collection of big data of poor quality, not aligned with the theory, or not organized in a way that allows the data to be analyzed by traditional and modern analysis tools. The approach to web scraping outlined in this paper widens the horizons of traditional, qualitative research tradition in IS.

References

Abraham, T., Beath, C., Bullen, C., Gallagher, K., Goles, T., Kaiser, K., & Simon, J. (2006). IT workforce trends: Implications for IS programs. *Communications of the Association for Information Systems*, *17*(1), 50.

Alharthi, A., Krotov, V., & Bowman, M. (2017). Addressing barriers to big data. *Business Horizons*. Advance online publication. doi:10.1016/j.bushor.2017.01.002

Alter, S. (2013). Work system theory: overview of core concepts, extensions, and challenges for the future. *Journal of the Association for Information Systems*, *14*(2), 72.

Banko, M., & Brill, E. (2001, July). Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting on association for computational linguistics* (pp. 26-33). Association for Computational Linguistics.

Benbasat, I., Goldstein, D. K., & Mead, M. (1987). The Case Research Strategy in Studies of Information Systems. *MIS Quarterly*, 11(3), 369-386.

Cecez-Kecmanovic, D., Galliers, R. D., Henfridsson, O., Newell, S., & Vidgen, R. (2014). The sociomateriality of information systems: current status, future directions. *MIS Quarterly*, *38*(3), 809-830.

Cisco Systems (2016). Cisco Visual Networking Index: Forecast and Methodology, 2014-2019 White Paper. Retrieved from http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html

Constantiou, I. D., & Kallinikos, J. (2015). New games, new rules: big data and the changing context of strategy. *Journal of Information Technology*, *30*(1), 44-57.

Corbin, J., & Strauss, A. (2008). Basics of qualitative research: Techniques and procedures for developing grounded theory: Sage.

Dattero, Ronald, and Stuart D. Galup. "Programming languages and gender." *Communications of the ACM* 47, no. 1 (2004): 99-102.

Davis, S., Siau, K., & Dhenuvakonda, K. (2003). A fit-gap analysis of e-business curricula vs. industry needs. *Communications of the ACM*, *46*(12), 167-177.

Debortoli, S., Müller, O., & vom Brocke, J. (2014). Comparing business intelligence and big data skills. *Business & Information Systems Engineering*, *6*(5), 289-300.

Debortoli, S., Junglas, I., Müller, O., & vom Brocke, J. (2016). Text Mining For Information Systems Researchers: An Annotated Topic Modeling Tutorial. *Communications of the Association for Information Systems*, *39*(1), 7.

Dennis, A. R., & Valacich, J. S. (2014). A replication manifesto. *AIS Transactions on Replication Research*, *1*(1), 1.

Denzin, N. K. (1994). The Art and Politics of Interpretation. In N. K. Denzin & Y. S. Lincoln (Eds.), Hanbook of Qualitative Research (pp. 500-515). London: SAGE Publications.

Denzin, N. K., & Lincoln, Y. S. (1998). Collecting and Interpreting Qualitative Materials. Thousand Oaks, California: SAGE Publications.

DHI Group (2017). About Us. Retrieved from http://www.dhigroupinc.com/home-page/default.aspx

Eisenhardt, K. M. (1989). Building Theories from Case Study Research. Academy of Management Review, 14(4), 532-550.

Feinerer, I., Hornik, K. & Meyer, D (2008). Text Mining Infrastructure in R. *Journal of Statistical Software* 25(5): 1-54. URL: http://www.jstatsoft.org/v25/i05/.

George, J. F. (2017). The Future of Academic MIS: Redux. *Journal of the Midwest Association for Information Systems*| *Vol*, *2017*(1), 9.

Granger, M. J., Dick, G., Jacobson, C. M., & Van Slyke, C. (2007). Information systems enrollments: Challenges and strategies. *Journal of Information systems education*, *18*(3), 303.

Goes, P. B. (2014). Editor's comments: big data and IS research. *MIS Quarterly*, *38*(3), iii-viii.

Goles, T., Hawk, S., & Kaiser, K. M. (2008). Information technology workforce skills: The software and IT services provider perspective. *Information Systems Frontiers*, *10*(2), 179-194.

Hirschheim, R., & Klein, H. K. (2012). A glorious and not-so-short history of the information systems field. *Journal of the Association for Information Systems*, *13*(4), 188.

Hirschheim, R., Loebbecke, C., Newman, M., & Valor, J. (2007). Offshoring and its implications for the information systems discipline: Where perception meets reality. *Communications of the Association for Information Systems*, *20*(1), 51.

Iacus, M.S. (2015). Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining. Book Review 3. *Journal of Statistical Software*, 68, 3, 1-3.

Indulska, M., Hovorka, D. S., & Recker, J. (2012). Quantitative approaches to content analysis: identifying conceptual drift across publication outlets. *European Journal of Information Systems*, *21*(1), 49-69.

Kennan, M. A., Willard, P., Cecez-Kecmanovic, D., & Wilson, C. S. (2008). IS knowledge and skills sought by employers: a content analysis of Australian IS early career online job advertisements.

Knuth, D. E. (1984). Literate programming. *The Computer Journal*, *27*(2), 97-111.

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

Light, B., & McGrath, K. (2010). Ethics and social networking sites: a disclosive analysis of Facebook. *Information Technology & People*, *23*(4), 290-311.

Lee, I. (2007). An architecture for a next-generation holistic e-recruiting system. *Communications of the ACM*, *50*(7), 81-85.

Markus, M.L. (2014). 10 questions with M. Lynne Markus. AIS, News & Press: InSider, Retrieved from https://aisnet.org/news/193546/10-Questions-with-M.-Lynne-Markus.htm.

Markus, M. L., & Robey, D. (1988). Information technology and organizational change: causal structure in theory and research. *Management science*, *34*(5), 583-598.

Mohr, L. B. (1982). *Explaining organizational behavior* (Vol. 1). San Francisco, CA: Jossey-Bass.

Müller, O., Junglas, I., vom Brocke, J., & Debortoli, S. (2016). Utilizing big data analytics for information systems research: challenges, promises and guidelines. *European Journal of Information Systems*, *25*(4), 289-302.

Munzert, S., Rubba, C., Meißner, P., & Nyhuis, D. (2015). *Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining*. Chichester, UK: John Wiley & Sons, Ltd

Olbrich, S., Frank, U., Hovorka, D., & Rowe, F. (2015, May). Trust, but Verify! The Value of Replication and Negation in IS Research. *ECIS 2015 Proceedings.*

Patterson, D. A. (2006). Revisiting my campaign statement. *Communications of the ACM*, *49*(5), 27-30.

Sandvig, J. C. (2007). Selection of server-side technologies for an e-business curriculum. *Journal of Information Systems Education*, *18*(2), 215.

Snell, J., & Care, D. (2013). Use of online data in the big data era: legal issues raised by the use of web crawling and scraping tools for analytics purposes. *Bloomberg BNA*. Retrieved from https://www.bna.com/legal-issues-raised-by-the-use-of-web-crawling-and-scraping-tools-for-analytics-purposes

Topi, H., Valacich, J. S., Wright, R. T., Kaiser, K., Nunamaker Jr, J. F., Sipior, J. C., & de Vreede, G. J. (2010). IS 2010: Curriculum guidelines for undergraduate degree programs in information systems. *Communications of the Association for Information Systems*, *26*(1), 18.

Peng, R. D. (2011). Reproducible research in computational science. *Science*, *334*(6060), 1226-1227.

Silva, L., Goel, L., & Mousavidin, E. (2009). Exploring the dynamics of blog communities: the case of MetaFilter. *Information Systems Journal*, 19(1), 55-81.

Simon, H. A. (1957). *Models of Man: Social and Rational: Mathematical Essay on Rational Human Behavior in a Social Setting*. Wiley & Sons.

Vargiu, E., & Urru, M. (2013). Exploiting web scraping in a collaborative filtering-based approach to web advertising. *Artificial Intelligence Research*, *2*(1), 44.

Watson, H. J. (2009). Tutorial: business intelligence-past, present, and future. *Communications of the Association for Information Systems*, *25*(1), 39.

Watson, H. J. (2014). Tutorial: Big data analytics: Concepts, technologies, and applications. *Communications of the Association for Information Systems*, *34*(1), 1247-1268.

Wickham, H. (2014). Tidy data. *Journal of Statistical Software, 59*(10), 1-23

Yin, R. K. (1994). Case study research: design and methods (2nd ed. Vol. 5). London: SAGE Publications.