Approaches and Practices

Journal of Business and Technical Communication
1-20

DOI: 10.1177/10506519211064619
journals.sagepub.com/home/jbt
SAGE

Project-Oriented Web Scraping in Technical Communication Research

John R. Gallagher[1] and Aaron Beveridge[2]

Abstract

This article advocates for web scraping as an effective method to augment and enhance technical and professional communication (TPC) research practices. Web scraping is used to create consistently structured and well-sampled data sets about domains, communities, demographics, and topics of interest to TPC scholars. After providing an extended description of web scraping, the authors identify technical considerations of the method and provide practitioner narratives. They then describe an overview of project-oriented web scraping. Finally, they discuss implications for the concept as a sustainable approach to developing web scraping methods for TPC research.

[1]English and Writing Studies, University of Illinois, Urbana – Champaign, Urbana, USA
[2]Informatics and Analytics, University of North Carolina at Greensboro, Greensboro, USA

Corresponding Author:
John R. Gallagher, English and Writing Studies, University of Illinois, Urbana – Champaign, 608 South Wright Street, Urbana, IL 60801, USA.
Email: johng@illinois.edu

Keywords
web scraping, data creation, data harvesting, sustainability, methods

Technical and professional communication (TPC) researchers need sustainable methods (e.g., Boettger & Friess, 2020; Friess et al., 2017; Lauer, 2017; Melonçon & St.Amant, 2019; St.Amant & Graham, 2019). According to Melonçon and St.Amant (2019), "sustainable research is that which can be replicated and confirmed, refuted, or modified through such iterative testing, and it must be considered trustworthy through its transparency of conducting and describing the actual research process" (p. 131). In response to this call for sustainable research practices, we propose web scraping as an approach that has many possible benefits for TPC. Web scraping uses command-line tools, programming languages, or other forms of computer software to collect data systematically from public websites and digital networks (e.g., Reddit). For a public website to function, it must share its underlying HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and other content with Internet browsers. This basic functionality of sharing information on browsers allows people to view and interact with websites. Whereas our browsers receive website data to display a page for viewing on a monitor, web scraping uses programming scripts or software to collect and save this data. Data collected from web scraping can include text and other content (e.g., audio, images, and video), and for digital networks, these data include user profiles, friends or followers, device data, location, timestamps, and other publicly available metadata.

Because empirically driven TPC research often requires information from websites and other digital sources, we advocate for web scraping as a viable, sustainable method for many areas of TPC research, which we address later in the article. Before delving into those areas, we first establish a general understanding of web scraping and its use for extracting large-scale, consistently structured data sets that can be tested through transparent replication. In terms of sustainable labor practices, web scraping (compared to manual data collection) speeds up data collection. Activities such as copying and pasting text from a website into a spreadsheet can be systematically automated with web scraping methods. Because web scraping automates and standardizes this process, it also reduces the possibility of missing a key data point or other data entry errors. From a "big data" perspective (e.g., Gallagher et al., 2020; Graham et al., 2015a), web scraping allows

for the creation of large-scale data sets that can test or extend previous studies conducted at a smaller scale, building on enduring findings derived from close readings, case studies, and manual collection methods. In this article, we (a) describe and define web scraping, (b) identify technical considerations of web scraping, (c) present two practitioner narratives of web scraping, (d) provide an overview of project-oriented web scraping, and (e) identify implications for the concept as a sustainable addition to TPC research methods.

Web Scraping: An Extended Description

Web scraping is the act of taking information from a website and placing it into a file for later analysis. There are different levels of scraping from using manual copy and paste to extracting data using a programming language (see Table 1). From a technical standpoint, most web scraping uses the markup language of a website, such as HTML, to identify the desired area of a web page. For instance, if researchers aimed to extract the text of the headers or paragraphs on a website, they would identify the corresponding tags (<header> or <p>). They would then determine a location on a local hard drive for that text to be placed. This information is then typically stored in a comma-separated value (CSV) file. Often, researchers scrape hundreds, thousands, or millions of web pages, targeting a specific field on the website. For example, researchers could scrape academic job ads on institutional websites (for technical resources, see Appendix).

Table 1. Types of Web Scraping According to Their Levels of Automation.

Level of Automation	Type of Web Scraping
Manual	Copy and paste
Partially automated	XML query language that pulls information based on (X)HTML tags; scraping software
Readily automated	Application programming interfaces (APIs) not always accessed through computer programming (often requires advanced programming skills)
Fully automated	Computer programming that engages full automation (Python or R are most common)

Note: HTML = HperText Markup Language.

According to Chasins et al. (2018), web scraping has two main elements: (a) extraction, such as "finding an element with a given semantic role (a title, an address), extracting a table of data," and (b) data access, such as "loading a URL, clicking a link, filling and submitting a form, using a calendar widget, autocomplete widget, or other interactive component" (p. 964).

In addition to being a data extraction tool, web scraping is a practice that asks researchers to think about how data are stored for later analysis. Consequently, web scraping underscores the labor not only of creating data sets but also of structuring them. *Structuring*, also referred to as preprocessing or wrangling, involves preparing a data set for analysis. Structuring usually involves cleaning data, which includes (a) removing needless items to reduce the size of the data set or (b) transforming scraped objects into Unicode or other machine-readable formats. An example of cleaning might be to ensure that all of the data points collected are consistently the same case (title case, tokenization, etc.) so that various forms of analysis can be conducted later in a research project (see Rawson & Muñoz, 2016). While a full discussion of preprocessing and wrangling is outside the scope of this article, we underscore the value of these qualitative aspects of web scraping because the choices made during structuring can have dramatic effects on research outcomes.

Web scraping tends to be targeted, focusing on specific communities, demographics, industries, topics, locations, or domains. This targeted approach to scalable data collection is not just for websites: Web scraping can also refer to the use of application programming interfaces (APIs) and other related approaches to collecting data from digital networks and web or mobile applications. Unlike web scraping that targets specific HTML or CSS tags to collect data, API scraping accesses already available data, often delivered in JavaScript Object Notation (JSON). For example, Twitter has one of the more accessible APIs of the larger social networks, and to access this data from Twitter, researchers upgrade their accounts to gain "developer" access to Twitter's API. Once access is granted, a researcher may access data on trends, users' tweets, and other data made available to developers by Twitter. For Twitter and similar websites, API access provides a viable alternative to directly scraping the website because Twitter actively blocks the Internet Protocol (IP) addresses of programs that scrape data from its site. Twitter and similar networks do this for two key reasons: (a) If they did not block scraping programs, then servers and bandwidth needed for nondeveloper use of their site would be spent on scraping programs that are collecting data without returning any value to the network, and (b) data are one of Twitter's most important revenue streams, and thus Twitter limits the amount of available data. These two

approaches to web scraping—using one's own programs to conduct web scraping and using a website's API—represent most web scraping activities.

Practitioner Stories of Web Scraping

This section provides practitioner narratives. The first narrative highlights how web scraping can enhance research inquiries by confirming intuitions and impressions with quantitative evidence. The second narrates practitioner concerns related to technical learning curves and the overall time and labor considerations for research.

John Gallagher's Narrative

In the spring and summer of 2013, I was conducting research for my dissertation. In this project, I was studying how a journalist, Tracy Monroe, moderated a private Facebook group she had created for civil political discussions (Gallagher, 2014, 2015). I realized from my qualitative coding and impressions that Monroe was essentially managing the group through her participation. From my observations, I thought she was the most active member; my impression was that she posted and commented the most. But my impressions were not satisfying to me; I wanted to determine if she was in fact the most active participant of the group. To answer this question, I wanted to go to the start of the group (chronologically) and download all of the group's activities. But doing so manually on Facebook was exceedingly tricky because Facebook used RSS feeds and infinite scrolling techniques to prevent easy access.

At this time, however, Facebook still allowed noncommercial users to access its API. (While Facebook's API is no longer publicly open to anyone, researchers can still request an API key.) After being granted permission to study the group and to become an administrator of it, I obtained a Facebook API key. Following the directions that Facebook provided, I scraped the entire group; this took about a month to figure out, including the technical side of the API. The syntax of each network's API can vary, but the developer documentation for each network provides details on how to navigate these differences. I did not have any programming experience before this and only moderate experience with web development. But with a moderate amount of effort (about 4 weeks during July 2013), I was able to structure the data, which came in a CSV file. Using this information, I found that Monroe was indeed the most active participant of the group (see Table 2).

Table 2. Analysis of Web Scraped User Activity Within a Facebook Group.

Activity	Number (%)
Total activity	
Total initial posts and comments	5,622
Total initial posts and comments by Monroe	1,468 (26.1)
Initial posts	
Total initial posts by entire group:	847
Total initial posts by Monroe	238 (28.1)
Comments	
Total number of comments	4,775
Total number of comments by Monroe	1,230 (25.8)
Average character length (ACL) of comments	
ACL	282
ACL of Monroe's comments	377
ACL of other comments	250

The data confirmed my intuitions about the degree to which Monroe participated in the group. For instance, my scraped data enabled me to substantiate my interpretation with a quantitative method: Monroe was responsible for about 25% of all activity in a group of over 100 users (128 members at the time of data collection). Moreover, her comments were substantially longer than the average comment, which meant that she was not only the most frequent participating member but also writing longer texts than most members. I then associated this evidence with my qualitative readings of Monroe's texts and the three interviews I conducted with her. Because Monroe was the group's moderator, I suspected she was modeling behavior for the group through her texts as a form of management and moderation. The evidence in Table 2 provided me with quantitative evidence that she was the most active member of the group; I qualitatively argued (via close reading) that many of those posts and comments demonstrated moderation through her explicit, active participation. My quantitative evidence thus complemented my qualitative readings.

Aaron Beveridge's Narrative

My story relates to a problem many researchers in TPC face: weighing the time investment associated with learning web scraping technologies against the time it takes to manually collect and code data for analysis. Although

many web scraping practitioners and programmers come from a computer science, statistics, or similar technical background, I taught myself how to program during my PhD in English at the University of Florida. With no formal training in computer science, I learned how to program with a project-oriented approach. This *project-oriented* approach to programming meant that for each new research project or publication I pursued, I learned just enough programming techniques to effectively complete that individual project—rather than having to learn an entire programming language. With each new project, my programming abilities were extended, and I was often able to repeat the same skills in new projects using revised versions of the program I already tested and trusted.

When weighing the work involved in hand-collecting web and digital artifacts for large research projects against the time involved in learning automated methods for completing those same tasks, many researchers often default to whatever methods they are already comfortable using. There is certainly nothing wrong with that approach, but what if it were faster to both complete a new study or publication and learn automated methods? In other words, if researchers compared the time invested in reusing manual methods to collect data against the time that they would take to learn automated methods while conducting a new study, they would certainly find it difficult to dismiss web scraping and programing if the overall time investment was less. This was precisely my experience when collaborating with Claire Lauer and Eva Brumberger to write "Hand Collecting and Coding Versus Data-Driven Methods in Technical and Professional Communication" (Lauer et al., 2018).

While our article focused primarily on analysis—comparing hand-coded analyses (close readings) with descriptive approaches to natural language processing (using computer programs to count words and compare their relative-document relationships statistically)—we also tested web scraping as a replacement for the manual data collection techniques that Lauer and Brumberger initially used to create the data for their study. As a collaborator on this project, my primary role was producing scripts in the R programming language for web scraping and for testing the comparative analyses. While I had previous experience in basic web design and in completing R analyses of text using the text mining (TM) package, I had never completed any form of web scraping that required custom programming scripts to systematically collect data.

Initially, Lauer and Brumberger spent around 2–3 hours per week hand collecting 3,000 PDF copies of technical communication jobs posted on Monster.com for a total of 8 weeks and then spent additional time reviewing

the PDFs to "[cull] duplicate postings; we also deleted any postings that were not clearly for technical communication positions." In all, the work expended left them with "approximately 1500 job postings" for the initial study (Lauer et al., p. 394). To reproduce these methods with web scraping, I spent approximately 2 days programming. The first day I learned web scraping with R, walking through an introductory tutorial and then writing a script that automatically processed the search results returned from Monster for technical writing positions. The second day I consulted with a trusted R expert to review my scripts and help me troubleshoot to ensure that the web scraping script produced data as expected. In all, I spent less than 15 hours of work, but I now had a program that I could use to extract the text and metadata from Monster.com job postings indefinitely, without any additional labor for data collection. It took my program 109.74 seconds to access the search results for "technical writer" on Monster.com and then visit each page in the results to scrape data on the 100 available jobs. If that script were automated over 6 months, it would have potentially resulted in over 27,000 job postings for analysis (assuming the test scales effectively over time), without any additional labor beyond checking that the program was continuing to return data as expected.

As I argue here, if the time expended on manual data collection and other similar practices were instead spent learning automated, sustainable techniques for those same tasks, the results would be not only less total time spent but also larger data sets covering longer periods of time. Certainly, not all research tasks can be automated, but there remain many manual data collection activities that can be augmented by web scraping methods.

Technical Considerations

Now that we have provided narratives for web scraping, we address four technical concerns related to web scraping. The first is the inconsistent nature of web design. If a website does not use a consistent, front-end architecture or framework, such as bootstrap or CSS grid, then an automated web scraper might not yield complete or consistent data. For example, if a blog uses manually crafted paragraph tags (<p>) and inconsistently uses CSS classes, then any automated program could miss or fail to collect key aspects of a website. While this is less of a concern when web scraping the websites of large organizations with automated systems for building websites (e.g., corporations and institutions), scraping the websites of smaller organizations with manually produced websites might produce incomplete or inconsistent data. Therefore, effective web scraping requires

qualitative work, such as spot-checking, to ensure that scraped data is consistent.

Second, researchers need to have patience with web scraping programs. If information is pulled too quickly, and if the program makes too many server requests ("pings"), it is possible to accidentally simulate a denial of service (DoS) attack. That is, researchers could end up shutting down a smaller website or being denied access to larger websites that block the researcher's IP address. Researchers should follow best practices in accessing a website's robots.txt file to see if the site has a preferred page visit delay time for scraping the site.

Third, information scraped from a website looks very different from the way it appears in file format. As a result, researchers need to determine the extent of such changes and consider how the transition from web page to data set affects the research design of the project. Researchers need to weigh to what extent scraped data differ from the source content that was rendered in a browser. This determination depends on which content is scraped from a site and how it addresses a particular research question in various subfields of TPC.

The fourth technical consideration is related to training in technical and quantitative methods. Because most TPC programs are housed in English departments (Scott & Melonçon, 2017), technical communication researchers might not receive adequate training in web scraping or the related quantitative methods needed to analyze large data sets. Disciplinary divides make methods such as web scraping fall outside the scope of English departments. Our suggestion here is to seek interdisciplinary opportunities with nonhumanities researchers. We also suggest identifying the many projects and tool sets funded by the National Endowment for the Humanities Office of Digital Humanities. Projects such as Voyant Tools and MassMine, along with adjacent projects in history such as *Exploring Big Historical Data: The Historian's Macroscope* (Graham et al., 2015b), have made significant progress in closing the technical gap for humanities scholars (for resources, see Appendix).

These technical considerations, combined with our narratives, demonstrate that while web scraping is a technical method requiring a moderate time investment, it does not require months of work or complete knowledge of a computer programming language. In fact, we would not recommend learning a computer language for its own sake. Rather, we recommend a project-based approach.

Project-Oriented Web Scraping

Project-oriented web scraping counters assumptions that are often made about the technical skills required to use various types of web scraping methods. Advanced programming knowledge and an extensive understanding of web design (HTML/CSS/JavaScript) are needed to address the most complex web scraping projects. But for many researchers, becoming fluent in these areas would not result in enough new data, publications, or projects to justify the overall time invested in doing so. In contrast, a project-oriented approach to web scraping (and likewise to programming and web design) is analogous to learning just enough words and phrases in a foreign language to order food and ask for help when traveling to a foreign country: If you simply need someone to point you in the direction of the nearest washroom or show you where the closest bus stop is located, then you only need to learn a few keywords and expressions. You do not need to become fluent in an entire language. A project-oriented approach to web scraping follows this same functional rationale.

On the surface, a project-oriented approach means just that: learning just enough technical skills to complete a single project. But if we are to place this project-oriented approach within a broader call for sustainable research practices, then there are macroscopic implications to address. For example, a simple browser add-on like webscraper.io can provide immense scalability and automate the collection of data across complex, dynamic websites, but cloud tools such as webscraper.io usually do not have a long shelf life. Cloud tools often use a no-fee model to gain customers and build a following only to later require substantial fees, or, worse, the tools become inaccessible when the owners of the tools sell out entirely to a larger corporation like Google or Microsoft. While we see nothing wrong with using cloud tools such as webscraper.io—especially for advanced researchers who are confident they will likely use web scraping only for a single project—there are long-term costs for replicability when methods workflows are erased with the loss of access or the sudden emergence of paywalls.

For junior faculty and graduate students, who may reuse web scraping methods across multiple projects and years, we believe that learning just enough of the R language using Rvest or the Python language using Beautiful Soup conveys many benefits without requiring a significant time investment. For either language, no previous experience is needed to learn just enough programming to web scrape, and both Rvest and Beautiful Soup (see Appendix) have immense, ubiquitous tutorial support in both

written and video form. Additionally, R and Python support for novice programmers is available on most university campuses, providing the opportunity to collaborate with more advanced programmers or get feedback on drafts of programming scripts (even from our own students). With supplementary tools such as SelectorGadget, which collects key page targets (HTML tags, CSS classes, and Xpaths) with simple point-and-click web page interactions, no additional time is needed to uncover complex web designs or learn developer tools for inspecting HTML, CSS, or JavaScript in web browsers. While the overall time investment would be a week or two—as opposed to a day or two spent learning a cloud tool such as webscraper.io—the clear benefit of learning to program comes from using technologies that are likely to remain open and accessible for decades into the future.

A project-oriented approach also allows researchers to extend their programming capabilities, web design knowledge, and related data-driven skills as they take on more complex projects, building on already-tested scripts and programs to use them across several projects. For example, for the first project, a researcher might employ some basic R programming using Rvest and SelectorGadget to automate the collection of page content and metadata across a large list of static websites and then, in subsequent projects, build on these skills by learning how to count the most frequent words and infer topics across thousands of website pages. Further, in a later project, that researcher could learn how to save the data in a database and provide a dashboard of visualizations using R Shiny or Plotly with Python (see Appendix for a list of resources). If each step is accompanied by a grant application, a methods or research publication, or some other credentialing artifact for scholarship and research, then the project-oriented approach provides a pathway for junior faculty and graduate student TPC researchers to sustainably import advanced programming and data-driven skills into the field.

Finally, a project-oriented approach could be undertaken in a collaborative form. With collaborative programming and documentation websites such as GitHub, it is easy for researchers to work together in documenting sustainable, scalable data collection techniques as applied to TPC research areas. We must continue to encourage the open sharing of programming scripts and extend already available workflows by engaging and testing emerging technologies related to automated data collection practices. With relatively new peer-review resources such as the *Journal of Open Source Software* and other similar endeavors, a project-oriented approach could eventually build momentum so that TPC researchers will develop our own tools and resources that have themselves been programmed and

tested by TPC researchers. Rather than front-loading graduate education with calls for programming and statistical education, we can call for a project-oriented approach that allows researchers to extend our field's technical and data-driven skill sets organically, based on the actual projects undertaken by researchers.

Conclusion: Web Scraping as a Sustainable Research Method

As a scalable method, web scraping supports sustainable methods for TPC research in four ways. First, web scraping supports transparent and data-driven methods, allowing researchers to share scraping programs and open data. Second, web scraping can be iterated to generate new variations of data or extend already available archives through community collaboration. Third, web scraping procedures can be documented with instructions for software or available scraping programs; this documentation might lead to a stronger sense of trust in research designs and results. Fourth, web scraping can yield either a random sample or a complete sample of website content and metadata. Although TPC researchers have long collected information from the Internet, they have been limited by manual collection processes until recent examples have deployed web scraping as a method (Gallagher et al., 2020; Gallagher & Holmes, 2019). But these studies have not gone into extensive detail about web scraping as a method, and significant work remains to determine which methodologies can be transformed by automated, data-driven practices.

Web scraping is an effective complement to the typical TPC research methods, such as interviews, surveys, experiments, usability tests, document reviews, case studies, and ethnographic work (for empirical studies of typical research methods, see Carliner et al., 2011; Melonçon & St.Amant, 2019). Web scraping augments interview and survey methods through verification or evaluation of user responses. If researchers have interview results or survey answers about websites, they can scrape a website to determine if the perceptions are aligned with actual behaviors. For example, if researchers are interested in the number of times users tweet, they may augment their survey methods by also collecting public data on user activity to compare to survey responses. A distinct advantage arises with such verification possibilities: the opportunity for quasi-experimental and fully experimental research designs. TPC researchers could design surveys or interviews about online habits, behaviors, or

activities and then scrape data from websites or digital networks to determine if their perceptions match the actualities. This opportunity would assist in creating more detailed and large-scale studies of user experiences as well as feasibility studies.

In terms of document reviews, case studies, and ethnographic work, web scraping potentially allows for more comprehensive samples to be collected, thereby increasing the number of data points within studies and reducing the possibility for bias. More data likely would yield more detailed studies, which again answers Melonçon and St.Amant's (2019) call for sustainability through scalability. Case study approaches, as one example, would benefit greatly from web scraping. Case studies, central to multiple research agendas in TPC and other writing studies fields (e.g., composition), are often used inconsistently (Moriarty et al., 2019) and without much reflection about what the concept of case study means (Gallagher, 2019). Case studies are holistic, bounded systems; web scraping can consequently yield systems that provide improved consistency through the information available online.

Scalable methods allow for more frequent collaboration in quantitative and computational methods, neither of which is used frequently in TPC. Quantitative methods tend toward reproducible findings and lead researchers to discover systemic issues or biases underlying technical documentation and other technical practices. Increasing our fluency in these methods improves our ability to collaborate with researchers in computer science, data science, and other STEM fields. Additionally, TPC could benefit greatly if its academic programs focused more graduate education on computational approaches that are directly relevant to our language and linguistic training, such as topic modeling, sentiment analysis, named entity recognition, and a multitude of other natural language-processing techniques. These approaches could produce results that supplement the types of findings we often gain from circulating surveys and enable researchers to see latent patterns that cannot be readily discerned through close readings or case studies.

With respect to TPC content areas, web scraping has a wide range of applications. It can be used to gather social media data, content metadata, user reviews, tutorial feedback, online comments, and discussion forum responses. Because web scraping enables large and holistic data sets, it can aid with content moderation (e.g., Andersen, 2014; Batova & Clark, 2015). It could also assist areas related to data visualization (e.g., Melonçon & Warner, 2017) and visual rhetoric (e.g., Welhausen, 2015) by enabling more charts, graphs, and other visual communications to be

not only analyzed but also deployed in our research. Web scraping assists with documenting and archiving public data, such as government data on climate change. Consequently, climate change research (Cagle & Tillery, 2015) and environmental design (Sackey, 2020) would be advanced through the creation of archives that save data before it is lost or destroyed. The subfields of public science writing (Mehlenbacher, 2017, 2019) and rhetoric of health and medicine would benefit from being able to use APIs to scrape social media and other forums to better understand the trends of users who identify as patients. Ultimately, these areas would benefit from the many ways that web scraping extends their observational and archival capacities—extending the sustainability of these areas of research.

While web scraping does not automatically help us to ask new research questions, it enables us to pose questions that are larger in scale than most questions posed in previous TPC research. Web scraping can also help reduce the labor of TPC researchers by shifting us away from manual collection procedures and toward automated collection procedures. While this turn to scale is not without its perils—larger data sets are not inherently better than smaller ones—we believe that the advantages outweigh the disadvantages. But web scraping promises other benefits in addition to scale. It can reduce errors when collecting data from websites. Web scraping places structured data into files that can be more easily shared across multiperson teams and for future researchers to access. If we adopt methods that build systematically from project-oriented web scraping, with considerations for other closely related areas such as natural language processing, we believe the accumulated benefits will make TPC research more sustainable for years to come.

Acknowledgments

The authors would like to thank the two anonymous reviewers and Jo Mackiewicz for their patience on working through this article.

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

ORCID iD

John R. Gallagher https://orcid.org/0000-0001-5980-4335

References

Andersen, R. (2014). Rhetorical work in the age of content management: Implications for the field of technical communication. *Journal of Business and Technical Communication*, *28*(2), 115–157. https://doi.org/10.1177/1050651913513904

Batova, T., & Clark, D. (2015). The complexities of globalized content management. *Journal of Business and Technical Communication*, *29*(2), 221–235. https://doi.org/10.1177/1050651914562472

Boettger, B. R. K., & Friess, E. (2020). Content and authorship patterns in technical communication journals (1996–2017): A quantitative content analysis. *Technical Communication*, *67*(3), 4–24.

Cagle, L. E., & Tillery, D. (2015). Climate change research across disciplines: The value and uses of multidisciplinary research reviews for technical communication. *Technical Communication Quarterly*, *24*(2), 147–163. https://doi.org/10.1080/10572252.2015.1001296

Carliner, S., Coppola, N., Grady, H., & Hayhoe, G. (2011). What does the transactions publish? What do transactions' readers want to read? *IEEE Transactions on Professional Communication*, *54*(4), 341–359. https://doi.org/10.1109/TPC.2011.2173228

Chasins, S. E., Mueller, M., & Bodik, R. (2018). Rousillon: Scraping distributed hierarchical web data. In The 31st Annual ACM Symposium on User Interface Software and Technology—UIST '18 (pp. 963–975). https://doi.org/10.1145/3242587.3242661

Friess, E., Boettger, R. K., Campbell, K. S., & Lam, C. (2017). Are we missing the boat? A roundtable discussion on research methods and how they define our field. *IEEE International Professional Communication Conference*, 1–3. https://doi.org/10.1109/IPCC.2017.8013942

Gallagher, J. R. (2014). *Interactive audience and the internet* [Doctoral dissertation, University of Massachusetts, Amherst]. https://scholarworks.umass.edu/dissertations_2/190/

Gallagher, J. R. (2015). Five strategies internet writers use to "continue the conversation." *Written Communication*, *32*(4), 396–425. https://doi.org/10.1177/0741088315601006

Gallagher, J. R. (2019). A framework for internet case study methodology in writing studies. *Computers and Composition*, *54*, 1–14. https://doi.org/10.1016/j.compcom.2019.102509

Gallagher, J. R., Chen, Y., Wagner, K., Wang, X., Zeng, J., & Kong, A. L. (2020). Peering at the internet abyss: Using big data audience analysis to understand online comments. *Technical Communication Quarterly*, *29*(2), 155–173. https://doi.org/10.1080/10572252.2019.1634766.

Gallagher, J. R., & Holmes, S. (2019). Empty templates: The ethical habits of empty state pages. *Technical Communication Quarterly*, *28*(3), 271–283. https://doi.org/10.1080/10572252.2018.1564367

Graham, S. S., Kim, S.-Y., DeVasto, D. M., & Keith, W. (2015a). Statistical genre analysis: Toward big data methodologies in technical communication. *Technical Communication Quarterly*, *24*(1), 70–104. https://doi.org/10.1080/10572252.2015.975955

Graham, S., Milligan, I., & Weingart, S. (2015b). *Exploring big historical data: The historian's macroscope*. World Scientific.

Lauer, C. (2017). Re-considering research: Why we need to adopt a mixed-methods approach to our work. *Communication Design Quarterly*, *4*(3), 46–50. https://doi.org/10.1145/3071078.3071085

Lauer, C., Brumberger, E., & Beveridge, A. (2018). Hand collecting and coding versus data-driven methods in technical and professional communication research. *IEEE Transactions on Professional Communication*, *61*(4), 389–408. https://doi.org/10.1109/TPC.2018.2870632

Mehlenbacher, A. R. (2017). Crowdfunding science: Exigencies and strategies in an emerging genre of science communication. *Technical Communication Quarterly*, *26*(2), 127–144. https://doi.org/10.1080/10572252.2017.1287361

Mehlenbacher, A. R. (2019). *Science communication online: Engaging experts and publics on the internet*. Ohio State University Press.

Melonçon, L., & St.Amant, K. (2019). Empirical research in technical and professional communication: A 5-year examination of research methods and a call for research sustainability. *Journal of Technical Writing and Communication*, *49*(2), 128–155. https://doi.org/10.1177/0047281618764611

Melonçon, L., & Warner, E. (2017). Data visualizations: A literature review and opportunities for technical and professional communication. In IEEE International Professional Communication Conference (pp. 1–9). https://doi.org/10.1109/IPCC.2017.8013960

Moriarty, D., Núñez De Villavicencio, P., Black, L. A., Bustos, M., Cai, H., Mehlenbacher, B., & Mehlenbacher, A. R. (2019). Durable research, portable findings: Rhetorical methods in case study research. *Technical Communication Quarterly*, *28*(2), 124–136. https://doi.org/10.1080/10572252.2019.1588376

Rawson, K., & Muñoz, T. (2016). *Against cleaning*. http://curatingmenus.org/articles/against-cleaning

Sackey, D. J. (2020). One-size-fits-none: A heuristic for proactive value sensitive environmental design. *Technical Communication Quarterly*, *29*(1), 33–48. https://doi.org/10.1080/10572252.2019.1634767

Scott, J. B., & Melonçon, L. (2017). Writing and rhetoric majors, disciplinarity, and "techne." In *Composition Forum* (Vol. 35). Association of Teachers of Advanced Composition. https://files.eric.ed.gov/fulltext/EJ1137850.pdf

St.Amant, K., & Graham, S. S. (2019). Research that resonates: A perspective on durable and portable approaches to scholarship in technical communication and rhetoric of science. *Technical Communication Quarterly*, *28*(2), 99–111. https://doi.org/10.1080/10572252.2019.1591118

Welhausen, C. A. (2015). Visualizing a non-pandemic: Considerations for communicating public health risks in intercultural contexts. *Technical Communication*, *62*(4), 244–257.

Author Biographies

John R. Gallagher is an assistant professor at the University of Illinois, Urbana–Champaign. He studies interfaces, digital rhetoric, participatory audiences, and technical communication. His book, *Update Culture and the Afterlife of Digital Writing* (2020), is available from Utah State University Press.

Aaron Beveridge is an assistant professor at the University of North Carolina at Greensboro. His research and teaching interests include social network content analysis, diffusion of innovations, rhetoric, and technical and professional communication.

Appendix

Web Scraping Tools and Related Resources

We have organized the following list of web scraping tools in terms of their power and broad applicability. Starting with programming libraries developed for R and Python, which provide the most powerful and extensive capabilities for any web scraping project, we then provide lists of tools that are less capable but more accessible. In general, the more scalable and capable a tool is for web scraping, the more extensive the technical knowledge that is required to deploy the tool for large research projects.

That said, both the web scraping tools made available in command-line environments and those that are available in graphical user interface (GUI) software can deliver massive data sets to address complex research questions. The main difference, however, is in the amount of customization that is available. For GUI software, in particular, researchers are always limited by the types of access and the types of data these tools are capable of delivering, but with web scraping libraries in programming languages such as R and Python, you can rely on the full creative power of Turing-complete programming. In other words, whatever limitations that exist for web scraping libraries in R and Python can be overcome by motivated programmers who add to those libraries or create entirely new, more extensive programs of their own.

Programming Libraries

Table A1 lists our preferred programming libraries in each of the languages. But there are many more libraries available for each of the categories listed below, and we encourage you to test others if you are comfortable programming in any of the languages. We provide these as starting points for researchers who might be learning to program in R or Python along with learning to scrape data from the web.

Aaron Beveridge produced these as introductory examples for an interdisciplinary graduate course in Text Mining and Natural Language Processing at UNCG. See the following GitHub repository for Rvest code examples: https://github.com/aabeveridge/rvest_tutorial1.

Software

While there are many paid solutions, we do not list them here because often the pricing is too prohibitive for research projects without a profit-bearing outcome or

other large funding sources backing the research. Additionally, most of the commercial software available has been developed for marketing and product development research and not with the academic researcher in mind. None of the tools listed in Table A2 require any advanced programming or technical knowledge.

Browser Add-Ons

Although there are many browser add-ons available to make web scraping easier for projects, the two listed in Table A3 are the most common ones used in our projects. If you encounter a certain task or method in your project that requires seemingly needless manual labor, we suggest searching for a browser add-on to potentially make that work more efficient. We have provided a list of Chrome extensions, but many of them have Firefox alternatives as well.

Table A1. Programming Libraries for Web Scraping.

Library	Description	Language
Rvest	General web scraping	R
RSelenium	General web scraping	R
twitteR	Twitter API scraping	R
tuber	YouTube API scraping	R
RedditExtractoR	Reddit API scraping	Python
Beautiful Soup	General web scraping	Python
search-tweets-python	Twitter API scraping	Python
PRAW	Reddit API scraping	Python
youtube-data-api	YouTube API scraping	Python

Note: API = application programming interface.

Table A2. Software for Web Scraping.

Software	Description
MassMine	General web scraping and API scraping
Octoparse	General web scraping, free up to 10k records
Parsehub	General web scraping

Note: API = application programming interface.

Table A3. Browser Add-Ons for Web Scraping.

Add-On	Description	Browser
webscraper.io	General web scraping	Chrome, Firefox
Scraper	General web scraping	Chrome
SelectorGadget	HTML/CSS item selector/inspector	Chrome
AutoScroll	Auto scrolls web pages	Chrome

Note: CSS = Cascading Style Sheets; HTML = HyperText Markup Language.