Leveraging Web Technologies for Statistical Graphics and Computing

Carson Sievert

2015-10-30

Contents

1 Chapter 1: Problem Statement

Web technologies offer exciting new opportunities to advance the field of statistical computing; particularly in the areas of data acquisition and statistical interactive graphics. In theory, freely available data on the Web is accessible; but in practice, one needs an extensive knowledge of Web technologies to acquire it. With better software for curating data on the Web, we can reduce barriers to access, and enable a reproducible workflow.

Reproducibility and portability have remained a challenge for interactive statistical graphics for decades. Interactive statistical graphics software is often difficult to install since it typically assumes non-standard software is available on users' systems. Modern Web browsers with `HTML5` support are now ubiquitous, and provide a rich ecosystem for interactive graphics. However, interfacing this ecosystem with statistical software remains difficult and, depending on the application, can require a myriad technologies. Again, with better software, we can reduce the startup costs involved with producing web-based interactive statistical graphics.

The current trend in web-based interactive statistical graphics is provide various language bindings to `JavaScript` charting libraries. To test whether the entire software stack is working as intended, it's common to verify properties of the data sent to the binding, but this does not guarantee that the end result is what we expect. A proper testing framework for this type of software should be able to construct and manipulate the Document Object Model (DOM) using technologies available to modern Web browsers. To our knowledge, **animint** is the first R package to implement this testing approach, and some of the lessons learned could be used to construct a more reliable and easier to use testing suite.

2 Chapter 2: Overview

This proposal is a collection of work towards making web-based interactive statistical graphics and data acquisition techniques more accessible to `R` programmers. To date, I have authored or co-authored over 7 `R` packages in this direction, and currently maintain these packages: **pitchRx**, **bbscrapeR**, **XML2R**, **rdom**, **LDAvis**, **animint**, **plotly**. This section provides background on work that has enabled these projects to exist. It also briefly summaries the contributions made by these packages, and points out remaining features yet to be implemented.

The `R` packages **pitchRx**, **bbscrapeR**, **XML2R**, and **rdom** all provide utilities for acquiring data hosted on the Web. Amongst them, **pitchRx** and **bbscrapeR** have the highest level interface and are aimed at a known set of Uniform Resource Locators (URLs). Their interface is designed so that the user does not need any understanding of underlying file formats that contain the data (e.g., `XML`, `JSON`). These packages provide function(s) that download, parse, and transform data of interest so that it is in a shape more suitable for statistical analysis. In a sense, they provide *access* to the data, rather than the data itself, so they avoid any legal issues with rehosting data. Furthermore, the packages are self-documenting, so users can verify the cleaning and transformations performed on the data to ensure its integrity.

pitchRx and **bbscrapeR** leverage **XML2R**: a framework for transforming XML content into tables. **XML2R** is a wrapper around the **XML** package which provides low-level `R` bindings to the libxml2 `C` library for parsing XML content (Lang and CRAN Team 2015) (Veillard 2006). **XML2R** makes it possible to parse, filter, and transform XML content into table(s) without any knowledge of the verbose `XPATH` and `XSLT` languages. These high-level semantics make it easier to maintain projects such as **pitchRx** and **bbscrapeR** since it drastically reduces the amount of code. Chapter Taming PITCHf/x Data with XML2R and pitchRx explains these ideas in more detail.

rdom makes it easy to render dynamic web pages and access the Document Object Model (DOM) from `R` via the headless browsing engine phantomjs (Sievert 2015). This fills a void where other web scraping packages in `R` currently fall short. They can download and parse the HTML page source, which is static, but they lack a browsing engine to fully render the DOM. If the DOM cannot be rendered, content that is dynamically generated (e.g., with `JavaScript`) cannot be acquired. The `R` package **RSelenium** can also render dynamic web pages and simulate user actions, but its design makes it harder to use and less reliable compared to **rdom**.

meta vis:

- Difference between technique-based vis and problem-based. **LDAvis** is inspired by a problem, which may require access to lower-level visualization techniques. **animint** is technique-based.

animint:

- Discovered **RSelenium** and authored the infrastructure to test the renderer.
- Wrote bindings for embedding plots inside of knitr/rmarkdown/shiny documents, before the advent of **htmlwidgets**, which provides standard conventions for writing such bindings. At the time of writing, **htmlwidgets** can only be rendered from the R console, the R Studio viewer, and using R Markdown (v2). For this reason, we decide to not use **htmlwidgets** since users may want to incorporate this work into a different workflow.
- Implemented facets.
- Mentored Kevin Ferris in the theming and selectize widget implementation.

The **LDAvis** package creates an interactive web-based visualization of a topic model fit to a corpus of text data using Latent Dirichlet Allocation (LDA). The visualization uses advanced interaction techniques that are not currently possible using higher-level tools such as **shiny**.

- Designed and authored most of the initial implementation -> https://gallery.shinyapps.io/LDAelife
- Helped implement the completely client-side application -> https://cpsievert.github.io/LDAvis/reviews/vis/

3 Chapter 3: Scope

This section details the contributions I've made on joint projects; in particular:**LDAvis**, **animint**, and **plotly**.

3.1 Motivating Reproducible Data Analysis

Although used inconsistently, reproducible research implies output(s) which support its findings can be recreated from a given dataset or methodology. Reproducible research is different from replicable research which means the findings can be supported from a different dataset or methodology.

As (Drummond 2009) points out, it is important to distinguish between reproducibility (ability to recreate *outputs* from a given experiment) and replicatibility (ability to recreate *findings* through a different experiment) in scientific research. The latter is certainly more robust in verifying the validity of scientific findings, but some would disagree with Drummond's opinion that reproducibility is "not worth having".

A survey by (Stodden 2010) found that the top reason for not sharing code and data from publications was because the process is too time-consuming.

Replication forces one to be organized and provide detailed instructions on

1. Replication encourages good organizational allows authors to quickly revisit and correct issues
2. Transparent replication fosters a better review process
3. Replication enhances chances of reproduction

- Point out the importance of *transparent* replication!
- Define Reproducible Computational Research (RCR)!

3.2 Literate programming

Literate programming should be viewed as a necessary, but not a sufficient, component of a RCR project (Knuth 1992). It removes any possibility of human errors during manual copy-and-paste actions, encourages transparency, and reduces distance from the actual analysis to its final presentation. As a result, its much easier for reviewers to verify correctness, for consumers of the research extend the work, and for authors to return to the work.

In literature programming, computer code and plain text are intermingled in a *source document*, then a *transformer* compiles the source into an output document (e.g. PDF, HTML) for viewing. A transformer must be able to evaluate code, weave together any output that the code may produce, and other output specific styling.

If we treat data as an input to the source document, the output document becomes *dynamic* in the sense that altering data input alters results in the output document.

For years, Sweave was used to implement literate programming for the R environment.

- Sweave -> knitr enables reproducibility with minimal start-up costs
- For those familiar with command-line tools, `make` provides a more fully featured and flexible way to define dependencies between inputs and outputs of an analysis.
-

3.3 R packages as a research compendium

> "Reproducibility is hard. It will probably always be hard, because it's hard keeping things organized." (Broman 2015)

Weave together ideas of (Rossini 2001); (Leisch 2002); (Gentleman and Lang 2004)

(Gentleman and Lang 2004) – compendiums enable reproducibility but aren't sufficient for independent verification.

3.4 Version control for auxiliary software

- **checkpoint**
- Pros:
  - easy to use (just specify a date)
- Cons:
  - Assumes all the software used for analysis existed on CRAN on a specific date (what about GitHub/SourceForge/etc packages?)
  - Only goes back to September 2014 https://twitter.com/revodavid/status/651482547063529477
- **packrat**
- Pros:
  - Packages can be installed off of CRAN, GitHub, or locally from source
  -
- Cons:
  -

Project specific libraries are cumbersome to maintain; particularly if one uses a common set of packages across their projects. One workaround to this inhibitor is to only "initialize" the project specific library after the project is done, so the author can work from their system libraries.

- **switchr**?
-

Even if the analysis is done entirely in R, and we can ensure the same package versions are used, it doesn't guarantee the analysis can be replicated on two different operating systems.

3.5 Virtual Machines

3.6 Curating Open Data

3.6.1 The Rise of Open Data

The World Wide Web brought about exciting new opportunities to publicly share information and data. Not all data on the Web is publicly available and free to use, but the data that is, is commonly referred to as "Open Data". Open data can be a controversial topic, but it can also be constructive, especially for the academic community, where verified conclusions maintain integrity and enable progress of the discipline. In fact, in recent years, we've seen several high profile works refuted, after they were published, when errors in their data analysis were identified (see (Baggerly and Coombes 2009); (Herndon, Ash, and Pollin 2014)). Open data will not, in itself, prevent wrong conclusions from being made, but it does gives us a greater potential to identify them.

It has been shown that researchers are generally willing to share their data, but the movement has been slow largely due to a lack of "systems that make it quick and easy to share data" (Tenopir et al. 2011); (Pampel 2013). Fortunately, there are a number of efforts to reduce this burden and make research data repositories more visible, usable, and worthwhile (Pampel et al. 2013); (King 2007). These efforts would certainly make the discovery and acquisition of open data much easier – a problem that is commonly overlooked, and has the potential of preventing analysis altogether.

(Stuart Dillon 2013) proposed a general search engine for discovering and acquiring data on the Web, but assumes data source(s) have been pre-identified and neatly organized into a relational form by domain experts. This approach might work in a perfect world where volunteers are abundant and publishers are aware of the analysis others may want to perform on the data, but it generally doesn't work for a Web that is constantly expanding, evolving, and increasing in complexity.

The applications of open data are not restricted in any way research experiments or academic research. From Wikipedia articles to Government records, the Web hosts the largest and most diverse set of publicly available data. Unfortunately, this data is often difficult to acquire and/or embedded within unstructured documents, making it difficult to incorporate into a data analysis workflow. This is especially true of data used to inform downstream Web applications. Thankfully, many skilled programmers spend many hours building a wide variety of tools to provide more convenient access to open data. A categorization of freely available tools designed to solve these problems is presented in Working with Open Data in `R`.

3.6.2 What Makes Data on the Web "Open"?

The ability to access and acquire data on the Web does not grant one permission to use it in whatever way they want. If the data is published without a license, it is owned by the publisher under copyright, and one must ask permission to use the data for their intended purpose. If the data is published with a license, that license will dictate the terms of use. Truly open data is published with a license, dedicating it to the public domain which waives ownership of copyright. Even if the license does not waive copyright, it will typically allow for individuals to use the data for non-commercial purposes.

3.6.3 On the Quality, Quantity, and Accessibility of Open Data

> "The Web is rarely perfectly honest, complete, and unbiased; but it's still pretty damn useful." - (Swartz 2013)

Extracting data placed within HTML `<table>` tags is often trivial, but they can contain uninteresting information from a data analysis perspective. In 2008, (Cafarella et al. 2008) estimated that 154 million HTML tables (out of the 14.1 billion considered) contained high quality relational data (TODO: what exactly do they mean by high quality?). Other studies have estimated the rate of "genuine" HTML tables to be around 15.2 percent (Y. Wang and Hu 2002).

Genuine structured data certainly exists outside of HTML tables, sometimes in the form of unstructured text or lists, making it difficult to automatically detect and acquire. There are a number of algorithms for automatic acquisition of data on the Web (Crescenzi, Mecca, and Merialdo 2001);(Ortona et al. 2015). We've also seen a number of free and paid Web services such as http://import.io and http://enigma.io implement such algorithms and as well as add crowd-sourcing features. These automated approaches typically work well in "nice" cases where HTML pages are static, sensibly structured, and consist mostly of data. In practice, these assumptions typically don't hold, and a extraction rule specific to the data source is required. An overview of tools for writing such wrappers in the `R` language is presented in Extracting Open Data with R.

Assuming that a data source has been identified and extracted, in many cases, that data has to be reshaped and/or cleaned so it's suitable for use in downstream statistical analysis. There are a number of interactive systems for performing such "data munging" tasks (see (Heer 2011); (Raman and Hellerstein 2001); (Verborgh and Wilde 2013)). These systems are especially helpful for discovering unknown problems with the data, but the data munging steps should eventually be programmed so they can be repeated faithfully and scale to a large number of tasks. An overview of tools for writing such wrappers in the `R` language is presented in Munging Open Data with R.

From here on, we refer to the cumulative process of identifying, extracting, munging, linking, and storing data on the Web as *curating* data. The `R` language, typically known for it's statistical modeling capabilities, provides a pragmatic set of tools for facilitating the data curation process and are covered in .

3.6.4 Best practices for publishing open data

3.6.5 Preserving Open Data

One valid concern when working with data on the Web is, "What if that resource goes missing?". This points out a major weakness in the design of HTTP, the transfer protocol of the Web. When we request a page on the Web, HTTP needs to look-up a specific file on a specific machine. If that file changes location, or if the Web server goes down for any reason, that file could be lost, in some cases, forever.

Of course, one can make copies of a known, existing resource(s) to keep backup(s) of data. For instance, one could run `rsync` (or similar) to keep local files in sync with files on another machine. However, `rsync` doesn't provide a mechanism for curating data from unstructured files, version control, or collaborating with others. These problems are currently being addressed by the open source project `dat` which borrow some algorithm design ideas from the revision control system `git`, but optimizes them to tabular data rather than source code. These tools provide a decent approach to preserving open data assuming their resource locations are known before being removed.

The Internet Archive's Wayback Machine https://archive.org/web is a gigantic effort to archive the Web. In October 2012, its archives topped 10 petabytes of data (Brown 2006), and three years later, a reported 439 billion web pages are available. A number of interesting projects are aimed at discovering and searching content in this archive (Lin, Gholami, and Rao 2014). Although a step in the right direction, the Internet Archive currently adheres to the Robots Exclusion Protocol, so sites requesting Web Robots to ignore their content will not be archived.

IPFS is a proposed alternative to HTTP meant to address it's weaknesses (Benet 2014). IPFS is content addressable, meaning that requests reference the actual content rather than a specific file on a specific machine. IPFS is also de-centralized, meaning that as long as one machine on the network has the content, it can be requested anyone on the network. Although IPFS implements many great ideas, given the massive amount of technology built on top of HTTP, it's hard to imagine that it will ever go away completely. Thus, if we want to curate open data in order to facilitate more complex data analysis workflows, we need better software tools to do so.

3.6.6 R as a Data Curation Engine

Building off the work of (Chambers 1999) and (Veillard 2006), the R Development Core Team included a number of convenient options within base R to download, and in some cases parse, files via HTTP/FTP. Assuming files are in plain text format, and contain properly formatted, tabular data; high-level functions such as `read.table()` and `read.csv()` can load data into R directly from a Uniform Resource Identifier (URI). This is a strong file format assumption, but there are a number of R packages aimed at input/output for different formats (R Core Team 2015); (Wickham and Miller 2015). (C.-h. Chan, Chan, and Leeper 2015) combines the capabilities of many of these packages into standard interface.

3.7 Interactive and Dynamic Statistical Graphics for Data Analysis on the Web

4 Chapter 2: Tools for Curating Open Data with R

If data isn't conveniently accessible as a tabular text file (such as csv or tsv), working with open data typically requires some knowledge of a constantly evolving set of network protocols and Web technologies. This presents a barrier to access for many researchers, but there is a large effort to lower the barrier, especially for the R project. The Web Technologies and Service CRAN Task View does a great job listing all of these efforts for the R project (Scott Chamberlain). Most of these tools can be grouped into one of three categories:

1. A high level interface with direct access to data ready for statistical analysis. In this case, the interface is typically restricted to a single data source, but users can obtain tidy data (Wickham 2014) without any knowledge of protocols or Web Technologies. Some interfaces, such as **pitchRx** (Sievert 2014), perform all the steps of data curation under the hood and require a tremendous amount of work by the author. Other interfaces may simply wrap an existing web API for accessing already tidy data.

2. A grammar for transforming non-tidy information into a tidy form. In this case, the tool is typically restricted to a specific file format such as HTML, XML or JSON. However, in some cases, it can remove any requirements/skills required for transforming these file formats. For example, **XML2R** (Sievert 2014) makes it possible to transform XML into a tidy form without XPATH and can make it easier build and maintain interfaces that fall under (1) (such as **pitchRx** and **bbscrapeR**).

3. A low level interface for working with network protocols and Web Technologies. Using these interfaces require an understanding of popular network protocols such as HTTP/HTTPS, data formats such as JSON and XML, and Web technologies such as HTML/JavaScript/etc. Tools that fall under (1) and (2) build on top of tools under (3).

This write-up focuses on how to build tools under (1) and (2) using tools under (3)?

Some of the lower-level tools require knowledge of technologies such as HTTP

A couple projects, such as **XML2R** (Sievert 2014) or **tidyjson** (needs citation) provide a high-level grammar for transforming . A more common, but less generic, approach is to provide a direct interface to specific data source(s) – one such example is . In some cases, this approach isn't ideal, since it may require assumptions about the analysis to be performed.

Better tools might also encourage statisticians to become more hands-on during the acquisition, transformation, and cleaning of data. This is important for reasons similar to why it's important to involve a statistician in the design of an experiment. A number of decisions made during data curation can effect data quality. In fact, it may be that a very large portion of the available data is bad, irrelevant to the analysis, and/or simply too large to preserve in bulk. Having a solid foundation in statistical modeling certainly helps in resolving these issues and can lead to more accurate conclusions.

4.0.0.1 Packaging Access to Specific Data Sources

`R` users are probably familiar with the **datasets** package (distributed with R itself) which provides direct access to classic datasets such as `iris` or `mtcars`. `R`'s packaging mechanism provides a very convenient way to distribute datasets, but it does come with some limitations, which presents problems for packaging open data sources:

1. CRAN imposes size restrictions on the size of the package.
2. CRAN imposes limitations on the frequency of package updates.
3. Some open data sources have terms of use which include restrictions on redistributing the data itself.

To avoid these problems, we've seen a number of `R` packages which provide function(s) to extract open data from it's source, clean and transform it into a usable form. One example of this approach is the **pitchRx** package where users just have to provide a range of dates to acquire data:

```
library(pitchRx)
dat <- scrape(start = "2008-01-01", end = Sys.Date())
```

Since the data source contains a number of observational units, the `scrape()` function returns a list of data frames (one data frame for each unit). These can easily be exported as database tables using a database of the users choice via R's database interface (Databases 2014).

```
# start a database connection
con <- DBI::dbConnect(RSQLite::SQLite(), "mydb.sqlite3")
# extract, transform, and load into a database
scrape(start = "2008-01-01", end = Sys.Date(), con = con)
```

It's worth noting that the three main jobs of the `scrape()` function is to extract, transform, and (optionally) load data into a database. These are three general operations that similar packages probably want to perform. Although the extract and transform steps will vary dramatically from package to package, the **etl** package provides a common interface to promote consistency across packages (Baumer and Sievert).

4.0.0.2 Extract, Transform, and Load Paradigm

> It is all too easy for statistical thinking to get swamped by programming tasks. (B. D. Ripley and Ripley 2001)

etl is to **bbscrapeR** as **DBI** is to **RSQLite** in the sense that **etl** and **DBI** provide a set of generic functions while **bbscrapeR** and **RSQLite** implement methods specific to an application. In other words, **bbscrapeR** implements methods for **etl** generics that extract, transform, and load a *specific* (basketball) data source and **RSQLite** implements methods for **DBI** generics that allow one to communicate with a MySQL database from `R`.

4.0.0.3 Working with Popular Web Data Formats

Although many formats exist, the majority of data transferred over the web comes in two forms: XML and JSON (needs citation). These formats are designed to be machine readable. That is, given a set of directions, a computer can store and parse information within these files. These formats are great for use in web applications where machines communicate and transfer data. In many cases, this data has valuable information that *humans* want to analyze. Before that can happen, however; analysts typically have to request and transform HTTP responses into forms suitable for analysis and/or visualization.

4.0.0.4 Requesting Web Content {#sec: request}

Before transforming XML/JSON data structures, one typically requests content from a web server via a communication protocol. The most ubiquitous protocol is the Hypertext Transfer Protocol (HTTP). Base R has built-in utilities for reading data (or `GET`ting) via HTTP, but this use case is quite limited. For example, if the analyst wishes to obtain data over a secure connection, such as HTTP Secure (HTTPS), other methods must be used.

cURL is a widely used command line tool which covers many protocols for transferring data between machines. The RCurl package provides a low-level interface to cURL with some additional tools for processing (Lang 2014)

for transferring data between machines using Uniform Resource Locators (URLs) and .

- RCurl
- httr
- does XML/rvest fit here?

4.0.0.5 Transforming Web Content into Structured Data {#sec: transform}

Working directly with XML/JSON presents challenges for data analysis and statistical modeling. The XML/JSON specifications allow for deeply nested and non-relational data structures; however, popular statistical computing software assumes data exists in a tabular format – each row represents the observational unit and each column represents attributes associated with each observation (Wickham 2014). A number of efforts exist for working with XML/JSON, but in many cases, there is no standard or well-defined way to transform these data structures into a tabular/tidy format. As a result, the analyst is left to handle reshaping of the data into a usable format for data analysis.

- introduce tidy data framework?
- XML2R
- Pros: Easy to *express* how to go from "unstructured" XML to tidy data.
- Cons: Must be able to pull XML into memory (consequence of the implementation, not of the concept itself)
- Future work: C++ backend to do dplyr-esque lazy computations? (TODO: look at connecting to a XML database)

4.0.0.6 Cleaning data for statistical analysis

- https://github.com/data-cleaning/editrules
- https://github.com/data-cleaning/validate

4.0.0.7 Working with databases

- DBI
- dplyr

==========

The CRAN task view on Open Data does a great job of summarizing of `R` clients which connect to these and other open data efforts.

Prerequisites to performing analysis on this data might require the discovery, parsing, cleaning, transformation, and integration of data with other sources.

A number of disciplines have produced substantial works aimed at simplifying various stages of the data curation process. Although some tasks can be automated or guided by clever systems, at some level, human intervention is required during the curation process. Curation typically requires a custom set of instructions for extracting and structuring information for downstream analysis. The domain knowledge required to write these instructions varies wildly depending on the application.

The contribution of this work is an overview of modern tools for curating data from the Web using R. An emphasis is placed on problems that data-driven researchers are likely to face and solutions that minimize the amount of prior knowledge and cognitive effort required. The goal is provide a taxonomy of practical problems and solutions that the community finds useful.

5 Chapter 3: Interactive Web Graphics for Data Analysis

Baggerly, Keith A, and Kevin R Coombes. 2009. "Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology." *The Annals of Applied Statistics* 3 (4): 1309–34.

Baumer, Ben, and Carson Sievert. *Etl: Extract-Transfer-Load Framework for Medium Data*. http://github.com/beanumber/etl.

Benet, Juan. 2014. "IPFS - Content Addressed, Versioned, P2P File System." *ArXiv.org*, July, 1–11. http://arxiv.org/abs/1407.3561.

Broman, Karl. 2015. "**Reproducibility Is Hard**." https://kbroman.wordpress.com/2015/09/09/reproducibility-is-hard/.

Brown, Adrian. 2006. *Archiving Websites: A Practical Guide for Information Management Professionals*. Facet Publishing.

Cafarella, Michael J, Alon Halevy, Zhe Daisy Wang, Eugene Wu, and Yang Zhang. 2008. "WebTables: Exploring the Power of Tables on the Web." *VLDB*, March, 1–12.

Chambers, John. 1999. *Programming with Data*. Springer Verlag.

Chan, Chung-hong, Geoffrey CH Chan, and Thomas J. Leeper. 2015. *Rio: A Swiss-Army Knife for Data File I/O*.

Crescenzi, Valter, Giansalvatore Mecca, and Paolo Merialdo. 2001. "RoadRunner: Towards Automatic Data Extraction from Large Web Sites." *Proceedings of the Th VLDB Conference*, June, 1–10.

Databases, R Special Interest Group on. 2014. *DBI: R Database Interface*. http://CRAN.R-project.org/package=DBI.

Drummond, Chris. 2009. "Replicability is not Reproducibility: Nor is it Good Science." *Proceedings of the Evaluation Methods for Machine Learning Workshop at the Th ICML*, April, 1–4.

Gentleman, Robert, and Duncan Temple Lang. 2004. "Statistical Analyses and Reproducible Research." *Bioconductor Project Working Papers*, November, 1–38.

Heer, Sean Kandel AND Andreas Paepcke AND Joseph Hellerstein AND Jeffrey. 2011. "Wrangler: Interactive Visual Specification of Data Transformation Scripts." In *ACM Human Factors in Computing Systems (CHI)*. http://vis.stanford.edu/papers/wrangler.

Herndon, Thomas, Michael Ash, and Robert Pollin. 2014. "Does High Public Debt Consistently Stifle Economic Growth? A Critique of Reinhart and Rogoff." *Cambridge Journal of Economics* 38 (2): 257–79. doi:10.1093/cje/bet075.

King, G. 2007. "An Introduction to the Dataverse Network as an Infrastructure for Data Sharing." *Sociological Methods & Research* 36 (2): 173–99.

Knuth, Donald. 1992. "Literate Programming." *Number 27 in CSLI Lecture Notes. Center for the Study of Language and Information*, August, 1–15.

Lang, Duncan Temple. 2014. *RCurl: General Network (HTTP/FTP/.) Client Interface for R*. http://CRAN.R-project.org/package=RCurl.

Lang, Duncan Temple, and the CRAN Team. 2015. *XML: Tools for Parsing and Generating XML Within R and S-Plus*. http://CRAN.R-project.org/package=XML.

Leisch, Friedrich. 2002. "Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis." *Compstat Proceed- Ings in Computational Statistics*, June, 1–7.

Lin, Jimmy, Milad Gholami, and Jinfeng Rao. 2014. "Infrastructure for Supporting Exploration and Discovery in Web Archives." *Proceedings of the 23rd International World Wide Web Conference Companion*, February, 1–5.

Ortona, Stefano, Giorgio Orsi, Marcello Buoncristiano, and Tim Furche. 2015. "WADaR: Joint Wrapper and Data Repair." *Proceedings of the VLDB Endowment* 8 (June): 1–4.

Pampel, Heinz. 2013. "How to Find an Appropriate Research Data Repository?" Blog. *PLOS*. http://blogs.plos.org/tech/how-to-find-an-appropriate-research-data-repository/.

Pampel, Heinz, Paul Vierkant, Frank Scholze, Roland Bertelmann, Maxi Kindling, Jens Klump, Hans-JÃ rgen Goebelbecker, Jens Gundlach, Peter Schirmbacher, and Uwe Dierolf. 2013. "Making Research Data Repositories Visible: The re3data.org Registry." *PLoS ONE* 8 (11): e78080–10.

R Core Team. 2015. *Foreign: Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, DBase, .* http://CRAN.R-project.org/package=foreign.

Raman, Vijayshankar, and Joseph Hellerstein. 2001. "Potters Wheel: An Interactive Data Cleaning System." *Proceedings of the Th VLDB Conference*, December, 1–10.

Ripley, B D, and R M Ripley. 2001. "Applications of R Clients and Servers." *Proceedings of the Nd International Workshop on Distributed Statistic Al Computing*, March, 1–7.

Rossini, A J. 2001. "Literate Statistical Practice." *Proceedings of the 2nd International Workshop on Distributed Statistical Computing*, March, 1–10.

Scott Chamberlain, Patrick Mair, Thomas Leeper. "CRAN Task View: Web Technologies and Services." http://cran.r-project.org/web/views/WebTechnologies.html.

Sievert, Carson. 2014. "Taming PITCHf/x Data with pitchRx and XML2R." *The R Journal* 6 (1). http://journal.r-project.org/archive/2014-1/sievert.pdf.

———. 2015. *Rdom: Access the DOM of a Webpage as HTML Using Phantomjs*. https://github.com/cpsievert/rdom.

Stodden, Victoria. 2010. *The Scientific Method in Practice: Reproducibility in the Computational Sciences*.

Stuart Dillon, Gottfried Vossen, Florian Stahl. 2013. "Towards the Web in Your Pocket: Curated Data as a Service." In *Advanced Methods for Computational Collective Intelligence*, edited by Radosław Katarzyniak Ngoc Thanh Nguyen Bogdan Trawiński. Springer Berlin Heidelberg.

Swartz, Aaron. 2013. "Aaron Swartz's A Programmable Web: An Unfinished Work." In *Synthesis Lectures on the Semantic Web Theory and Technology*, edited by Ying Ding and James Hendler, 1–66.

Tenopir, Carol, Suzie Allard, Kimberly Douglass, Arsev Umur Aydinoglu, Lei Wu, Eleanor Read, Maribeth Manoff, and Mike Frame. 2011. "Data Sharing by Scientists: Practices and Perceptions." *PLoS ONE* 6 (6): e21101–21.

Veillard, Daniel. 2006. "Libxml: The XML c Parser and Toolkit of Gnome Parsing." http://www.xmlsoft.org.

Verborgh, Ruben, and Max De Wilde. 2013. *Using OpenRefine*. PACKT Publishing. https://www.packtpub.com/big-data-and-business-intelligence/using-openrefine.

Wang, Yalin, and Jianying Hu. 2002. "A Machine Learning Based Approach for Table Detection on The Web." *WWW*, May, 1–9.

Wickham, Hadley. 2014. "Tidy Data." *The Journal of Statistical Software* 59 (10). http://www.jstatsoft.org/v59/i10/.

Wickham, Hadley, and Evan Miller. 2015. *Haven: Import SPSS, Stata and SAS Files*. http://CRAN.R-project.org/package=haven.