Web Technologies for Computing with Data

Carson Sievert

2015-09-08

Contents

1 Chapter 1: Literature Review

> "[The Web] has helped broaden the focus of statistics from the modeling stage to all stages of data science: finding relevant data, accessing data, reading and transforming data, visualizing the data in rich ways, modeling, and presenting the results and conclusions with compelling, interactive displays." - (Nolan and Lang 2014)

1.1 Curating Open Data

1.1.1 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, 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 workflow 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 social media activity, the Web provides 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`.

1.1.2 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 acquistion 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 faciliting the data curation process.

1.1.3 R as a Data Curation Engine

1.1.3.1 Using R's built-in facilities (e.g. `read.table()`)

Building off the work of (Chambers 1999) and (Veillard 2006), the R Development Core Team included a number of convenient options within base R for downloading web documents via HTTP/FTP.

This development continues to be valuable for a number of reasons. 1. Analyses are more accessible/portable (don't have to send data/scripts directly to each user) 2.

1.1.3.2 Helpful add-on packages for HTTPS

- **curl**
- **downloader**

1.1.3.3 Custom client interfaces to relational data

- XML::readHTMLTable()
- rvest::html_table()
- pitchRx::scrape()
- bbscrapeR::rebound()

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.

1.1.3.4 Packaging Access to Data for Statistical Analysis

`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).

1.1.3.5 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`.

1.2 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.

1.3 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?

1.4 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)

1.5 Cleaning data for statistical analysis

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

1.6 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.

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.

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.

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.

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.

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

Nolan, Deborah, and Duncan Temple Lang. 2014. *XML and Web Technologies for Data Sciences with R*. Edited by Robert Gentleman Kurt Hornik and Giovanni Parmigiani. Springer.

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-JÃrgen Goebelbecker, Jens Gundlach, Peter Schirmbacher, and Uwe Dierolf. 2013. "Making Research Data Repositories Visible: The re3data.org Registry." *PLoS ONE* 8 (11): e78080–10.

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.

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.

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/.