Curating Data From the Web Using R

Carson Sievert

Preface

There is no question the World Wide Web has profoundly impacted the way we perform, present, and think about data analysis. As (Nolan and Lang 2014) put it:

> "[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."

The Web revolutionized the way we find, and consequently work with, data. Nowadays anyone can start with a problem, find relevant data, and perform an analysis to provide insight. Although there is new opportunity for finding data, acquiring and transforming it into a form that is suitable for statistical analysis remains a huge challenge. Some have estimated that 80 percent of the data science workflow is spent acquiring, transforming, and cleaning data (from this point we refer to this set of tasks as data *curation*) leaving only 20 percent left for statistical analysis (Lohr). By providing better tools for data curation, we can spend more time performing analysis and reporting results.

If data isn't conveniently accessed as a tabular text file (such as csv or tsv), curation typically requires some knowledge of a constantly evolving set of network protocols (for accessing) and Web technologies (for transforming). 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 (Scott Chamberlain). Most of these tools can be grouped into one of three categories:

1. A high level interface for directly accessing curated data. 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 or 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.

In addition to data discovery, the Web browser provides an exciting platform for presenting data analysis. Reactive documents enable readers to interactively explore complex findings and question the authors assumptions.

The following sections provide an overview of concepts and implementations of statistical software aimed at various stages of the data science workflow. Since modeling is very, very large area, we cover a only a few types of models that can scale to large datasets.

Acquiring and Transforming Data on the Web using R

R's built-in facilities for acquiring data

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.

Popular formats for data on the Web

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.

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?

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)

Directly "Requesting" Structured Data

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

Interactive Web Documents

- RServe
- FastRWeb
- opencpu
- shiny

Web-based Interactive Statistical Graphics for Data Analysis

History of Interactive Statistical Graphics

Over the course of several decades, many advances in the area of dynamic and interactive statistical graphics were made. Most of this work was done before the web-browser became a promising platform for interactive graphics.

Modern Web-based Interactive Statistical Graphics

- D3js
- Vega
- ggvis
- htmlwidgets
- rCharts

- animint
- Tableau
- LDAvis

shiny versus animint

- The main benefit of having an R Web server back-end for Web-based visualizations is the ability to performs statistical computations on-the-fly (tour example?).

Enabling Reproducibility

Continuous Integration

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

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

Lohr, Steve. "For Big-Data Scientists, 'Janitor Work' Is Key Hurdle to Insights." http://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights-is-janitor-work.html?_r=1.

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.

Scott Chamberlain, Patrick Mair, Thomas Leeper. http://cran.r-project.org/web/views/WebTechnologies.html.

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

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

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