Web Technologies Task View

by Patrick Mair and Scott Chamberlain

Abstract This article presents the CRAN Task View on Web Technologies. We describe the most important aspects of Web Technologies and Web Scraping and list some of the packages that are currently available on CRAN. Finally, we plot the network of Web Technology related package dependencies.

Introduction

In modern Statistics and Data Science, Web technologies have become an essential tool for accessing, collecting, and organizing data from the Web. Well known companies, including Google, Amazon, Wikipedia, and Facebook, and increasingly all data providers, are providing APIs (Application Programming Interfaces) that include specifications for variables, data structures, and object classes which facilitate the interaction with other software components such as R.

Data collection from the Web can be performed at several levels: From low-level HTML/XML/JSON parsing at the one end up to calling simple functions implemented by numerous convenience packages (which, e.g., make use of API specifications internally) on the other end. A comprehensive description of Web technology strategies can be found in Nolan and Temple Lang (2014).

Within the R community, projects like *omegahat* (`http://www.omegahat.org/`) or rOpenSci (`http://ropensci.org/` where people can contribute through `https://github.com/ropensci/webservices`) made an effort to provide infrastructures that allow R to interact with the Web. Recently, a vast amount of corresponding Web technology packages have been developed which are now collected, organized, and structured in a *Web Technology CRAN Task View* available at `http://cran.r-project.org/web/views/WebTechnologies.html`. By setting up this task view, which we present in this paper and which will be updated regularly, we hope that this facilitates useRs to get an overview of what is available and, at the same time, it motivates future use and development of R packages that interact with the Web.

Note that not all packages listed in the Task View are on CRAN. Some of them are in the omegahat repository, and others on GitHub `https://github.com/` or R-Forge `https://r-forge.r-project.org/`. The corresponding package links can be found in the links section of the Task View.

Low level Web scraping

Web scraping refers to the process of extracting information from websites. In low-level scraping tasks, the two basic concepts are *retrieving* and *parsing*. The core packages in R that allow for either retrieving or parsing (or both) are **XML** (both), **RCurl** (retrieving), and **rjson**/**RJSONIO**/**jsonlite** (parsing). The XML package contains functions for parsing XML and HTML, and supports XPath for searching XML. A helpful function in **XML** to extract data from one or more HTML tables is `readHTMLTable()`. The **RCurl** package makes use of the `libcurl` library for transferring data using various protocols and provides a low level curl wrapper that allows one to compose general HTTP requests and provides convenient functions to fetch URLs, get/post forms, etc. and process the results returned by the Web server. This provides a great deal of control over the HTTP/FTP connection and the form of the request while providing a higher-level interface than is available just using R socket connections. It also provides tools for Web authentication. Within this context, the **httr** package (a higher level wrapper around **RCurl**) provides easy-to-use functions. JSON stands for JavaScript Object Notation and is a data interchange format becoming the most common data format on the web. The **rjson**, **RJSONIO**, and **jsonlite** packages convert R objects into JSON objects and vice-versa.

Authentication

Using Web resources can require authentication, either via API keys, OAuth, `username:password` combination, or via other means. Additionally, sometimes Web resources require authentication to be in the header of an http call, which requires a little bit of extra work. API keys and `username:password` can be combined within a URL for a call to a web resource (api key: `http://api.foo.org/?key=yourkey`; `user/pass: http://username:password@api.foo.org`), or can be specified via commands in **RCurl** or **httr**. OAuth is the most complicated authentication process, and can be most easily done using **httr** which includes demos for OAuth 1.0 (linkedIn, Twitter, Vimeo) and demos for OAuth 2.0 (Facebook, GitHub, Google). The **ROAuth** package provides a separate R interface to OAuth.

Javascript

Javascript has become a dominant programming language for building web applications, including for response web site elements, as well as maps. An increasing number of R packages are bringing the power of Javascript to useRs. In our Task View we highlight many of these. One of these packages, not on CRAN yet, is **rCharts**, which wraps a suite of Javascript libraries for visualization. That is, **rCharts** lets you pass in typical R objects like data frames and lists and you get out Javascript visualizations in your browser (or in the "Viewer" in recent versions of RStudio's IDE). A framework RStudio has built, called **shiny**, combines Javascript with CSS, and HTML to make building web applications with R relatively painless.

Data repositories

Providing datasets in (Open Access) repositories in order to foster reproducibility and subsequent analyses on these data is a crucial component in modern science. The importance of reproducible research in various fields has been widely discussed in over the years (Gentleman and Temple Lang, 2004; Koenker and Zeileis, 2009; Peng, 2009; Pebesma et al., 2012). General Open Access repositories like the Dataverse network (`http://thedata.org/`) or PubMed related databases in biomedical literature (`http://www.ncbi.nlm.nih.gov/pubmed`) provide a highly important contributions to this field.

Recently, many R packages have been developed that interface Open Access data repositories using corresponding API specifications internally. In fact, this type of packages represents a large portion of all packages included in the Task View. Here, we just give a few examples from different areas. Within the area of Ecology and Evolutionary Biology we have packages such as **rgbif** and **rnbn** for biodiversity data and **rfishbase** and **rfisheries** for fishery data. In the area of Genomics **rsnps** and **rentrez** allow for the interaction with various SNP (Single-Nucleotide Polymorphism) repositories and NCBI (National Center for Biotechnology Information), respectively. In Earth Science, for example, implementations for downloading data from the Climate Reference Network (**crn** package) and obtaining data from National Centers for Environmental Prediction (**RNCEP** package).

In the Economics and Business area the **WDI** scrapes World Bank's world development indicators, in the Finance area packages like **TFX** connects to TrueFX for free streaming real-time and historical tick-by-tick market data for interbank foreign exchange rates, and in the Marketing area the **anametrix** package is bidirectional connector to the Anametrix API.

The **rpubchem** package interfaces the PubChem collection in the Chemistry area whereas **cimis** is a package for retrieving data from CIMIS, the California Irrigation Management Information System in the area of Agriculture. To quote an example from Sports, the **nhlscrapr** compiles the NHL (National Hockey League) real time scoring system database. More packages are listed in the Task View.

Collecting data from the Web

A core aspect of Web scraping is to harvest unstructured data, often texts, from the Web. The Internet provides massive amounts of text data from sources like blogs, online newspapers, social network platforms, etc. Especially in the areas like Social Sciences and Linguistics, this type of data provides a valuable resource for research. Once the data are harvested, the **tm** package offers many functionalities to handle text data in R.

Companies such as Google, Facebook, Twitter, or Amazon provide APIs which allow analysts to retrieve data. Several convenience packages have been implemented in R which facilitate the use of these API. With respect to Google we have **RGoogleStorage** which provides programmatic access to the Google Storage API, **RGoogleDocs** interfaces the Google Documents API, **translate** provides bindings for the Google Translate API, **scholar** provides functions to extract citation data from Google Scholar, **RGoogleTrends** gives programmatic access to Google Trends data, and **RgoogleMap** allows to import Google maps into R. In order to interact with Facebook, **Rfacebook** package provides an interface to the Facebook API whereas **twitteR** and **streamR** interact with Twitter. Regarding Amazon services, the **AWS.tools** package provides access to Amazon Web Services (EC2/S3) and **MTurkR** gives access to the Amazon Mechanical Turk Requester API.

In terms of newspapers, the **GuardianR** package provides an interface to the Open Platform's Content API of the Guardian Media Group whereas **RNYTimes** interfaces to several of the New York Times Web services for searching articles, meta-data, user-generated content, and best seller lists. Again, the packages we list here represent only a small portion of packages contained in the Task View.

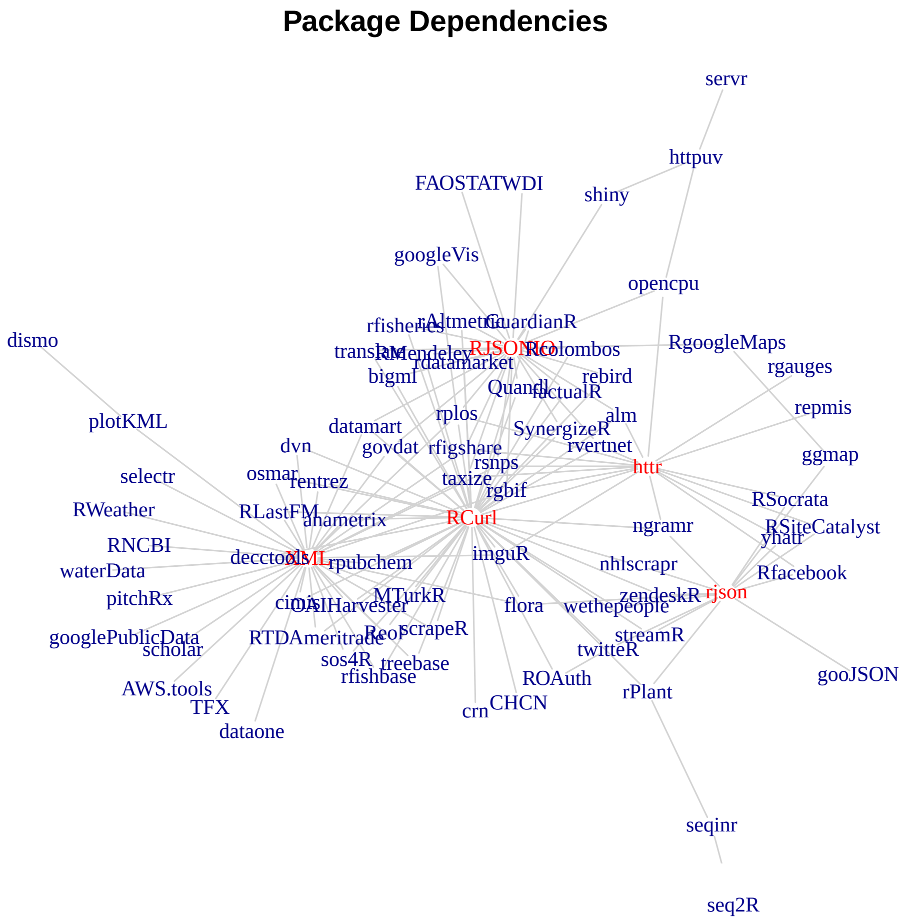

Figure 1: A network plot that shows the package dependencies and imports (only packages listed in the Web Technologies Task View are considered).

Conclusion

In this article we presented the Web Technologies Task View which is now online on CRAN. In order to give a final package overview, the network plot in Figure 1 represents all CRAN packages from the Task View including the dependency structure. The network plot, produced using **igraph**, is based on a scraping job where we harvested all corresponding package dependencies (and imports) from CRAN. Therefore, this plot does not include Task View packages that are only hosted on GitHub and Omegahat. The edges in Figure 1 reflect dependencies between packages that are listed in the Task View. We see that the low level packages such as **XML**, **RCurl**, **rjson**, **RJSONIO**, and **httr** are the core packages whose functionalities are used by many high level scraping packages. The Web Technologies Task View will be updated on a regular basis, and therefore, the network plot will change accordingly. The R code for producing the plot can be found at `https://gist.github.com/sckott/7831696`.

Bibliography

R. Gentleman and D. Temple Lang. Statistical analyses and reproducible research. Technical Report 2, Bioconductor Project Working Papers, 2004. URL `http://biostats.bepress.com/bioconductor/paper2/`. [p179]

R. Koenker and A. Zeileis. On reproducible econometric research. *Journal of Applied Econometrics*, 24(5): 833–847, 2009. [p179]

D. Nolan and D. Temple Lang. *XML and Web Technologies for Data Sciences with R*. Springer, New York, 2014. [p178]

E. Pebesma, D. D. Nüst, and R. Bivand. The R software environment in reproducible geoscientific research. *Eos Transactions American Geophysical Union*, 93(16):163–163, 2012. [p179]

R. D. Peng. Reproducible research and Biostatistics. *Biostatistics*, 10(3):405–408, 2009. [p179]

Patrick Mair
Department of Psychology
Harvard University
USA mair@fas.harvard.edu

Scott Chamberlain
Biology Department
Simon Fraser University
Canada scott@ropensci.org