# Web Scraping in R: A Primer

*Noah Rubin (nar62)*

*October 26, 2015*

This tutorial, as the title above suggests, is a primer on web scraping in R using the *rvest* (https://github.com/hadley/rvest) library from Hadley Wickham. This library allows one to easily scrape websites by tag, class, id, and attributes. If you have ever used the Beautiful Soup (http://www.crummy.com/software/BeautifulSoup/) library for python, rvest was created with that library in mind.

If you do not have R installed on your machine, go here (https://cran.r-project.org/) to download the proper type for your OS. **Note, I wrote this tutorial using R version 3.1.3 (2015-03-09), A.K.A. “Smooth Sidewalk”, and so you may experience issues with packages and such if you are using a different version. If you would like to attempt to manage and use multiple versions of R, you may find this resource (https://support.rstudio.com/hc/en-us/articles/200486138-Using-Different-Versions-of-R) to be helpful.

Once you have R installed, I would also suggest installing RStudio (https://www.rstudio.com/products/rstudio/download/), which at the time of this writing is at version 0.99.489. Aside from being a fairly decent IDE for R, it comes with some kickass features, like R Markdown (the thing I used to create this tutorial). For more information on how to create awesome documents with R Markdown, go here (http://rmarkdown.rstudio.com/).

The packages used in this tutorial are:

1. rvest (https://github.com/hadley/rvest)
   Web scraping library, explained in some detail later on

2. ggmap (https://cran.r-project.org/web/packages/ggmap/index.html)
   Amongst other things, contains a ‘geocode’ method that returns latitude and longitude coordinates for a given location (in this tutorial, locations are given as strings of names of places)

3. purrr (https://github.com/hadley/purrr)
   Definitely a must-have library if you’re into functional-style programming. Contains all sorts of mapping and other functional methods, in the spirit of the javascript libarary underscore.js (http://underscorejs.org/)

I would also be doing you a disservice if I failed to mention the devtools (https://github.com/hadley/devtools) library, also by Hadley Wickham. At this point you may have realized, Hadley is the man (and also chief scientist at RStudio). Devtools is a development library that will make your life a lot easier if you begin to use R more heavily. You can install it by using the command ‘install.packages(“devtools”)’, which is one way to install any package available through the CRAN project. However, once you download devtools, and you may have noticed already, perusing the website for rvest and purrr, that you can also use the command ‘devtools::install(“package_name”)’, or to install a package from github, ‘devtools::install_github(“package_name”)’.

To illustrate the simplicity of scraping with rvest, I have chosen the task of extracting dining hall names and locations from the Cornell dining website, as well as the location API provided through ggmap (although you’ll see the issue with relying on ggmap later). The general task for this case study is to get the

names of dining halls on Cornell's campus, then get their latitudes and longitudes. Let's dive in!

```
# xml2 is a dependency for rvest
library(xml2);
library(rvest);
# ggplot 2 is dependency of ggmap
library(ggplot2);
library(ggmap);
library(purrr);

# set the url to be parsed
namesURL = "https://living.sas.cornell.edu/dine/wheretoeat/";
# get the HTML, parse into DOM
labelsHTML = read_html(namesURL);
names(labelsHTML);
```

```
## [1] "node" "doc"
```

The most important parts of rvest are:

- Create an HTML document from a url, file on disk, or string HTML *read_html()*.
- Use css selectors on the document with the *html_nodes()* method. For example, html_nodes(".cls") will select all nodes with the class '.cls'. A useful resource for learning more about this method is the rvest github page, linked above. I also encourage you to experiment with it on your own!
- Extract components by tag using *html_tag()*, get all text within a tag with *html_text()*, and get the attributes with *html_attrs()*.

```
# get the link nodes (a tags)
linkNodes = labelsHTML %>%
  html_nodes("a");

# extract the links from the link nodes
links = linkNodes %>%
  html_attr("href");

# extract the dining location names
# from the link tags
labels = linkNodes %>%
  html_text();

# preview links and labels
head(links); head(labels);
```

```
## [1] NA                        "##content"
## [3] "http://www.cornell.edu/" "/living/sitesearch.cfm"
## [5] "/living/"                "/living/get_started/"
```

```
## [1] ""                    "Skip to main content" ""
## [4] "Site Search"         ""                     ""
```

Thus, we find that a lot of the links and labels are not worth keeping, so it's time to clean them up. First, however, we will combine them into a data frame for manipulation. Data frames are the way tables are represented in R, and can be thought of as a collection of vectors as columns with equal length (but often varying types). R makes data manipulation in data frames extremely easy. You can read more about data frames here (http://www.r-tutor.com/r-introduction/data-frame).

```
diningLocations = data.frame(labels = labels, links = links);
# view data frame
diningLocations[44:80,]
```

```
##                                    labels
## 44                              Trillium's
## 45                          Cornell Dairy Bar
## 46                         convenience stores
## 47                           Bear Necessities
## 48                            Jansen's Market
## 49                         Cornell Dining Now
## 50                           Bear Necessities
## 51                               Carol’s Café
## 52                     North Star Dining Room
## 53                               Risley Dining
## 54          Robert Purcell Marketplace Eatery
## 55                           Sweet Sensations
## 56                  Amit Bhatia's Libe Café
## 57                                Atrium Café
## 58                                 Bear's Den
## 59                               Big Red Barn
## 60                             Bus Stop Bagels
## 61                                Café Jennie
## 62                           Cornell Dairy Bar
## 63                                Green Dragon
## 64                                    Goldie's
## 65                                    Ivy Room
## 66                               Martha's Café
## 67                               Mattin's Café
## 68                                 Okenshields
## 69                                      Rusty's
## 70                               Synapsis Café
## 71                                    Trillium
## 72                  Becker House Dining Room
## 73                    Cook House Dining Room
## 74 Jansen’s Dining Room at Hans Bethe House
## 75                             Jansen’s Market
## 76                  Keeton House Dining Room
## 77                    Rose House Dining Room
## 78                                    104West
## 79                                what's open
## 80                               Order online
##
links
## 44                              /living/dine/wheretoeat/cafescoffeehouses/trilliu
m.cfm
## 45                     /living/dine/wheretoeat/cafescoffeehouses/cornelldairyba
r.cfm
## 46                                           /living/dine/wheretoeat/cstores/inde
x.cfm
## 47                                 /living/dine/wheretoeat/cstores/bearnecessitie
s.cfm
## 48                                   /living/dine/wheretoeat/cstores/jansensmarke
t.cfm
```

```
## 49                                                          http://now.dining.cornel
l.edu
## 50                              /living/dine/wheretoeat/cstores/bearnecessitie
s.cfm
## 51                           /living/dine/wheretoeat/cafescoffeehouses/carolscaf
e.cfm
## 52                             /living/dine/wheretoeat/AYCTEdiningrooms/northsta
r.cfm
## 53                              /living/dine/wheretoeat/AYCTEdiningrooms/risle
y.cfm
## 54       /living/dine/wheretoeat/AYCTEdiningrooms/robertpurcellmarketplaceeater
y.cfm
## 55                        /living/dine/wheretoeat/cafescoffeehouses/sweetsensation
s.cfm
## 56                    /living/dine/wheretoeat/cafescoffeehouses/amitbhatialibecaf
e.cfm
## 57                           /living/dine/wheretoeat/cafescoffeehouses/atriumcaf
e.cfm
## 58                            /living/dine/wheretoeat/cafescoffeehouses/bears_de
n.cfm
## 59                           /living/dine/wheretoeat/cafescoffeehouses/bigredbar
n.cfm
## 60 http://living.sas.cornell.edu/dine/wheretoeat/cafescoffeehouses/busstopbagel
s.cfm
## 61                          /living/dine/wheretoeat/cafescoffeehouses/cafe-jenni
e.cfm
## 62                        /living/dine/wheretoeat/cafescoffeehouses/cornelldairyba
r.cfm
## 63                           /living/dine/wheretoeat/cafescoffeehouses/greendrago
n.cfm
## 64                           /living/dine/wheretoeat/cafescoffeehouses/goldiescaf
e.cfm
## 65                              /living/dine/wheretoeat/cafescoffeehouses/ivyroo
m.cfm
## 66                           /living/dine/wheretoeat/cafescoffeehouses/marthascaf
e.cfm
## 67                           /living/dine/wheretoeat/cafescoffeehouses/mattinscaf
e.cfm
## 68                            /living/dine/wheretoeat/AYCTEdiningrooms/okenshield
s.cfm
## 69                               /living/dine/wheretoeat/cafescoffeehouses/rusty
s.cfm
## 70                          /living/dine/wheretoeat/cafescoffeehouses/synapsiscaf
e.cfm
## 71                              /living/dine/wheretoeat/cafescoffeehouses/trilliu
m.cfm
## 72                        /living/dine/wheretoeat/AYCTEdiningrooms/beckerhousedinin
g.cfm
## 73                          /living/dine/wheretoeat/AYCTEdiningrooms/cookhousedinin
g.cfm
## 74                        /living/dine/wheretoeat/AYCTEdiningrooms/jansensdiningroo
```

```
m.cfm
## 75                                  /living/dine/wheretoeat/cstores/jansensmarke
t.cfm
## 76                      /living/dine/wheretoeat/AYCTEdiningrooms/keetonhousedinin
g.cfm
## 77                        /living/dine/wheretoeat/AYCTEdiningrooms/rosehousedinin
g.cfm
## 78                             /living/dine/wheretoeat/AYCTEdiningrooms/104westne
w.cfm
## 79                                                    /living/dine/whentoeat/inde
x.cfm
## 80                                       /living/dine/wheretoeat/orderahead/inde
x.cfm
```

As you can see from the frame above, the data.frame creates a data frame from the given columns. Note how I assigned names to each column, which is then the name used to access that column. The columns can be accessed by diningLocations$links or diningLocations$labels. Now I'm going to manually clean up the frame and only keep the relevant link:label pairs.

```
# only keep rows 45 to 78
diningLocations = diningLocations[45:78,];
# remove duplicates
diningLocations = diningLocations[!duplicated(diningLocations$labels),];
#remove unnecessary labels
exclude = c("convenience stores", "Cornell Dining Now");
diningLocations = diningLocations[-which(diningLocations$labels %in% exclude),];
# add domain to links
domain = "https://living.sas.cornell.edu";
diningLocations$links = paste(domain, diningLocations$links);
diningLocations$labels
```

```
##  [1] Cornell Dairy Bar
##  [2] Bear Necessities
##  [3] Jansen's Market
##  [4] Carol’s Café
##  [5] North Star Dining Room
##  [6] Risley Dining
##  [7] Robert Purcell Marketplace Eatery
##  [8] Sweet Sensations
##  [9] Amit Bhatia's Libe Café
## [10] Atrium Café
## [11] Bear's Den
## [12] Big Red Barn
## [13] Bus Stop Bagels
## [14] Café Jennie
## [15] Green Dragon
## [16] Goldie's
## [17] Ivy Room
## [18] Martha's Café
## [19] Mattin's Café
## [20] Okenshields
## [21] Rusty's
## [22] Synapsis Café
## [23] Trillium
## [24] Becker House Dining Room
## [25] Cook House Dining Room
## [26] Jansen’s Dining Room at Hans Bethe House
## [27] Jansen’s Market
## [28] Keeton House Dining Room
## [29] Rose House Dining Room
## [30] 104West
## 79 Levels:    ... Your Community
```

```
# Looks like the duplicate function missed Jansen's Market
diningLocations = diningLocations[-which(diningLocations$labels == "Jansen’s Marke
t"),];
```

Ok, for the cool part: getting latitude and longitude of each location. As a first pass, I’m going to use the ggmap library, created by David Kahle and Hadley Wickham. As mentioned above, this lib provides a geocode method that returns the latitude and longitude of a given location. The ‘map’ method is provided by the purrr library, also mentioned above. It is important to note that, unlike Java or Python, where imported methods are attached to the namespace of the imported module (unless you use from ____ import *), R dumps all methods into the same namespace (essentially). So, even though ‘map’ comes from the purrr, it can be used just by calling ‘map()’.

```
# locales stores the lat and longs given by geocode
locales = diningLocations$labels %>%
  # adding "Cornell University" for accuracy's sake
  paste(., " Cornell University", sep="") %>%
  # get lat and long
  map(geocode);
```

```
head(locales);
```

```
## [[1]]
##       lon      lat
## 1 -76.483 42.44702
##
## [[2]]
##       lon      lat
## 1 -76.483 42.44702
##
## [[3]]
##       lon      lat
## 1 -76.483 42.44702
##
## [[4]]
##       lon      lat
## 1 -76.483 42.44702
##
## [[5]]
##       lon      lat
## 1 -76.483 42.44702
##
## [[6]]
##       lon      lat
## 1 -76.483 42.44702
```

Well, ggmap (as well as really any other library in R) didn't perform very well for getting coordinates of very specific and unpopular destinations like dining halls on west campus at Cornell University. Alas, all that is left is to do change the information after the decimal points for lat and long, then we are done.

```
# separate latitudes and longitudes for altering
latitudes = locales %>%
  map(function(position) position$lat) %>%
  unlist;
longitudes = locales %>%
  map(function(position) position$lon) %>%
  unlist;

# construct latitude decimal point alterations
lat_alts = c(.4473495, .4423221, .4534478, .4534452, .4535259, .453041, .4559407,
.4535825, .4480805, .4457864, .4467172, .4484227, .4482542, .4466849, .4508134, .4
50099, .4467172, .4500624, .4443652, .446679, .4472154, .446891, .4478062, .448343
5, .4488146, .4472589, .4468473, .4479873, .4439353);
# construct longitude decimal point alterations
long_alts = c(.4732484, .4872978, .4910123, .4910123, .4783553, .4841109, .479658
1, .4783072, .4867747, .4854713, .4878249, .4832169, .4814839, .4863926, .4862022,
.4836112, .4878249, .4814158, .4845526, .4857208, .484334, .4799113, .4815657, .49
17249, .4916712, .4909148, .4916337, .4893484, .4897629);

# combine latitude and longitude corrections
latitudes = round(latitudes, 0) + lat_alts;
longitudes = round(longitudes, 0) - long_alts;
diningLocations$Latitude = latitudes;
diningLocations$Longitude = longitudes;

# fini! final data frame:
diningLocations[-2];
```

```
##                                      labels Latitude Longitude
## 45                        Cornell Dairy Bar 42.44735 -76.47325
## 47                         Bear Necessities 42.44232 -76.48730
## 48                          Jansen's Market 42.45345 -76.49101
## 51                             Carol’s Café 42.45345 -76.49101
## 52                   North Star Dining Room 42.45353 -76.47836
## 53                              Risley Dining 42.45304 -76.48411
## 54        Robert Purcell Marketplace Eatery 42.45594 -76.47966
## 55                          Sweet Sensations 42.45358 -76.47831
## 56                 Amit Bhatia's Libe Café  42.44808 -76.48677
## 57                               Atrium Café 42.44579 -76.48547
## 58                                Bear's Den 42.44672 -76.48782
## 59                              Big Red Barn 42.44842 -76.48322
## 60                           Bus Stop Bagels 42.44825 -76.48148
## 61                               Café Jennie 42.44668 -76.48639
## 63                              Green Dragon 42.45081 -76.48620
## 64                                  Goldie's 42.45010 -76.48361
## 65                                  Ivy Room 42.44672 -76.48782
## 66                             Martha's Café 42.45006 -76.48142
## 67                             Mattin's Café 42.44437 -76.48455
## 68                               Okenshields 42.44668 -76.48572
## 69                                   Rusty's 42.44722 -76.48433
## 70                             Synapsis Café 42.44689 -76.47991
## 71                                  Trillium 42.44781 -76.48157
## 72                 Becker House Dining Room 42.44834 -76.49172
## 73                   Cook House Dining Room 42.44881 -76.49167
## 74 Jansen’s Dining Room at Hans Bethe House 42.44726 -76.49091
## 76                Keeton House Dining Room  42.44685 -76.49163
## 77                  Rose House Dining Room  42.44799 -76.48935
## 78                                  104West 42.44394 -76.48976
```

And that’s it! If you want to write a file to a directory, follow these commands:
*1) setwd(“path/to/directory”);*
*2) write.csv(data_frame, file=“filename.csv”);*
where data_frame is the data frame object. This of course assumes you want to write to .csv file, but there are plenty of write methods depending on the type of file you want to create.