Lab #1: Web Scraping

Due in Sakai February 1, 2019 (by 10:00a)

Introduction

This lab is designed to give you hands-on experience with web scraping. You have three options: (1) scrape Twitter data, (2) scrape *New York Times* data, or (3) develop you own custom scraper using `rvest` to get Amazon product reviews.

Getting Started

First, set your working directory. Open up a fresh RStudio session (if you open RStudio and a previous work session is loaded, be sure you save any R and/or RData files before you close them out). Then set your working directory:

```r
setwd("working_directory_here")
```

Then open up a fresh R script. Save it using your first initial, full last name, and then "_lab1." So my script would be titled **MTaylor_lab1.R**.

You should always strive to keep your scripts tidy. At the top of your R script, type this (substituting in your first initial and last name):

```
##############################
##  MTaylor_lab1.R
##  Note: Code for lab assignment #1
##  Author: Marshall A. Taylor
##############################

###BEGIN###
```

You are ready to begin your code. Be sure to include all the code necessary for me to check your work. Document your code thoroughly (using "#").

Remember that saving your R work is a two-step process. You save your R script using Cmd+Enter (Mac) or Cntrl+Enter (Windows). You save your R data objects like this:

```r
save.image("MTaylor_lab1.RData")
```

Lastly, prep a Word, Pages, or LaTeX document that has the same title structure: e.g., **MTaylor_lab1.docx**. This is where you put your write-ups and visualizations.

You should turn in three documents to Sakai: your **R script** that shows the code you used, **RData file** that provides the data your scraped, and **Word document** (or whatever text processor you choose to use) with your write-ups.

Assignment Options

Your task is to scrape your own dataset from a web source, visualize some trends, and provide a write-up. You can scrape Twitter, *New York Times*, or build a custom scraper for Amazon reviews. Each has its own set of instructions.

Option #1: Twitter

Your first option is to scrape some Twitter data, either from a public Twitter profile or using a search term (or series of search terms).

1. Pick a public profile, search term, or series of search terms that interest you (search terms can include hashtags). You may want to use a series of search terms if you want to gather tweets that mention more than one thing or that mention at least one of a series of things (e.g., get tweets that mention "Notre Dame," "ND," or "NotreDame"). If you want to use a series of terms, look at page 53 here, which shows you how you need to structure your syntax to get what you want.
2. In your Word document, provide 3-5 sentences on why you chose to scrape what you want to scrape. What's interesting about it?
3. Scrape some tweets using the search terms or profile handle. Be sure to set your scraper to only gather English language tweets and to exclude retweets. The Twitter REST API won't let you collect more than 18,000 tweets every 15 minutes, so in order to make sure the API doesn't terminate your request, keep your request to under 10,000 tweets. Note that there are several reasons that you won't get exactly the number of tweets you request, so don't worry if you don't get as many as you asked for.
4. In the Word document, copy and paste the first 10 tweets. Recall that you'll want to use the `head()` function in R to get these.
5. What does the frequency of tweets look like over time? Create a line graph that shows the time trend, save it as a PDF, and put it in the Word document. Provide 3-5 sentences explaining what the line graph is telling us.
6. **Optional:** If you scraped using a search term or series of search terms, re-collect the data—this time geocoding the tweets. Plot them on a U.S. map (be sure to limit your scraper to get only U.S. tweets). Provide 3-5 sentences explaining what the map is telling us.
7. **Optional:** If you scraped tweets from a profile, create a bar graph showing what devices or services are being used to post the tweets. Provide 3-5 sentences explaining what the bar graph is telling us.

Option #2: *New York Times*

Your second option is to scrape some *NYT* data.

1. Pick a search term or series of search terms that interest you. Note that the `rtimes` function treats spaces in the search query as AND—meaning that it only cares that the two (or more) terms are both used in the article, not necessarily in the order you provide. If the order matters (say, if it's a particular phrase), put your search in double quotes within single quotes. Look at how I searched the "Unite the Right" phrase in the slides for pres22 to see what I mean. If younonly care that at least one of the terms is mentioned in an article, use "OR" to separate the search terms (e.g., "finances OR financial OR finance"). Use the `begin_date()` and `end_date()` options if you want to limit your search to a particular date range. If you want to try to scrape *all* articles that use that search term (not recommended for this assignment), set `all_results = TRUE` and `sleep = 10`.
2. In your Word document, provide 3-5 sentences on why you chose to scrape what you want to scrape. What's interesting about it?
3. Scrape some articles using the search terms.
4. In the Word document, copy and paste the first 10 main headlines. Recall that you'll want to use the `head()` function in R to get these.
5. What does the frequency of articles using the search term(s) look like over time? Create a line graph that shows the time trend, save it as a PDF, and put it in the Word document. Provide 3-5 sentences explaining what the line graph is telling us.

6. **Optional:** What news desks are the articles coming from? What types of material? Make a couple of bar graphs showing this information. Provide 3-5 sentences explaining what the bar graphs are telling us.
7. **Optional:** What do the article word counts look like over time? Create a line graph that shows the time trend, save it as a PDF, and put it in the Word document. Provide 3-5 sentences explaining what the line graph is telling us.

Option #3: Custom Scraper

Your third option is to build a custom scraper to get some Amazon product reviews.

1. Think of Amazon review data you'd like to scrape.
2. In your Word document, provide 3-5 sentences on why you chose to scrape what you want to scrape. What's interesting about it?
3. Scrape the reviews. If there are a lot of web pages (e.g., more than 200 review pages), limit your scraper to 50 or so pages. Scrape from as many HTML nodes as you want (locating them in the web page with SelectorGadget), but get at least the body of the texts, author names, review title, posting date, and rating.
4. In the Word document, copy and paste the first 10 reviews. Recall that you'll want to use the `head()` function in R to get these.
5. What does the frequency distribution of ratings look like? Create a bar graph to display this information. Provide 3-5 sentences explaining what the bar graph is telling us.
6. **Optional:** How did the ratings change over time? Create a bar graph using time periods as the x-axis and proportion of ratings that fall within each time period as the y-axis (see Figure 11 in pres22.pdf). Provide 3-5 sentences explaining what the graph is telling us.

Hints

The R scripts for the web scraping slides (pres21.R and pres22.R) might be very helpful. Like, *really really helpful*.

The `ts_plot()` function is native to `rtweet`, not `rtimes`. This means that if you choose to scrape *NYT*, you will need to install and load the `rtweet` package in order to make the line graph. You will also need to restructure the publication date variable so it is in a form that the `ts_plot()` function will recognize. Say my scraped *NYT* data are in an object titled "data":

```r
data$time <- as.Date(nyarticle.data$pub_date) + 1
data$datetime <- as.POSIXct(nyarticle.data$time)
```

An example of this is in pres22.R.