The rvest Package in R

Amber Brodeur

5/11/2017

Contents

Introduction

Three tutorials are explored in this project that use various commands in the `rvest` package. Each tutorial explores different commands to scrape a webpage in the `rvest` package. The `rvest` package is used to scrape data from the Internet and is used in conjuction with the `magrittr` package. The tutorials use webpages from TripAdvisor, United, and Zillow. The project is focused on learning how to use the CSS selector tool to select nodes in HTML. The CSS selector tool used in this tutorial is called Selectorgadget. Firstly, "CSS stands for Cascading Style Sheets and is a language used for describing the look and formatting of a document written in a markup language."[1] The Selectorgadget is used to extract specific components from HTML. More specifically, "Selectorgadget is a javascript bookmarklet that allows you to interactively figure out what css selector you need to extract desired components from a page."[2]

You might be asking, why webscrape? Below are a few reasons.

- "Sometimes data are not downloadable.
- The only source might be a web page.
- You want to update the data (in the future).
- There are lots of 'high quality relational data' out there to be scraped." [3]

Objective

The objective of this project is to master the `rvest` package and the CSS selector tool. This will be accomplished by going through the three tutorials available in the `rvest` package, which uses different commands in the `rvest` package.

[1] https://en.wikipedia.org/wiki/Cascading_Style_Sheets
[2] Wickham, Hadley; Selectorgadget
[3] https://rpubs.com/ryanthomas/webscraping-with-rvest

Dataset Description

There are three datasets that represent the three webpages scraped in the three tutorials. The first tutorial scrapes reviews on the JW Marriot Indianapolis hotel from the TripAdvisor webpage. The second tutorial scrapes flights from the United Airlines webpage by using your MileagePlus number and password. The third tutorial scrapes data from the Zillow webpage.

Variables

There is one set of variables for each tutorial with a total of three sets of variables. Below are three tables representing each set.

TripAdvisor Variables

The table below is the data scraped of the reviews on the JW Marriot Indianapolis hotel from the TripAdvisor webpage.

Variable Name	Description	Unit
id	primary key / unique id	id
quote	quote from the hotel review	character
date	date of the review	month, day, year
review	hotel review	character

United Variables

The table below is the data scraped on the flights from the United webpage.

Variable Name	Description	Unit

Zillow Variables

The table below is the data scraped from the Zillow webpage.

Variable Name	Description	Unit
z_id	unique id	id
address	location	character
price	listing price of house	USD
beds	number of bedrooms	numeric
baths	number of bathrooms	numeric
house_area	area size of house	square feet

Preparation

In this tutorial we used the CSS selector tool (Selectorgadget). To get started with webscraping, we will first take a look at how to use the Selectorgadget. Again, "Selectorgadget is a javascript bookmarklet that allows you to interactively figure out what css selector you need to extract desired components from a page."[4] To grasp a basic understanding of how to use it in R, use the command `vignette("selectorgadget")` to return installation and user instructions on the Selectorgadget.

```r
vignette("selectorgadget")
```

A tutorial to practice with the Selector Gadget can be found at the following website: http://flukeout.github.io/#.

Install the packages used in this project.

```r
install.packages('rvest')
install.packages('magrittr')
```

Package descriptions:

- The `rvest` package is used to scrape data from the webpage.
- The `xml2` and `httr` packages are downloaded with the `rvest` package. These two packages are used to download and manipulate XML and HTML data.
- The `magrittr` package is used for the pipe operator.

Load the libraries into the workspace.

```r
library(rvest)
library(magrittr)
library(tidyr) #used in zillow tutorial
```

We will explore the `rvest` package by reviewing demonstrations available in the `rvest` package. The command below lists the demonstrations in the `rvest` package.

```r
demo(package = "rvest")
```

There are three demonstrations available for the `rvest` package:

- `tripadvisor` - Scrape review data from TripAdvisor
- `united` - Scrape mileage details from united.com
- `zillow` - Scrape housing information from Zillow

"The basic functions in rvest are powerful, and you should try to utilize the following functions when starting out a new project.

- `html_nodes()`: identifies HTML wrappers.
- `html_nodes(".class")`: calls node based on css class
- `html_nodes("#id")`: calls node based on
  id
- `html_nodes(xpath="xpath")`: calls node based on xpath
- `html_attrs()`: identifies attributes (useful for debugging)
- `html_table()`: turns HTML tables into data frames
- `html_text()`: strips the HTML tags and extracts only the text
- Note on plurals: html_node() returns metadata for the , html_nodes() iterates over the matching nodes." [5]

Tutorial 1 - TripAdvisor

In this section, the TripAdvisor webpage is scraped for review data on the JW Marriott Hotel in Indianapolis, Indiana.

[4] Wickham, Hadley; Selectorgadget
[5] https://rpubs.com/ryanthomas/webscraping-with-rvest

The command below explores the TripAdvisor demonstration from the `rvest` package. When the `ask` setting is set to true, the demonstration pauses between pages. The results are not shown for the code-chunk below.

```
demo(tripadvisor, package = "rvest", ask = FALSE)
```

Below, the TripAdvisor url is saved as an object in the R environment.

```
url <- "http://www.tripadvisor.com/Hotel_Review-g37209-d1762915-Reviews-JW_Marriott_Indianapolis-Indiana
```

The `read_html` command is used to read the the Tripadvisor url. The pipe operator is used to forward the result of this expression to the `html_nodes` command. The `html_nodes` command is used to select nodes from a HTML document. In this case, to select the nodes of the reviews from the Tripadvisor webpage. The Selector gadget is used to find the nodes. The result is assigned to `reviews`.

```
reviews <- url %>%
   read_html() %>%
   html_nodes("#REVIEWS .innerBubble")
```

The command `length` is a base R command. This command returns the length of an object. The line of code below returns the number of reviews extracted from the TripAdvisor website.

```
length(reviews)
```

```
## [1] 10
```

There were ten reviews extracted.

The `html_text` command extracts text from HTML. The code below extracts the text of the ten Tripadvisor reviews.

```
html_text(reviews)
```

```
##  [1] ""Cancelled our room the day of our Anniversary"Reviewed yesterday  NEWMaybe this hotel is great
##  [2] ""Terribly managed hotel"Reviewed yesterday  NEWI travel extensively (150 to 170 room nights pe
##  [3] ""Would recommend"Reviewed 3 days ago  NEWNice place. Would recommend. Happy and friendly staff
##  [4] ""Three Star at best"Reviewed 3 days ago  NEWClean hotel with spacious rooms and high prices! C
##  [5] ""Wedding Night Stay"Reviewed 4 days ago  NEWvia mobileAs we agree, this is a beautiful hotel,
##  [6] ""Beautiful "Reviewed 6 days ago  NEWvia mobileNo words can describe how awesome this hotel is.
##  [7] ""Great stay"Reviewed 1 week ago  via mobileNever a doubt by here, close to everything. Clean,
##  [8] ""Stolen Medication"Reviewed 1 week ago  I remembered to put cash and other valuables in the sa
##  [9] ""Great Service, great location"Reviewed 1 week ago  Check In: excellent, fast, easy and got an
## [10] ""Great stay"Reviewed 1 week ago  via mobileThis hotel is very contemporary and very big. The v
```

The `reviews` are piped to the`html_node` command. The

With the Selector gadget, click on the title/quote of the review. When hovering over this selection, in small print you will see an `a` and `span`. These are two nodes for the quote. The `a` node is used to extract the id from the reviews. The `span` node will be used later. The `html_attr` command extracts attributes from HTML.

```
id <- reviews %>%
   html_node(".quote a") %>%
   html_attr("id")
```

The line of code below returns the number of ID's extracted from the TripAdvisor website.

```
length(id)
```

```
## [1] 10
```

There were ten ID's extracted.

The id attribute is returned below.

```
id
```

```
##  [1] "rn483230381" "rn483221901" "rn482756824" "rn482662837" "rn482440147"
##  [6] "rn481963785" "rn481436053" "rn481102030" "rn481100806" "rn480758183"
```

The span node mentioned above is used here. The span node extracts the quote from the Tripadvisor review. The html_text command extracts text from HTML.

```
quote <- reviews %>%
   html_node(".quote span") %>%
   html_text()
```

The quote attribute is returned below.

```
quote
```

```
##  [1] "Cancelled our room the day of our Anniversary"
##  [2] "Terribly managed hotel"
##  [3] "Would recommend"
##  [4] "Three Star at best"
##  [5] "Wedding Night Stay"
##  [6] "Beautiful "
##  [7] "Great stay"
##  [8] "Stolen Medication"
##  [9] "Great Service, great location"
## [10] "Great stay"
```

The date is extracted from the reviews object. The nodes are found using the css Selector gadget. The html_attr command extracts the attribute from HTML. The strptime command converts character vector to class “POSIXlt” and each input string is processed for the specified format. “POSIXlt” represents calendar dates and times and is a list of vectors that represents: seconds, minutes, hours, day of the month, year, day of the week, day of the year, daylight saving time, and time zone.

```
date <- reviews %>%
   html_node(".rating .ratingDate") %>%
   html_attr("title") %>%
   strptime("%b %d, %Y") %>%
   as.POSIXct()
```

The date attribute is returned below.

```
date
```

```
##  [1] "2017-05-10 EDT" "2017-05-10 EDT" "2017-05-08 EDT" "2017-05-08 EDT"
##  [5] "2017-05-07 EDT" "2017-05-05 EDT" "2017-05-03 EDT" "2017-05-02 EDT"
##  [9] "2017-05-02 EDT" "2017-05-01 EDT"
```

There are 10 dates in date.

The reviews are extracted from the reviews object. The nodes are found using the css Selector gadget. The html_text command extracts text from HTML.

```
review <- reviews %>%
   html_node(".entry .partial_entry") %>%
   html_text()
```

The review attribute is returned below.

```
review
```

```
##  [1] "Maybe this hotel is great, but we never got to find out. We booked through a third party site
##  [2] "I travel extensively (150 to 170 room nights per year) on business and believe I am easy to ple
##  [3] "Nice place. Would recommend. Happy and friendly staff...even when making a slight error, they r
##  [4] "Clean hotel with spacious rooms and high prices! Centrally located to all major attractions. Be
##  [5] "As we agree, this is a beautiful hotel, and although the room was pretty and up to date, we had
##  [6] "No words can describe how awesome this hotel is. The view from the windows are unbelievable. Th
##  [7] "Never a doubt by here, close to everything. Clean, comfortable, cozy, and accommodating. The en
##  [8] "I remembered to put cash and other valuables in the safe, I used a luggage lock etc. While in a
##  [9] "Check In: excellent, fast, easy and got an upgrade to the Concierge business floor\nRoom: Very
## [10] "This hotel is very contemporary and very big. The view from some of the rooms is all you could
```

The `id`, `quote`, `date`, and `review` are combined to form a dataframe named `ta`.

```r
ta <- data.frame(id, quote, date, review, stringsAsFactors = FALSE)
```

Review the structure of the `ta` dataframe.

```r
str(ta)
```

```
## 'data.frame':    10 obs. of  4 variables:
##  $ id    : chr  "rn483230381" "rn483221901" "rn482756824" "rn482662837" ...
##  $ quote : chr  "Cancelled our room the day of our Anniversary" "Terribly managed hotel" "Would reco
##  $ date  : POSIXct, format: "2017-05-10" "2017-05-10" ...
##  $ review: chr  "Maybe this hotel is great, but we never got to find out. We booked through a third p
```

The dataframe has 10 observations and four variables. Conveniently, the data looks good and does not need any clean-up.

In order to save the data set as a csv file, we set the working directory and then used the write.csv command to export the data to the file path. Below, the csv file was saved on the desktop.

```r
setwd("/Users/amberbrodeur/Desktop/Web Scrape")
write.csv(ta, 'tripadvisor.csv', row.names=FALSE)
```

Check to make sure that the data was sucessfully and properly saved in the the working directory location.

Tutorial 2 - United Airlines

In this section, a tutorial from the `rvest` package on United Airlines is reviewed.

The command below explores the `united` demonstration from the `rvest` package. When the `ask` setting is set to true, the demonstration pauses between pages.

```r
demo(united, package = "rvest", ask = FALSE)
```

NEED A UNITED MILEAGE PLUS USERNAME AND PASSWORD.

The `html_session` command, "Simulates a session in an html browser. A session object responds to a combination of httr and html methods: use `cookies()`, `headers()`, and `status_code()` to access properties of the request; and html_nodes to access the html."[6] Below, we want to simlulate a session using the United url.

```r
united <- html_session("http://www.united.com/")
```

To access your MileagePlus account, the following function is used. The node for the login form is found using the CSS selector tool. The values need to be set for the MilagePlus number and password.

[6]R Documentation, html_session {rvest}

```
login <- united %>%
   html_node("form[name=LoginForm]") %>%
   html_form() %>%
   set_values(
     MpNumber = "GY797363",
     Password = password
     )
```

After the MileagePlus number and password is entered. The data on your flights can be extracted using similar techniques from this tutorial.

Tutorial 3 - Zillow

In this section, a tutorial from the `rvest` package on Zillow is reviewed.

The command below explores the `zillow` demonstration from the `rvest` package. When the `ask` setting is set to true, the demonstration pauses between pages.

```
demo(zillow, package = "rvest", ask = FALSE)
```

The `read_html` command is used to read the Zillow url into R. The url is saved as an object named `page`.

```
page <- read_html("http://www.zillow.com/homes/for_sale/Greenwood-IN/fsba,fsbo,fore,cmsn_lt/house_type/
```

The Zillow url `page` is read. The pipe operator is used to forward the Zillow webpage to the `html_nodes` command. The `html_nodes` command is used to select nodes from a HTML document. In this case, to select the nodes of the houses from the Zillow webpage. The Selector gadget is used to find the nodes. The result is assigned to `houses`. The first six observations are displayed below.

```
houses <- page %>%
  html_nodes(".photo-cards li article")
```

```
head(houses)
```

```
## {xml_nodeset (6)}
## [1] <article data-photocount="43" data-audiencetesteventlabel="" data-gr ...
## [2] <article data-photocount="49" data-audiencetesteventlabel="" data-gr ...
## [3] <article data-photocount="38" data-audiencetesteventlabel="" data-gr ...
## [4] <article data-photocount="8" data-audiencetesteventlabel="" data-gro ...
## [5] <article data-photocount="14" data-audiencetesteventlabel="" data-gr ...
## [6] <article data-photocount="25" data-audiencetesteventlabel="" data-gr ...
```

The `html_attr` command extracts attributes from HTML. In the command below, the `id` attribute is extracted from the `houses` object. The object is saved as `z_id`. The first six observations are displayed below.

```
z_id <- houses %>% html_attr("id")
```

```
head(z_id)
```

```
## [1] "zpid_124612912" "zpid_85449900"  "zpid_85450049"  "zpid_85448849"
## [5] "zpid_85463450"  "zpid_85452917"
```

The `html_nodes` command is used to select the photo card address from the `houses` object. The `html_text` command extracts text from HTML. The code below extracts the text from the node that was extracted. The object is saved as `address`. The first six observations are displayed below.

```
address <- houses %>%
  html_node(".zsg-photo-card-address") %>%
  html_text()
```

```
head(address)
```

```
## [1] "1312 Brentford Ln, Greenwood, IN" "1819 Dockside Dr, Greenwood, IN"
## [3] "2124 Cheviot Ct, Greenwood, IN"   "36 Westridge Blvd, Greenwood, IN"
## [5] "1352 Oxford Run, Greenwood, IN"   "864 Lionshead Ln, Greenwood, IN"
```

The `html_nodes` command is used to select the photo card price from the `houses` object. The `html_text` command extracts text from HTML node. The `readr::parse_number()` command is from the `readr` library and, "Drops any non-numeric characters before or after the first number. The grouping mark specified by the locale is ignored inside the number."[7] The object is saved as `price`. The first six observations are displayed below.

```
price <- houses %>%
  html_node(".zsg-photo-card-price") %>%
  html_text() %>%
  readr::parse_number()
```

```
head(price)
```

```
## [1] 359900 650000 489900 142000 159000 144900
```

The `html_nodes` command is used to select the photo card information from the `houses` object. The `html_text` command extracts text from HTML node. The object is saved as `price`. The `strplit` command, "Split the elements of a character vector x into substrings according to the matches to substring split within them."[8] In this case, it breaks up the number of bedrooms, number of bathrooms, and square footage of the houses. The object is saved as `params`. The first six observations are displayed below.

```
params <- houses %>%
  html_node(".zsg-photo-card-info") %>%
  html_text() %>%
  strsplit("\u00b7")
```

```
head(params)
```

```
## [[1]]
## [1] "4 bds "      " 3 ba "      " 3,385 sqft"
##
## [[2]]
## [1] "5 bds "      " 3 ba "      " 5,204 sqft"
##
## [[3]]
## [1] "6 bds "      " 5 ba "      " 4,960 sqft"
##
## [[4]]
## [1] "3 bds "      " 3 ba "      " 2,221 sqft"
##
## [[5]]
## [1] "3 bds "      " 2 ba "      " 1,460 sqft"
##
## [[6]]
## [1] "3 bds "      " 2 ba "      " 1,514 sqft"
```

[7]R Documentation, parse_number {readr}
[8]R Documentation, strsplit {base}

The `purrr::map_chr(1)` command, "Returns vectors of the corresponding type." [9] The `readr::parse_number()` command is from the `readr` library and, "Drops any non-numeric characters before or after the first number. The grouping mark specified by the locale is ignored inside the number."[10]. The first object in the `params` object is mapped and parsed. The object is saved as `beds`. The first six observations are displayed below.

```
beds <- params %>%
  purrr::map_chr(1) %>%
  readr::parse_number()
```

```
head(beds)
```

```
## [1] 4 5 6 3 3 3
```

The `purrr::map_chr(2)` command, "Returns vectors of the corresponding type." [11] The `readr::parse_number()` command is from the `readr` library and, "Drops any non-numeric characters before or after the first number. The grouping mark specified by the locale is ignored inside the number."[12]. The second object in the `params` object is mapped and parsed. The object is saved as `baths`. The first six observations are displayed below.

```
baths <- params %>%
  purrr::map_chr(2) %>%
  readr::parse_number()
```

```
head(baths)
```

```
## [1] 3 3 5 3 2 2
```

The `purrr::map_chr(3)` command, "Returns vectors of the corresponding type." [13] The `readr::parse_number()` command is from the `readr` library and, "Drops any non-numeric characters before or after the first number. The grouping mark specified by the locale is ignored inside the number."[14]. The third object in the `params` object is mapped and parsed. The object is saved as `house_area`. The first six observations are displayed below.

```
house_area <- params %>%
  purrr::map_chr(3) %>%
  readr::parse_number()
```

```
## Warning: 1 parsing failure.
## row col expected actual
##  18  -- a number      -
```

```
head(house_area)
```

```
## [1] 3385 5204 4960 2221 1460 1514
```

The `z_id`, `address`, `price`, `beds`, `baths`, and `house_area` are combined to form a dataframe named `z`.

```
z <- data.frame(z_id, address, price, beds, baths, house_area, stringsAsFactors = FALSE)
```

Review the structure of the `z` dataframe.

```
str(z)
```

```
## 'data.frame':    27 obs. of  6 variables:
##  $ z_id      : chr  "zpid_124612912" "zpid_85449900" "zpid_85450049" "zpid_85448849" ...
##  $ address   : chr  "1312 Brentford Ln, Greenwood, IN" "1819 Dockside Dr, Greenwood, IN" "2124 Chevi
```

[9] R Documentation, map {purrr}
[10] R Documentation, parse_number {readr}
[11] R Documentation, map {purrr}
[12] R Documentation, parse_number {readr}
[13] R Documentation, map {purrr}
[14] R Documentation, parse_number {readr}

```
##  $ price     : num  359900 650000 489900 142000 159000 ...
##  $ beds      : num  4 5 6 3 3 3 3 3 4 3 ...
##  $ baths     : num  3 3 5 3 2 2 3 3 3 2 ...
##  $ house_area: atomic  3385 5204 4960 2221 1460 ...
##   ..- attr(*, "problems")=Classes 'tbl_df', 'tbl' and 'data.frame':  1 obs. of  4 variables:
##   .. ..$ row     : int 18
##   .. ..$ col     : int NA
##   .. ..$ expected: chr "a number"
##   .. ..$ actual  : chr "-"
```

The dataframe has 27 observations and 6 variables.

In order to save the data set as a csv file, we set the working directory and then used the write.csv command to export the data to the file path. Below, the csv file was saved on the desktop.

```r
setwd("/Users/amberbrodeur/Desktop/Web Scrape")
write.csv(z, 'zillow.csv', row.names=FALSE)
```

Check to make sure that the data was sucessfully and properly saved in the the working directory location.

Conclusion

This tutorial drove you through the steps for the three tutorials in the `rvest` package. You have learned how to scrape HTML tables, clean the data, and export the data as a csv file. Well done!