H#7-Milad

Milad Tatari

10/19/2019

(A) Pick at least 2 web scraping toolkits (either automated tools like Import.io or R packages such as rvest) and try to use them to extract data from the Yelp website. In particular, create a search in Yelp to find good burger restaurants in the Boston area. You must try out at least two toolkits, but you will use only one to actually extract and save the full data.

(B) Import the data you extracted into a data frame in R. Your data frame should have exactly 30 rows, and each row represents a burger restaurant in Boston.

Answer: We want to extract infomation of resturants serving burger in Boston area via Yelp. We will do this using rvest package in R and the other method which is a graphical interface called import.io. First we will start with "rvest" as follows to extract information of first 3 pages and finally save it as a data frame. (90 restaurants and 7 variables)

```
#Let's first install all required packages:
#install.packages("stringr")
library('stringr')
library('ggplot2')
library('rvest')
```

```
## Loading required package: xml2
```

```
library('png')

# Restaurants are searched in Allston, Brighton, Back Bay, Beacon Hill, Downtown Area, Fenway, South En
# Restaurants Yelp Links for the first 3 pages (each page has 30 restaurants )
Res.YL <- c("https://www.yelp.com/search?find_desc=burgers&find_loc=Boston%2C%20MA&l=p%3AMA%3ABoston%3A
            "https://www.yelp.com/search?find_desc=burgers&find_loc=Boston%2C%20MA&l=p%3AMA%3ABoston%3A%
            "https://www.yelp.com/search?find_desc=burgers&find_loc=Boston%2C%20MA&l=p%3AMA%3ABoston%3A%
length(Res.YL)
```

```
## [1] 3
```

```
# Making a data frame to store all the desired results
 Res.Burger.Bos<- data.frame(Name=character(),
                    PhoneNo.=character(),
                    Address=character(),
                    Price.Range=character(),
                    Categories=character(),
                    ReviewNo.=character(),
                stringsAsFactors=F)
#Writing a for loop to extract as many pages as we want. (here we have 3 pages)
for (i in 1:length(Res.YL)){
  theurl <- read_html(Res.YL[i])

# Extracting the name of restaurants by html nodes, element id
Res.Names<-theurl %>%
  html_nodes("h3 + p > a") %>%
```

```
html_text()
Res.Names <- Res.Names[-1]
if (length(Res.Names)==29){
f <- html_nodes(theurl,"h3 + p > a")
    Res.Names <- html_text(f)
  }

# Extracting the phones of restaurants by html nodes, class id
Res.Phones <- theurl %>%
  html_nodes(".text-align--right__373c0__3fmmn") %>%# It has the phone information
  html_text() %>%
  str_extract('[(][0-9]{3}[)] [0-9]{3}-[0-9]{4}')
Res.Phones <- Res.Phones[-1] #We remove the advertisement restaurant
Res.Full.Address <- theurl %>%
  html_nodes(".text-align--right__373c0__3fmmn") %>%
  html_text() %>%
  str_replace("[(][0-9]{3}[)] [0-9]{3}-[0-9]{4}","")
Res.Full.Address <- Res.Full.Address[-1]




#Extracting the price info (like $ or $$ or $$$) and service category of the restaurants
Res.price.categ<-theurl %>%
  html_nodes(".priceCategory__373c0__3zW0R") %>%
  html_text()

Res.price.categ <- Res.price.categ[-1]
Res.Price <- str_extract(Res.price.categ, '[$]+')
  Res.Categ <- gsub('[$]+', '', Res.price.categ)

# Now, we extract the number of reviews
  Res.Re.Co<-theurl %>%
  html_nodes(".reviewCount__373c0__2r4xT") %>%
  html_text()  %>%
  str_replace( "review[s]*","")
Res.Re.Co <- Res.Re.Co[-1]
# if the length is 29 we do not want to remove an observation
if (length(Res.Re.Co)==29){
f <- html_nodes(theurl,".reviewCount__373c0__2r4xT")
    Res.Re.Co <- gsub(' review[s]*', '', html_text(f))
  }
# placing the data in the data frame that has already been created
 Res.info.Boston<- data.frame(Name=Res.Names,
                   PhoneNo=Res.Phones,
                   Address=Res.Full.Address,
                   Price.Range=Res.Price,
                   Categories=Res.Categ,
                   ReviewNo=Res.Re.Co,
               stringsAsFactors=F)
  # Now, we add all the results of the pages 2 and 3 to the end of 1st page as it changes the page:
```

```
  Res.Burger.Bos <- rbind(Res.Burger.Bos, Res.info.Boston)
}

head(Res.Burger.Bos, 10)
```

```
##                              Name        PhoneNo
## 1  MOOYAH Burgers, Fries & Shakes (857) 277-0176
## 2                    Tasty Burger (617) 425-4444
## 3                     The Gallows (617) 425-0200
## 4                       Jm Curley (617) 338-5333
## 5                            Coda (617) 536-2632
## 6                     Wahlburgers (617) 927-6810
## 7                     Shake Shack (617) 933-5050
## 8                     Lion's Tail (857) 239-9276
## 9           Boston Baddest Burger           <NA>
## 10                    Saltie Girl (617) 267-0691
##                        Address Price.Range
## 1       140 Tremont StDowntown           $
## 2       1301 Boylston StFenway           $
## 3  1395 Washington StSouth End          $$
## 4         21 Temple PlDowntown          $$
## 5     329 Columbus AveBack Bay          $$
## 6      132 Brookline AveFenway          $$
## 7   234-236 Newbury StBack Bay          $$
## 8    354 Harrison AveSouth End          $$
## 9   Stuart And TrinityBack Bay        <NA>
## 10    281 Dartmouth StBack Bay         $$$
##                                                 Categories ReviewNo
## 1  Burgers, American (Traditional), Ice Cream & Frozen Yogurt      95
## 2                                Burgers, Hot Dogs, Fast Food    1114
## 3                      Burgers, Bars, American (Traditional)     852
## 4                                American (New), Lounges     786
## 5                     American (New), Burgers, Cocktail Bars     588
## 6                           American (Traditional), Burgers     830
## 7                Burgers, Hot Dogs, Ice Cream & Frozen Yogurt     375
## 8                             Cocktail Bars, American (New)     124
## 9                                       Burgers, Sandwiches       1
## 10                        Seafood, Wine Bars, Cocktail Bars     897
```

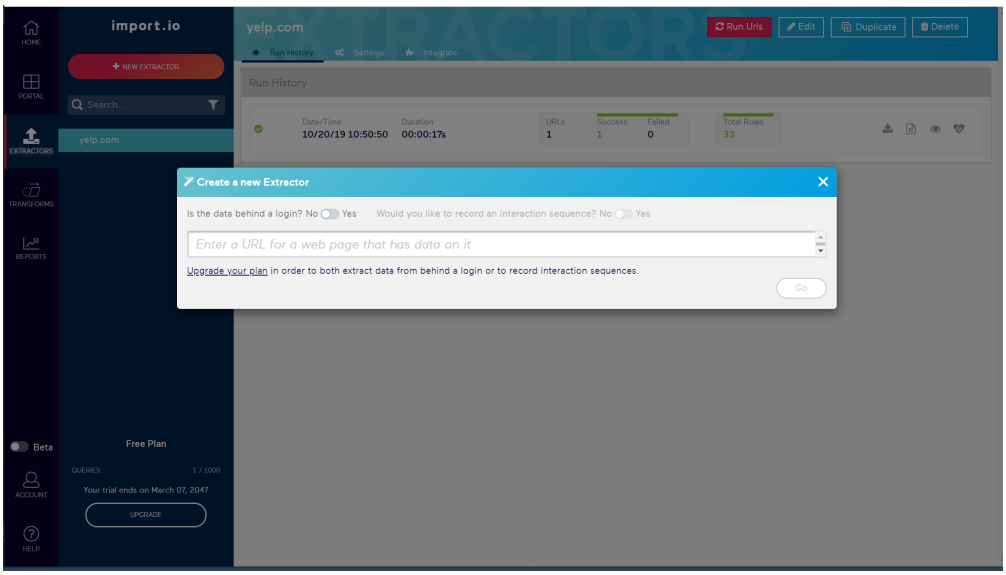

Figure 1: Second Method: import.io

The next too that I am going to use to extract the data is import.io. It is a user interface tool to extract the desired information from a website with no knowledge of programming and save the results as excel, CSV and .... For this problem, I am using free version of import.io tool which is enough to extract the information of up to 1000 URLs. That being said, lets start using this tool. I have taken some screen shots of different steps to make it clear and write the process down as a report in RMarkdown.

Step 1: import.io interface

Step 2: enetring the URL address

Step 3: It automatically gives us data columns that need to be modified and costomized depending on what we need.

Step 4: Costomizing the columns as name, address, price, categories of restaurants and so on.

Step 5: How to select appropriate boxes, green boxes (street addresses)

Step 6: Making the full data columns as we wanted, we just need to extract it and save as .CSV file

```
# we read the data via read.csv and then it is loaded as a data frame
mydata <- read.csv(file="Yelp.Res.Bos-(Crawl-Run)---2019-10-20T161808Z.csv")
mydata <- mydata[,2:7]
head(mydata [1:5,])
```

```
##                              Name      Phone.No.             Street
## 1 MOOYAH Burgers, Fries & Shakes (857) 277-0176     140 Tremont St
## 2                     The Gallows (617) 425-0200 1395 Washington St
## 3                     Wahlburgers (617) 927-6810  132 Brookline Ave
```

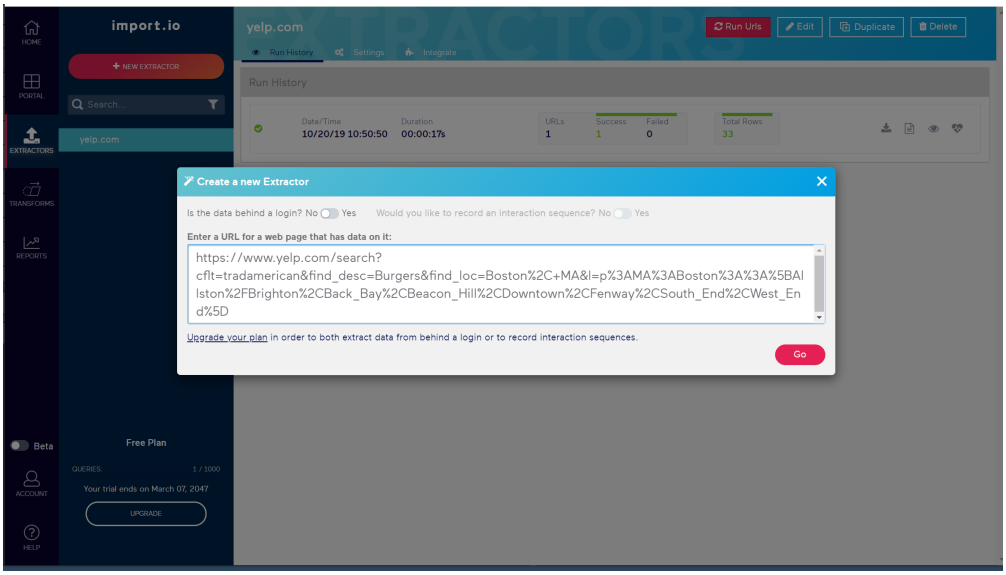

Figure 2: import.io: Entering the URL address

Figure 3: import.io: Getting the default data columns

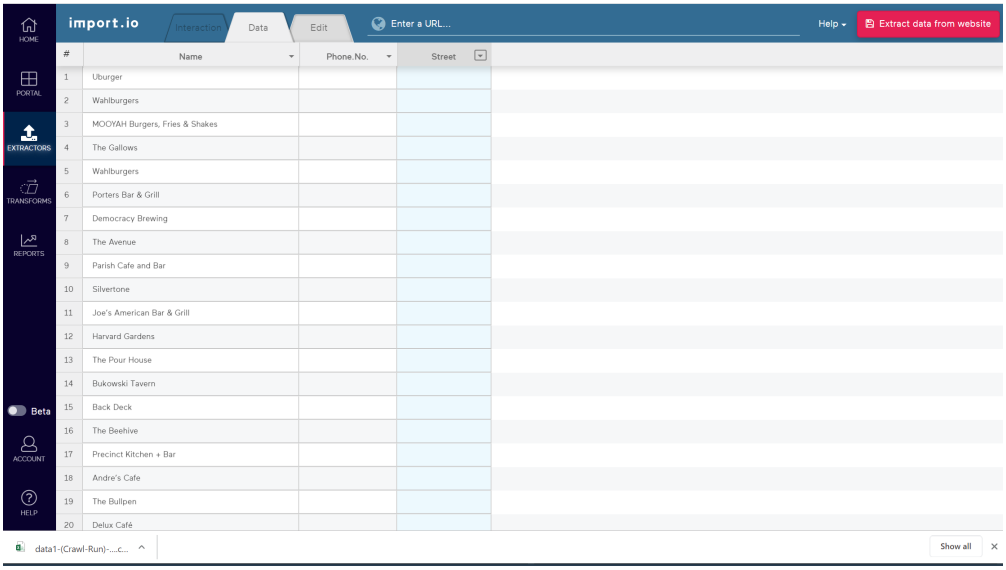

Figure 4: import.io: costomizing the data columns

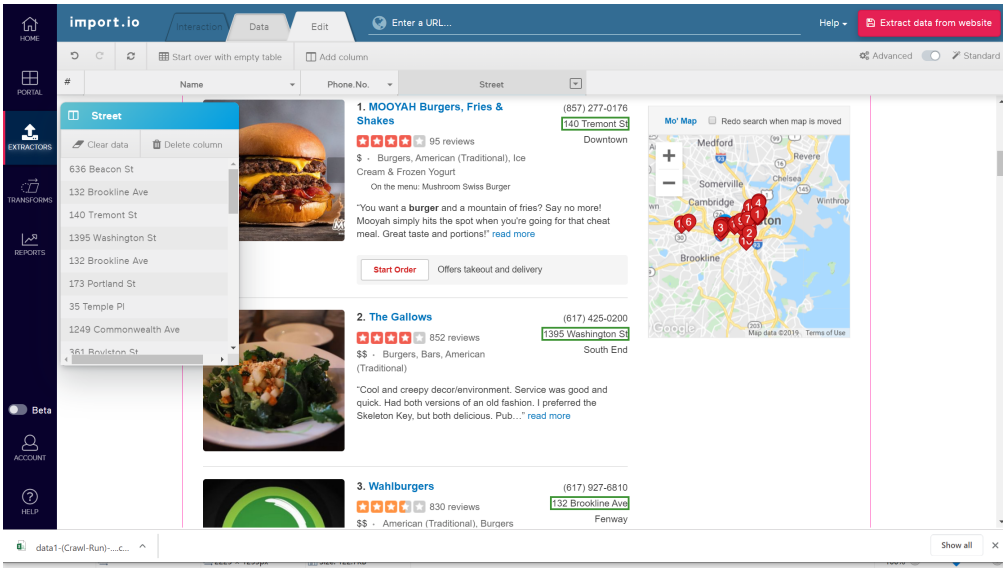

Figure 5: import.io: selecting addresses of restaurants

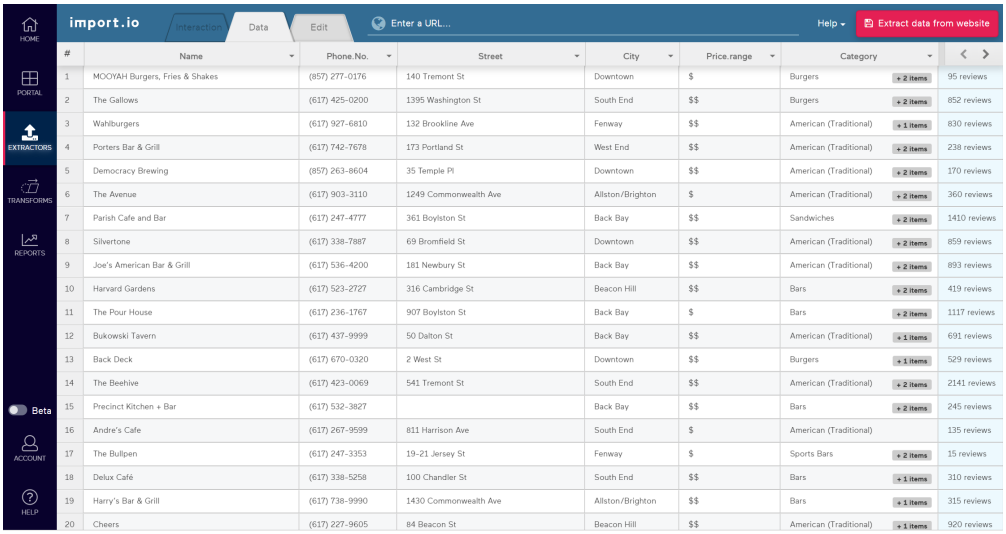

Figure 6: import.io: Full data table is derived

```
## 4            Porters Bar & Grill (617) 742-7678    173 Portland St
## 5              Democracy Brewing (857) 263-8604       35 Temple Pl
##        City Price.range
## 1  Downtown           $
## 2 South End          $$
## 3    Fenway          $$
## 4  West End          $$
## 5  Downtown          $$
##                                                          Category
## 1 Burgers; American (Traditional); Ice Cream & Frozen Yogurt
## 2                      Burgers; Bars; American (Traditional)
## 3                            American (Traditional); Burgers
## 4                      American (Traditional); Pubs; Burgers
## 5   American (Traditional); Breweries; Venues & Event Spaces
```

(C)Write a report that compares the tools with a focus on cost, ease of use, features, and your recommendation. Discuss your experience with the tools and why you decided to use the one you picked in the end. Use screenshots of toolkits and your scraping process to support your statements. Also include a screenshot or an excerpt of your data in the report. ## R Markdown

Answer: In terms of cost, import.io as I mentioned before is a free graphical user interface up to 1000 URLs and easy to use if an individual does not have any programming experience. "rvest" is also a free package. Furtheremore, It always needs data cleaning as it is customizable and by default, it initiates a dada frame that is not useful. Furtheremore, it is somehow exsausting for programmers to do all these steps manually to extract the web data. Personally, I am much more comfortable to use a programming package like rvest and develop my own cod by html nodes and elements that work perfectly for that website.

For a particular website, if the data is only needed for just one time, import.io might be a good option, but if we want to keep track of the data it is much better to write our own code and have the data directly in R without need to import it every time to R. Packages in R give also the abilty of online data analysis and desicsion making as it can directly read the data and store it.

In addition, when you are working with import.io, you need to be carefull since it sometimes selects some unuseful information and we have to double chack (and potentially clean) every selected column in import.io. What I am trying to say is that there is more chance of having mistakes working with import.io.

(D) Within your report describe what you have derived about the URL for yelp pages. What are the differences between the three URLs? What are the parameters that determined your search query (Boston burger restaurants in 8 selected neighborhoods)? What is(are) the parameter(s) used for pagination? Without opening Yelp.com in the browser, what is your guess of the URL for the 7th page of Chinese restaurants in New York?

Every URL has some sperate elements giving information about the website. As an instance, I am going to investigate the URL elements of YELP searches in this assignment.

For the first page we have : https://www.yelp.com/search?find_desc=Burgers&find_loc=&l=p%3AMA%3ABoston%3A%3A%5BAllston%2FBrighton%2CBack_Bay%2CBeacon_Hill%2CDowntown%2CFenway%2CSouth_End%2CWest_End%5D

For the second page starting from restuarant 31:

https://www.yelp.com/search?find_desc=Burgers&find_loc=&l=p%3AMA%3ABoston%3A%3A%5BAllston%2FBrighton%2CBack_Bay%2CBeacon_Hill%2CDowntown%2CFenway%2CSouth_End%2CWest_End%5D&start=30

FOr URLs, after https://www.yelp.com/ we have search item which is "Burgers" here. and then there are all locations as "loc=&l=p%3AMA%3ABoston%3A%3A%5BAllston%2FBrighton%2CBack_Bay%2CBeacon_Hill%2CDowntow untile the end with all 8 locations as Allston, Brighton, Back Bay, Beacon Hill, Downtown Area, Fenway, South End, and West End.

For the second page we se that "&start=30" has been added to end of URL. It means that in the second page resaurant numbers start with 31. and for third page "&start=60" will be added to the end. It means that the 3rd page starts from restaurant 61.

For chinese restaurants in New York and 7th page, we expect to have "chinese restaurants" in the search field, and location of "New York" and for the 7th page "&start=180" at the end.

https://www.yelp.com/search?find_desc=Chinese%20Restaurants&find_loc=New%20York&start=180