Week\_4\_FLS\_assignment

Todd Garner

2023-01-21

Table of Contents

# FLS - Week 4

## FLS Page 2 - Run the file “Unit\_4\_Web\_Scraping.R”

Run the example code in the file Unit 4 Web Scraping.R This will help tremendously with the web scraping in Part 1. You will have to learn the grep and grepl functions which I think you will find very useful and we will extend on in the future. (1 hour)

I opened the existing/provided file and copied and pasted into the R code chunks below. Let’s see how we do!

#Live Session 4 For Live Session Web Scraping Code

#install.packages("XML")  
library(XML) #xml\_Parse  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(tidyr)  
library(stringi)  
library(rvest) #html\_table, html\_node  
library(ggplot2)  
#install.packages("RCurl")  
library(RCurl) #getURL

##   
## Attaching package: 'RCurl'

## The following object is masked from 'package:tidyr':  
##   
## complete

#Basics of Scraping XML

# Method 1: XML

data <-getURL("https://www.w3schools.com/xml/simple.xml")  
doc <- xmlParse(data)  
names <- xpathSApply(doc,"//name",xmlValue)  
price <- xpathSApply(doc,"//price",xmlValue)  
description <- xpathSApply(doc,"//description",xmlValue)  
bfasts = data.frame(names,price,description)  
bfasts

## names price  
## 1 Belgian Waffles $5.95  
## 2 Strawberry Belgian Waffles $7.95  
## 3 Berry-Berry Belgian Waffles $8.95  
## 4 French Toast $4.50  
## 5 Homestyle Breakfast $6.95  
## description  
## 1 Two of our famous Belgian Waffles with plenty of real maple syrup  
## 2 Light Belgian waffles covered with strawberries and whipped cream  
## 3 Light Belgian waffles covered with an assortment of fresh berries and whipped cream  
## 4 Thick slices made from our homemade sourdough bread  
## 5 Two eggs, bacon or sausage, toast, and our ever-popular hash browns

bfasts$description

## [1] "Two of our famous Belgian Waffles with plenty of real maple syrup"   
## [2] "Light Belgian waffles covered with strawberries and whipped cream"   
## [3] "Light Belgian waffles covered with an assortment of fresh berries and whipped cream"  
## [4] "Thick slices made from our homemade sourdough bread"   
## [5] "Two eggs, bacon or sausage, toast, and our ever-popular hash browns"

length(grep("covered",bfasts$description))

## [1] 2

grepl("covered",bfasts$description)

## [1] FALSE TRUE TRUE FALSE FALSE

sum(grepl("covered",bfasts$description))

## [1] 2

which(grepl("covered",bfasts$description))

## [1] 2 3

# Method 2: rvest

hp<-read\_html("https://www.w3schools.com/xml/simple.xml")  
hp\_nameR <- html\_nodes(hp,"name")  
hp\_priceR <- html\_nodes(hp,"price")  
hp\_descR <- html\_nodes(hp,"description")  
hp\_nameR

## {xml\_nodeset (5)}  
## [1] <name>Belgian Waffles</name>  
## [2] <name>Strawberry Belgian Waffles</name>  
## [3] <name>Berry-Berry Belgian Waffles</name>  
## [4] <name>French Toast</name>  
## [5] <name>Homestyle Breakfast</name>

hp\_name = stri\_sub(hp\_nameR,7,-8)

## Warning in stri\_sub(hp\_nameR, 7, -8): argument is not an atomic vector; coercing

hp\_name

## [1] "Belgian Waffles" "Strawberry Belgian Waffles"   
## [3] "Berry-Berry Belgian Waffles" "French Toast"   
## [5] "Homestyle Breakfast"

hp\_price = stri\_sub(hp\_priceR,8,-9)

## Warning in stri\_sub(hp\_priceR, 8, -9): argument is not an atomic vector;  
## coercing

hp\_price

## [1] "$5.95" "$7.95" "$8.95" "$4.50" "$6.95"

hp\_desc = stri\_sub(hp\_descR,14,-15)

## Warning in stri\_sub(hp\_descR, 14, -15): argument is not an atomic vector;  
## coercing

hp\_desc

## [1] "Two of our famous Belgian Waffles with plenty of real maple syrup"   
## [2] "Light Belgian waffles covered with strawberries and whipped cream"   
## [3] "Light Belgian waffles covered with an assortment of fresh berries and whipped cream"  
## [4] "Thick slices made from our homemade sourdough bread"   
## [5] "Two eggs, bacon or sausage, toast, and our ever-popular hash browns"

bfast = data.frame(hp\_name,hp\_price,hp\_desc)  
grep("toast", bfast$hp\_desc)

## [1] 5

grepl("toast",bfast$hp\_desc)

## [1] FALSE FALSE FALSE FALSE TRUE

sum(grepl("toast",bfast$hp\_desc))

## [1] 1

# Part 1: Restaurant Data from Baltimore!

**Note: one of the methods learned in the last unit will work with the the improper XML code that the URL below yields and the other will create an error. Use one to get the job done.**

*You have been hired by a restaurateur to do some research on Sushi restaurants in Baltimore.* *You have come across data on the web contained in the following XML file.*  
*Data:* [*https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Frestaurants.xml*](https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Frestaurants.xml)

1. Scrape the XML page for name, zipcode and city council district. (Use either the XML or rvest package.) <  
>>>>>>>>>DONE  
2. Make a dataframe with just those columns.   
>>>>>>>>>DONE  
3. Are there any Sushi restaurants in Baltimore? (Where the dataset is from.)  
 a. If so, can you estimate how many?

To solve question 3, what makes sense is to search the names column we just created for the word “sushi.” We should use “ignore.case = TRUE” and count the number of times the word sushi appears. We can study the data results.

There appears to be 9 occurences of the word “sushi” in the names column. Perhaps we should search for other possible sushi restaurant names. I appear to have discovered a very cool way of displaying text. The right pointing “pointer” repeated reveals interesting results as seen in the knitted .html file result. Candidly, that’s hard to believe that there are only 9 sushi restaurants in Baltimore. I think I need to dig deeper….somehow.

I substituted alternative names for “sushi” like tempura, hibachi, hibatchi, sashimi, to no avail. I went to Google and searched “How many sushi restaurants are there in Baltimore.” My first result was, “Top 10 sushi restaurants in Baltimore.” If there was a category “Description” I think this would make all the difference in the world. But, with the data available, it appears that there are 9 with the word sushi in the restaurant name. This is altogether unsatisfying as I’m virtually certain there are more. I’m going to add neighborhood to my results in a separate column. Chinatown may or may not be an indicator and at this point I’m not sure how it would be utilized.

At that point, it makes sense to count the number of occurrences of districts. This “may” be telling in that sushi may be found in “restaurant row” or in certain Asian neighborhoods or near certain businesses like “high tech” or the like. Let’s find out! Result: Not much help.

# 1. Scrape the XML page for name, zipcode and city council district. (Use either the XML or rvest package.)  
  
data <-getURL("https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Frestaurants.xml")  
doc <- xmlParse(data)  
#doc  
names <- xpathSApply(doc,"//name",xmlValue)  
zipcode <- xpathSApply(doc,"//zipcode",xmlValue)  
councildistrict <- xpathSApply(doc,"//councildistrict",xmlValue)  
neighborhood <- xpathSApply(doc,"//neighborhood",xmlValue)  
area = data.frame(names,zipcode,councildistrict)  
head(area)

## names zipcode councildistrict  
## 1 410 21206 2  
## 2 1919 21231 1  
## 3 SAUTE 21224 1  
## 4 #1 CHINESE KITCHEN 21211 14  
## 5 #1 chinese restaurant 21223 9  
## 6 19TH HOLE 21218 14

head(area$names)

## [1] "410" "1919" "SAUTE"   
## [4] "#1 CHINESE KITCHEN" "#1 chinese restaurant" "19TH HOLE"

#area$neighborhood  
length(grep("sushi",area$names, ignore.case = TRUE))

## [1] 9

length(grep("sashimi",area$names, ignore.case = TRUE))

## [1] 0

length(grep("asian",area$neighborhood, ignore.case = TRUE))

## [1] 0

#grepl("sushi",area$names)  
sum(grepl("sushi",area$names))

## [1] 0

which(grepl("sushi",area$names))

## integer(0)

head(area$names)

## [1] "410" "1919" "SAUTE"   
## [4] "#1 CHINESE KITCHEN" "#1 chinese restaurant" "19TH HOLE"

head(area$neighborhood)

## NULL

length(grep("sushi",area$names, ignore.case = TRUE))

## [1] 9

length(grep("sashimi",area$names, ignore.case = TRUE))

## [1] 0

length(grep("asian",area$neighborhood, ignore.case = TRUE))

## [1] 0

head(grepl("sushi",area$names))

## [1] FALSE FALSE FALSE FALSE FALSE FALSE

sum(grepl("sushi",area$names))

## [1] 0

which(grepl("sushi",area$names))

## integer(0)

## 4. Filter the dataframe for just downtown restaurants (Council District 11).

area\_11 <- area[area$councildistrict == "11",]   
head(area\_11)

## names zipcode councildistrict  
## 15 A TASTE OF CHINA 21202 11  
## 16 ABACROMBIE FINE FOODS 21201 11  
## 22 AKBAR RESTAURANT 21201 11  
## 29 BAY ATLANTIC CLUB 21212 11  
## 33 BEDROCK 21201 11  
## 34 BEDROCK (KITCHEN AREA) 21201 11

There are 277 total restaurants in District 11.

## 5. Are there any Sushi restaurants downtown? # research the “grep” function

length(grep("sushi",area\_11, ignore.case = TRUE))

## [1] 1

There appears to be one with the word sushi in the restaurant name. Once again, it’s unimaginable to me that there is only one sushi restaurant in downtown Baltimore. If there was a description of the business, I believe the search would be much more fruitful.

## 6. If so, estimate how many “Sushi” restaurants are in Downtown

From the data available, it would appear that there is only one sushi restaurant in Downtown Baltimore.

## 7. Make a barplot of the estimated number of restaurants (Sushi or otherwise) in each council.

unique(area$councildistrict)

## [1] "2" "1" "14" "9" "13" "7" "10" "5" "11" "6" "12" "3" "4" "8"

There appears to be 13 districts. I need to get the names of the restaurants AND the individual districts and then length (to count) and then once defined, use ggplot to plot a bar graph showing the number of restaurants in each district.

list <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13)  
#final\_sushi < list  
#Dist\_sushi <- data.frame(area$councildistrict)  
#Dist\_sushi <- data.frame(Dist\_sushi)  
for (list in 1:length(list)) {  
#grep(list, area$councildistrict == list)  
District\_sushi <- length(grep("sushi",area$names, ignore.case = TRUE, value = FALSE))  
#append(Dist\_sushi, District\_sushi)  
print(paste(District\_sushi))  
#append(District\_sushi, area$councildistrict)  
}

## [1] "9"  
## [1] "9"  
## [1] "9"  
## [1] "9"  
## [1] "9"  
## [1] "9"  
## [1] "9"  
## [1] "9"  
## [1] "9"  
## [1] "9"  
## [1] "9"  
## [1] "9"  
## [1] "9"

head(District\_sushi)

## [1] 9

#area\_11 <- area[area$councildistrict == "11",]

These names do not have the word sushi in them, but I’ll keep working on it to refine it. I must admit I’m stumped by this one. I need a match to a name in one column, based on the value in another column. Seems easy enough, but here we are. I feel like I should know this.