Computational Social Science

Scraping the web II

Dr. Thomas Davidson

Rutgers University

February 15, 2024

Plan

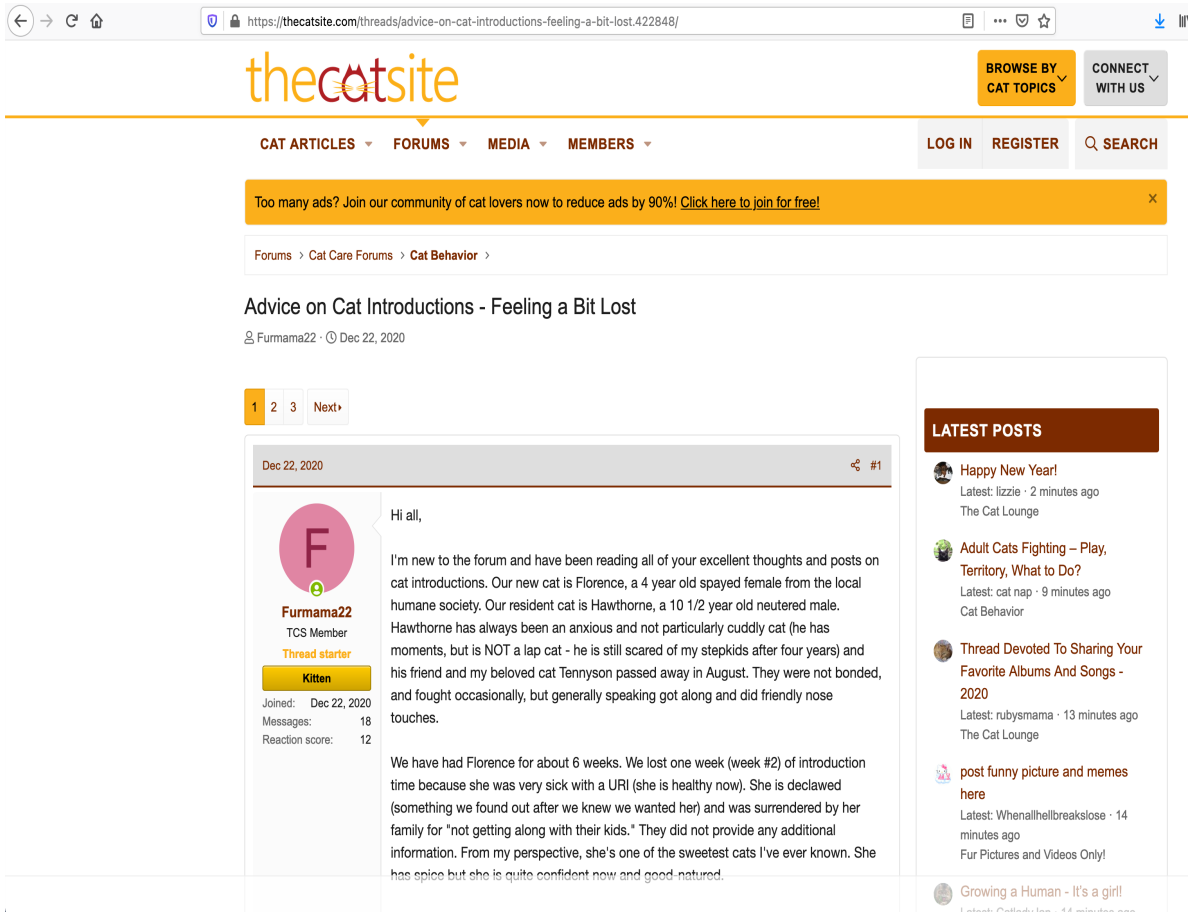
1. How to scrape a website in R, part II
2. Crawling websites using R
3. `selenium` and browser automation
4. Next week

How to scrape a web page

Using `rvest` to scrape HTML

```
library(rvest)
library(tidyverse)
library(stringr)
library(lubridate)
```

How to scrape a web page



How to scrape a web page

Using `rvest` to scrape HTML

We used `rvest` to read in this URL.

```
url <- "https://thecatsite.com/threads/advice-on-cat-introductions-feel
thread <- read_html(url)
```

How to scrape a web page

Creating a function to collect and store data

```
get.posts <- function(thread) {
  messages <- thread %>% html_nodes(".message-body") %>%
    html_text() %>% str_trim()
  users <- thread %>% html_nodes(".message-userDetails") %>%
    html_text() %>% str_trim() %>% str_split('\n') %>% map(1)
  timestamps <- thread %>% html_nodes(".u-concealed .u-dt") %>%
    html_attr("datetime") %>% ymd_hms(tz="EST")
  timestamps <- timestamps[-1] # remove first timestamp
  data <- as_tibble(cbind(messages, unlist(users), timestamps))
  colnames(data) <- c("message","user", "timestamp")
  return(data)
}
```

How to scrape a web page

Using the function

We then used this function to extract information from the forum.

```
results <- get.posts(thread)
tail(results)
```

```
## # A tibble: 6 x 3
##   message                                                                use
##   <chr>                                                                  <ch
## 1 "That's great, thank you \n pearl99\n! Our five minute visual~ Furm
## 2 "Furmama22 said:\n\n\n\nThanks so much for commenting \nA\n A~ Artl
## 3 "Thank you for your perspective! I'll keep that in mind."      Furm
## 4 "Furmama22 said:\n\n\n\nThat's great, thank you \n pearl99\n!~ pea
## 5 "Furmama22 said:\n\n\n\nThank you \nC\n calicosrspecial\nand ~ cal
## 6 "Furmama22 said:\n\n\n\nWhen he does need to go in the room, ~ Mam
```

How to scrape a web page

Pagination

The next step is to figure out how we can navigate the different pages of the thread. Inspection of the HTML shows the CSS element `pageNav-jump` contains the relevant information.

```
thread %>% html_nodes(".pageNav-jump")
```

```
## {xml_nodeset (2)}
## [1] <a href="/threads/advice-on-cat-introductions-feeling-a-bit-lost
## [2] <a href="/threads/advice-on-cat-introductions-feeling-a-bit-lost
```

How to scrape a web page

Pagination

In this case I want both the links *and* the descriptions.

```
links <- thread %>% html_nodes(".pageNav-jump") %>%
  html_attr("href")
desc <- thread %>% html_nodes(".pageNav-jump") %>%
  html_text()
pagination.info <- data.frame(links, desc) %>%
  filter(str_detect(desc, "Next")) %>% distinct()
head(pagination.info)
```

```
##                                                                                li
## 1 /threads/advice-on-cat-introductions-feeling-a-bit-lost.422848/pag
```

How to scrape a web page

Pagination

We can then use the base URL to get the link to the next page.

```
base <- "https://thecatsite.com"
next.page <- paste(base, pagination.info$links, sep = '')
print(next.page)
```

```
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
```

How to scrape a web page

Pagination

Let's verify this works by using the `get.posts` function.

```
results <- get.posts(read_html(next.page))
results[1:5,]
```

```
## # A tibble: 5 x 3
##   message                                                              use
##   <chr>                                                                <ch
## 1 "Thank you all for responding! Merry Christmas to all of you!~ Fur
## 2 "Sounds like a reason to be merry to me!"                            Mam
## 3 "Well I suppose it's always one step forward two steps back. ~ Fur
## 4 "AWWWWWWWWW! She is adorable! And that really wasn't even a w~ Mam
## 5 "Thank you!"                                                         Fur
```

How to scrape a web page

Pagination function

```
get.next.page <- function(thread){
  links <- thread %>% html_nodes(".pageNav-jump") %>%
    html_attr("href")
  desc <- thread %>% html_nodes(".pageNav-jump") %>%
    html_text()
  pagination.info <- data.frame(links, desc) %>%
    filter(str_detect(desc, "Next")) %>% distinct()
  base <- "https://thecatsite.com"
  next.page <- paste(base, pagination.info$links, sep = '')
  return(next.page)
}
get.next.page(thread)
```

```
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
```

How to scrape a web page

Testing the pagination function

We can easily use this function to paginate. In this case we use `get.next.page` to get the link to page 2, read the HTML for page 2, then use `get.next.page` to extract the link to page 3.

```
thread.2 <- read_html(get.next.page(thread))
page.3 <- get.next.page(thread.2)
page.3
```

```
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
```

How to scrape a web page

Testing the pagination function

What happens when we run out of pages? In this case there is no link to the next page. The `get.next.page` function does not produce an error, but only returns the base URL.

```
get.next.page(read_html("https://thecatsite.com/threads/advice-on-cat-i
```

```
## [1] "https://thecatsite.com"
```

How to scrape a web page

Improving the function

```
get.next.page <- function(thread){
  links <- thread %>% html_nodes(".pageNav-jump") %>%
    html_attr("href")
  desc <- thread %>% html_nodes(".pageNav-jump") %>%
    html_text()
  pagination.info <- data.frame(links, desc) %>%
    filter(str_detect(desc, "Next")) %>% distinct()
  if (dim(pagination.info)[1] == 1) {
    base <- "https://thecatsite.com"
    next.page <- paste(base, pagination.info$links, sep = '')
  return(next.page)
    } else {
    return("Final page")
  }
}
```

How to scrape a web page

Testing the pagination function

We now get this message when we try to paginate on the final page.

```
get.next.page(read_html("https://thecatsite.com/threads/advice-on-cat-i
```

```
## [1] "Final page"
```

How to scrape a web page

Paginate and scrape

```
paginate.and.scrape <- function(url){
  thread <- read_html(url)
  posts <- get.posts(thread)
  next.page <- get.next.page(thread)

  while (!str_detect(next.page, "Final page"))
  {
    print(next.page)
    thread <- read_html(next.page)
    posts <- rbind(posts, get.posts(thread))
    next.page <- get.next.page(thread)
    Sys.sleep(0.5) # wait 0.5 seconds
  }
  return(posts)
}
```

How to scrape a web page

Paginate and scrape

```
full.thread <- paginate.and.scrape(url)
```

```
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
## [1] "https://thecatsite.com/threads/advice-on-cat-introductions-feel
```

How to scrape a web page

Crawling a website

- Now we have a function we can use to paginate and scrape the data from threads on the website
- The next goal is to write a crawler to traverse the website and retrieve information from all of the threads we are interested in.
- Fortunately, these threads are organized in a similar way
  - Each page contains 20 threads and links to the next page

How to scrape a web page

Crawling a website

```
get.threads <- function(url) {
  f <- read_html(url)
  title <- f %>% html_nodes(".structItem-title") %>%
    html_text() %>% str_trim()
  link <- f %>% html_nodes(".structItem-title a") %>%
    html_attr("href")  %>% str_trim()
  link <- data.frame(link)
  link <- link %>% filter(str_detect(link, "/threads/"))
  threads <- data.frame(title, link)
  return(threads)
}
```

How to scrape a web page

Crawling a website

```
forum.url <- "https://thecatsite.com/forums/cat-behavior.5/"

threads <- get.threads(forum.url)
```

How to scrape a web page

Crawling a website

```
print(threads$title)
```

```
##  [1] "Featured\nPooping near the box"
##  [2] "Are they playing or not??"
##  [3] "Resident cat charge swatting new kitten repeatedly"
##  [4] "New cat is neutered, male resident cat is not. Will they get a
##  [5] "Introductions while grieving the loss of bonded cat"
##  [6] "Curious about single kitten syndrome"
##  [7] "Pees on bed early morning"
##  [8] "Sweet cat with redirected aggression, please help: euthanasia
##  [9] "Older cat hurts new kitten"
## [10] "15 week kitten and 6yr old cat...confused!"
## [11] "Dealing with cat's aggression towards people"
## [12] "16 week kittens"
## [13] "Curious behaviour"
## [14] "Unsuccessful introduction, now reintroducing"
## [15] "Kitten to multi-cat household introduction not progressing"
## [16] "Vocal destructive cat"
```

How to scrape a web page

Crawling a website

```
print(threads$link)
```

```
##  [1] "/threads/pooping-near-the-box.462889/"
##  [2] "/threads/are-they-playing-or-not.462894/"
##  [3] "/threads/resident-cat-charge-swatting-new-kitten-repeatedly.45
##  [4] "/threads/new-cat-is-neutered-male-resident-cat-is-not-will-the
##  [5] "/threads/introductions-while-grieving-the-loss-of-bonded-cat.4
##  [6] "/threads/curious-about-single-kitten-syndrome.462720/"
##  [7] "/threads/pees-on-bed-early-morning.461994/"
##  [8] "/threads/sweet-cat-with-redirected-aggression-please-help-euth
##  [9] "/threads/older-cat-hurts-new-kitten.462789/"
## [10] "/threads/15-week-kitten-and-6yr-old-cat-confused.462703/"
## [11] "/threads/dealing-with-cats-aggression-towards-people.461595/"
## [12] "/threads/16-week-kittens.462695/"
## [13] "/threads/curious-behaviour.462798/"
## [14] "/threads/unsuccessful-introduction-now-reintroducing.462794/"
## [15] "/threads/kitten-to-multi-cat-household-introduction-not-progre
## [16] "/threads/vocal-destructive-cat.462787/"
```

How to scrape a web page

Crawling a website

Iterating over the first 5 pages of threads and collecting all threads in each set.

```
pagination.max <- 5
url <- forum.url
results <- tibble()

for (p in 1:pagination.max) {
  threads <- get.threads(url)
  for (t in 1:dim(threads)[1]) {
    page.url <- paste(base, threads$link[t], sep = '')
    new.results <- paginate.and.scrape(page.url)
    new.results$threads <- threads$title[t]
    results <- bind_rows(results, new.results)
  }
  url <- get.next.page(read_html(url))
}
```

How to scrape a web page

Storing the results

The results should consist of a few thousand messages and associated metadata. Save the results of this crawl to as a CSV file.

```
write_csv(results, "../data/cat_crawl.csv")
```

How to scrape a web page

Data storage

- If you try to collect all the data you need before saving it, you run the risk of data loss if you script crashes
  - This risk increases as you collect more data
    - More memory on your computer is being used
    - Increased likelihood of encountering anomalies that cause errors
- Reasonable solutions
  - Continuously save results to disk (e.g. concatenate each thread to a CSV)
  - Store results in chunks (e.g. each thread in a new CSV)

How to scrape a web page

Data storage

- A more robust solution
  - Write output to a relational database
    - This helps to organize the data and makes it easier to query and manage, particularly with large datasets
    - I recommend PostgreSQL, a free open-source, SQL-compatible relational database

How to scrape a web page

Data storage

- If collecting a lot of data, I recommend use a server to run the code and to store scraped data
- Requirements
  - Access to a server
    - But most universities have free computing clusters
  - Command line knowledge
  - Database knowledge
- It is beyond the scope of this class to cover this material, but I highly recommend you develop this infrastructure if you continue to work in this area

How to scrape a web page

Logging

- Log the progress of your webscraper
  - Simple option:
    - Print statements in code
  - Better option:
    - Use a log file
  - To keep yourself updated:
    - Use a Slack App to send yourself messages

Advanced webscraping

Javascript and browser automation

- Many websites use Javascript, which cause problems for web-scrapers as it cannot directly be parsed to HTML
- We can get around this by doing the following
  - Automatically to open a browser (e.g. Google Chrome)
  - Load a website in the browser
  - Read the HTML from the browser into R

Advanced webscraping

Selenium

- Selenium WebDriver and the package `RSelenium` (https://github.com/ropensci/RSelenium) is the most popular approach
- **However**, `RSelenium` requires a complicated set up using a Docker container
- It will be easier to use `selenium` in Python then read the data into R
  - https://python-bloggers.com/2020/07/rvspython-3-setting-up-selenium-limitations-with-the-rselenium-package-getting-past-them/

Advanced webscraping

Running Selenium using Python

This Python code uses selenium to open up a Chrome browser, visit a website, and collect the HTML. It then closes the browser.

```
from selenium import webdriver
from time import sleep
driver = webdriver.Chrome()
driver.get('https://www.sociology.rutgers.edu')
html = driver.page_source
sleep(15)
driver.close()
```

This will only work if the Chrome driver has been downloaded and is in your PATH. See https://chromedriver.chromium.org/getting-started

Advanced webscraping

Using reticulate to obtain results using R

We can use `reticulate` to pass objects from Python to R. Note the Python objects are shown in the `Environment` tab.

```
library(reticulate)
html.text <- read_html(py$html) %>% html_text()
```

Advanced webscraping

Inspecting the results in R

Reticulate allowed us to run a Python script then pass the results to R. We can then use the same commands as above to process it.

```
print(substr(html.text, 1, 35))
```

Advanced webscraping

Browser automation

- Using browser automation you can perform other tasks that simulate a human user
  - Clicking buttons
  - Filing forms
  - Logging in
  - Saving images

Next week

- Online surveys and experiments
- Intro to RShiny
- Homework 2 released