Computational Social Science

Scraping the web I

Dr. Thomas Davidson

Rutgers University

September 29, 2025

Plan

1. Ethics and data science
2. Introduction to webscraping
3. When to use it

Ethics and data science

Four ethical principles in social research

- *Respect for persons*
  - Treating people as autonomous and honoring their wishes
- *Beneficence*
  - Understanding risks and benefits; finding the right balance
- *Justice*
  - Even distribution of the risks and benefits
- *Respect for law and public interest*
  - Extends beyond research participants to other stakeholders
  - Compliance and transparency-based accountability

Ethics and data science

Two ways of thinking about research ethics

- *Consequentialism*
  - Focus on the consequences of research
  - Ends
- *Deontology*
  - Consideration of ethical duties, irrespective of consequences
  - Means
- Salganik argues that both perspectives most useful when combined

Ethics and data science

Case study

- Researchers at Rutgers decide to use information from Reddit to help improve student services
- They use Reddit API to collect the complete posting history of all users who posted on r/rutgers
- A small group of these users is sent a survey. They are also asked for consent to merge their Reddit history and confidential student records
- The survey results are used to build a statistical model to predict the race, gender, sexual orientation, school year, major, and GPA of *all* r/rutgers posters
- This information is used to study how the content of posts varies across different groups of students

Ethics and data science

Discussion

- How might this study violate some of the four ethical principles?
- What issues arise when thinking about this study from a consequentialist or deontlogical perspective?
- Could we design the study in a more ethical way?

Ethics and data science

Four challenges in digital research

- Informed consent
  - When is it practical to get consent to participate?
  - When is it acceptable to proceed without consent?
- Managing informational risk
  - Risks of disclosure of personal information
  - Anonymization is often imperfect
- Privacy
  - What information is public or private?
  - Context-relative informational norms
- Ethical decisions and uncertainty
  - Minimal risk standard
  - Power analysis

What is web-scraping?

Terminology

- Web-scraping is a method to collect data from websites
  - We use the code underlying a webpage to collect data (**scraping**)
  - The process is then repeated for other pages on the same website in an automated fashion (**crawling**)

What is web-scraping?

Challenges

- Different websites have different structures, so a script used to scrape one website will likely have to be changed to scrape another
- Websites can be internally inconsistent, making them difficult to scrape
- Some websites are easier to crawl than others
- Some websites limit or prohibit scraping

When should I use it?

Commercial use cases

- Search engines
  - Google scrapes websites to create a searchable index of the internet
- Price comparison
  - Kayak scrape airlines to compare flight prices, other websites do the same for hotels and rental cars
- Recruitment
  - Recruitment companies scrape LinkedIn to get data on workers

When should I use it?

Social scientific use cases

- Web-scraping is a useful tool to collect data from websites without APIs
  - Large social media platforms and other sites have APIs but smaller websites do not
    - Local newspapers, forums, small businesses, educational institutions, etc.
- Often we want to collect data from a single website
  - e.g. All posts written on a forum
- Sometimes we might want to collect data from many websites
  - e.g. All schools in a school district

When should I use it?

Ethical and legal considerations

No Robots, Spiders, or Scrapers: Legal and Ethical Regulation of Data Collection Methods in Social Media Terms of Service

Casey Fiesler,[1]* Nathan Beard,[2] Brian C. Keegan[1]
[1]Department of Information Science, University of Colorado Boulder
[2]College of Information Studies, University of Maryland

Abstract

Researchers from many different disciplines rely on social media data as a resource. Whereas some platforms explicitly allow data collection, even facilitating it through an API, others explicitly forbid automated or manual collection processes. A current topic of debate within the social computing research community involves the ethical (or even legal) implications of collecting data in ways that violate Terms of Service (TOS). Using a sample of TOS from over one hundred social media sites from around the world, we analyze TOS language and content in order to better understand the landscape of prohibitions on this practice. Our findings show that

opportunities for digital social research, with new ways of collecting, analyzing, and visualizing data; it also allows for ordered collection, so that messy online data can become usable, well-ordered data sets (Marres and Weltevrede 2013).

However, even when data collection is possible technically, sometimes it is prohibited by terms of service (TOS), which restrict certain behaviors and uses of a site. Whether it is permissible, or ethical, for researchers to violate TOS in the course of collecting data is currently an open question within the social computing research community (Vaccaro et al. 2015; Vitak, Shilton, and Ashktorab 2016).

When should I use it?

Ethical and legal considerations

- Fiesler, Beard, and Keegan (2020)s review the legal cases related to web-scraping and analyze website terms of service
  - "In short, it is an unsettled question as to whether it is explicitly illegal (or even a criminal act) to violate TOS."
  - No academic or journalist has ever been prosecuted for violating a website terms of service to collect data for research
- They analyze terms of service of over 100 social media websites
  - Terms of service are ambiguous, inconsistent, and lack context

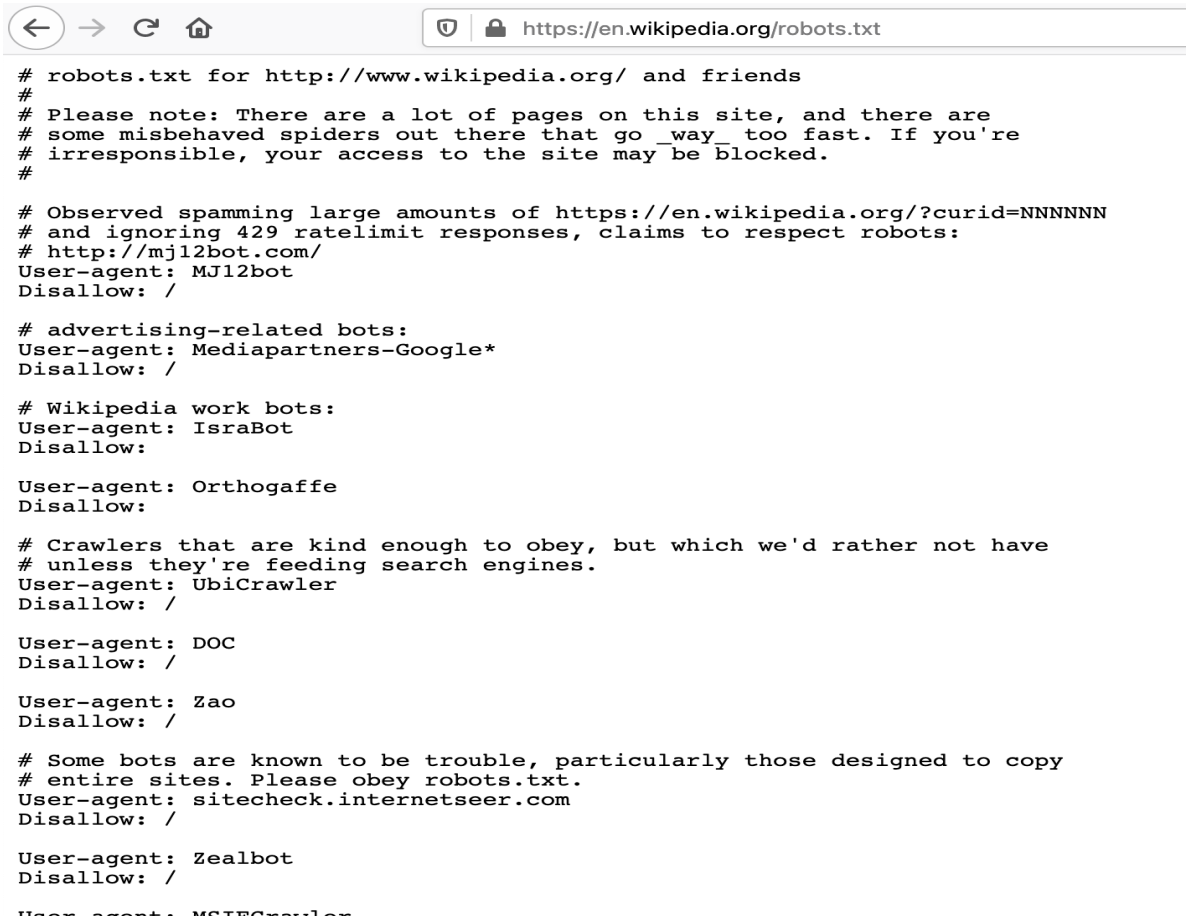
When should I use it?

Best-practices

- Only scrape publicly available data
  - i.e. You can access the page on the web without logging in
- Do not scrape copyright protected data
- Try not to violate website terms of service
- Do not burden the website
  - Limit the number of calls you make (similar to rate-limiting in APIs)
- Avoid using the data in a way that may interfere with business
  - i.e. Don't copy valuable data from a small business and share it on Github

How to scrape a web page

Start by looking up "robots.txt''



How to scrape a web page

Decoding `robots.txt`

- `User-agent` = the name of the scraper
  - `*` = All scrapers
- `Allow: /path/` = OK to scrape
- `Disallow: /path/` = Not OK to scrape
  - `Disallow: /` = Not OK to scrape any pages
- `Crawl-Delay: N` = Wait `N` miliseconds between each call to the website

How to scrape a web page

Exercise

- Choose a website
- Locate the robots.txt file
  - Does the website allow webscraping?
  - Are there any restrictions on which pages can be accessed?

How to scrape a web page

Terminology

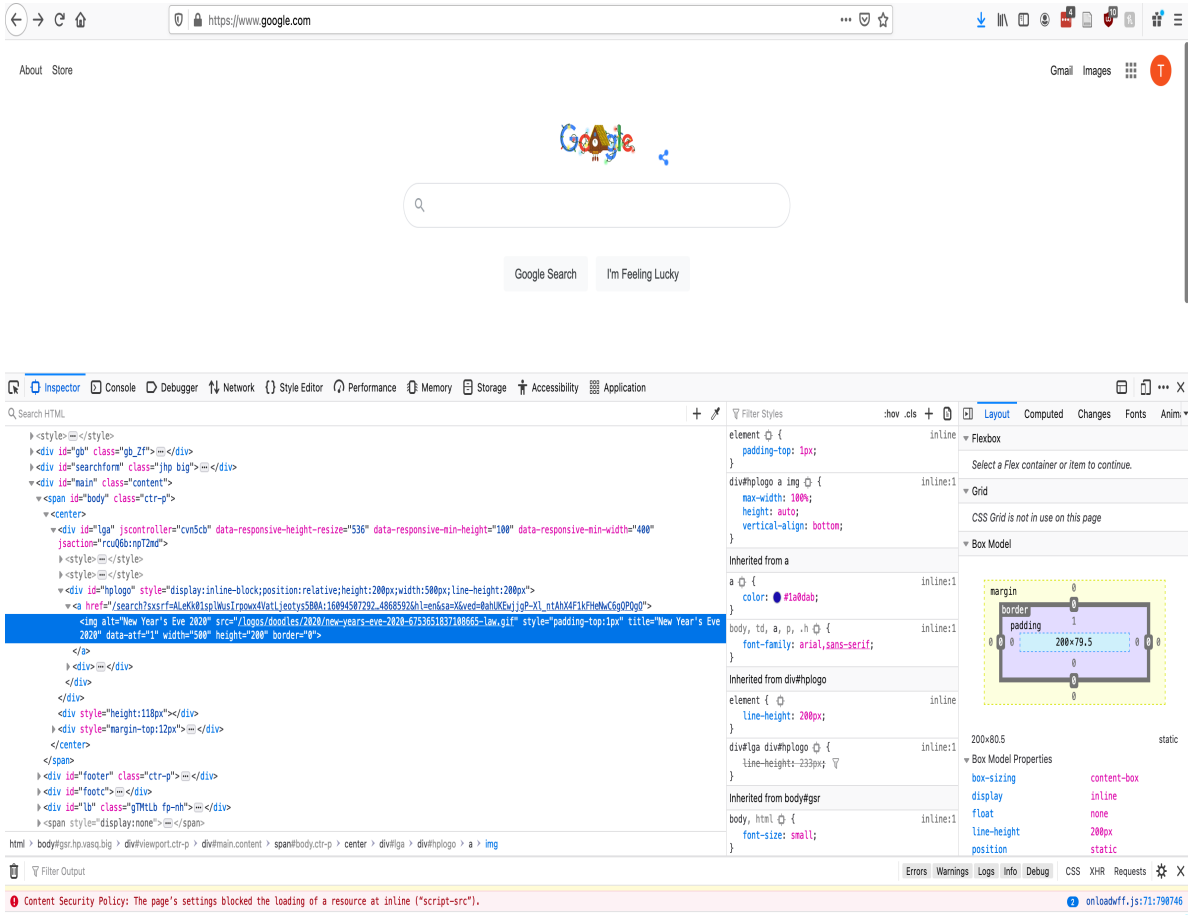
- A web-page is loaded using a **URL** (Uniform Resource Locator)
- The underlying code we are interested in is usually **HTML** (Hypertext Markup Language)
- Many websites use **CSS** (Cascading Style Sheets) to structure HTML
  - This will help us to find what we are interested in
    - See https://flukeout.github.io/ for an interactive tutorial on using CSS selectors
    - Chrome Plugin to help find CSS elements: https://selectorgadget.com/

How to scrape a web page

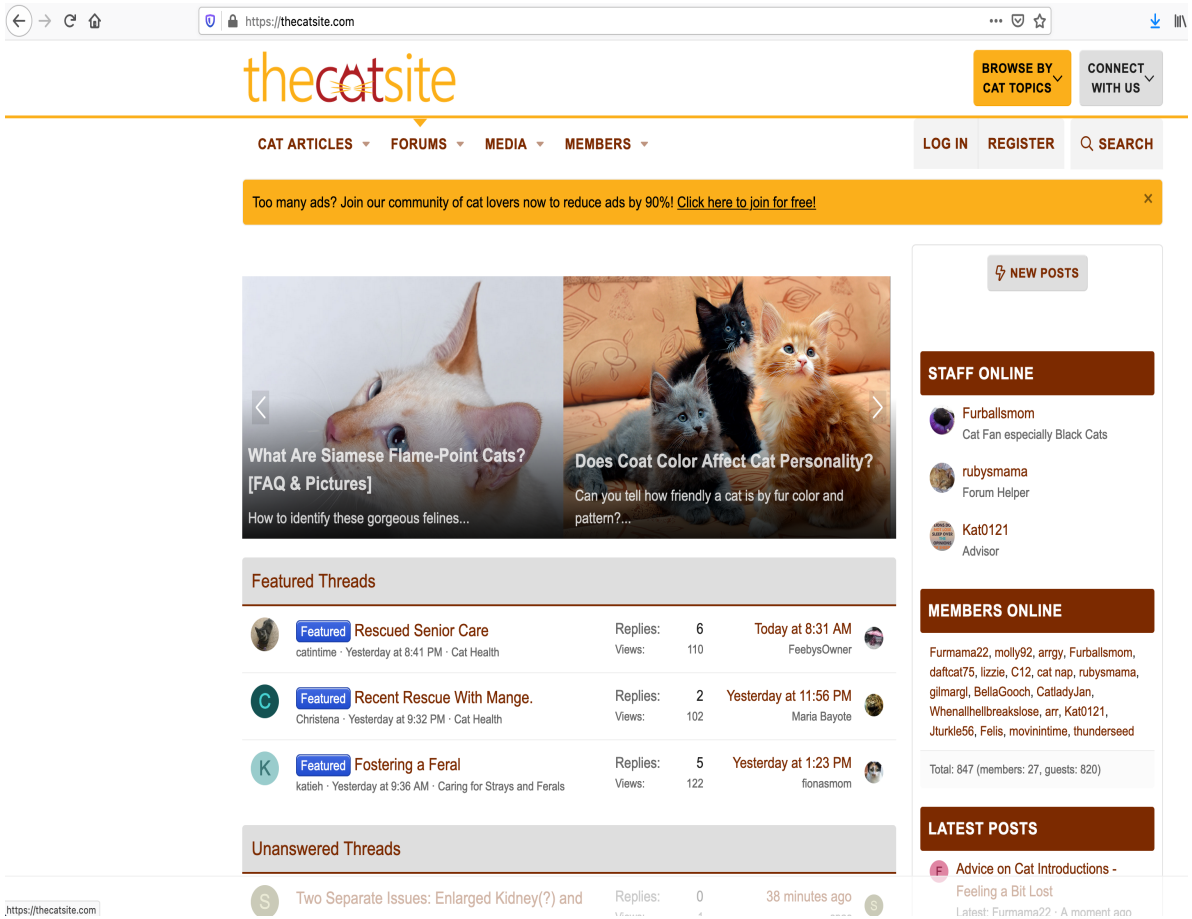
Inspecting HTML

- Open up a website and right click on any text or image on the screen
  - You should see an option `Inspect Element`
  - This will allow you to see the code used to generate the page

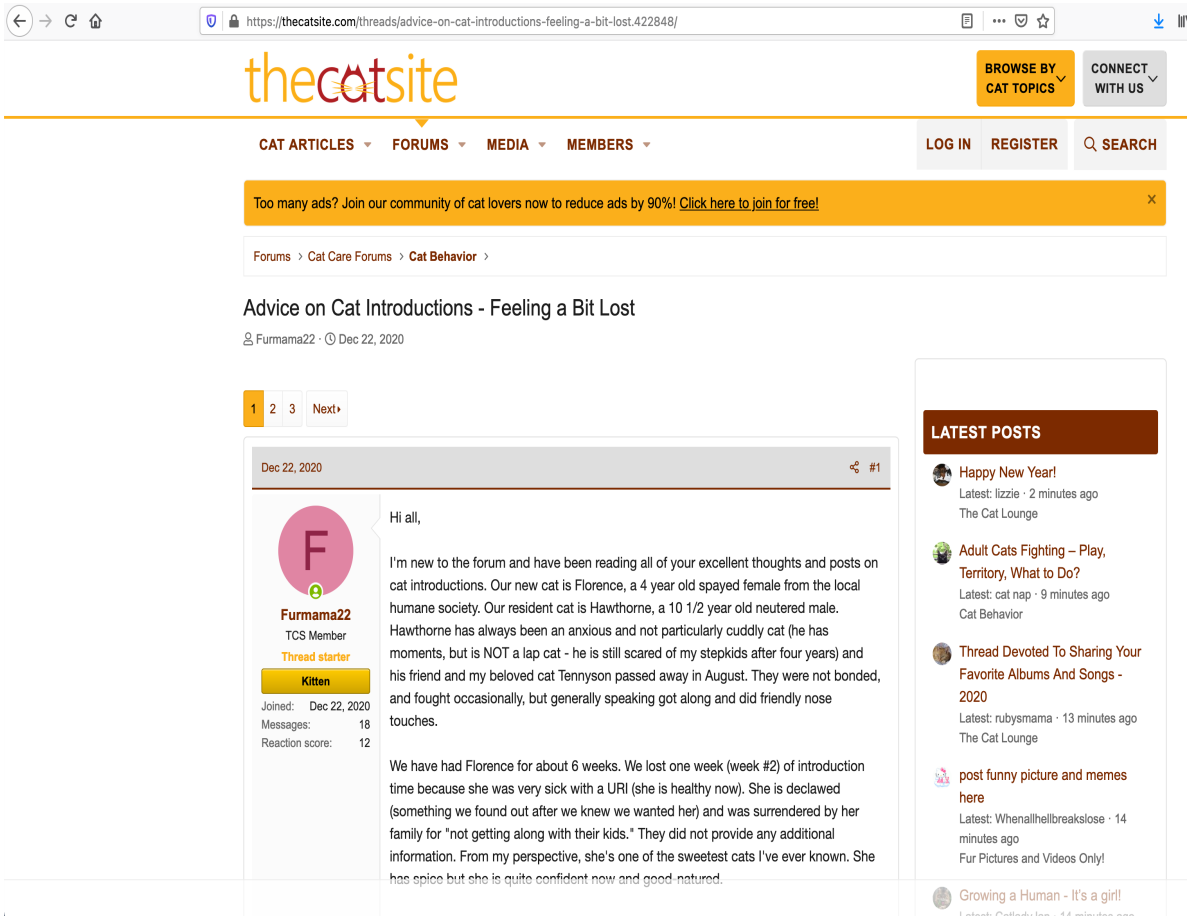
How to scrape a web page



How to scrape a web page



How to scrape a web page



How to scrape a web page

Using `rvest` to scrape HTML

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

How to scrape a web page

Using `rvest` to scrape HTML

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

How to scrape a web page

Using `rvest` to scrape HTML

```
class(thread)
```

```
## [1] "xml_document" "xml_node"
```

```
print(thread)
```

```
## {html_document}
## <html id="XF" lang="en-US" dir="LTR" data-xf="2.3" data-app="public"
## [1] <head>\n<meta http-equiv="Content-Type" content="text/html; char
## [2] <body data-template="thread_view">\n\n<div class="p-pageWrapper"
```

How to scrape a web page

Collecting messages

First, we parse the HTML to obtain the text of each message on the page. Here we use the CSS selector `.message-body`, which selects all elements with class `message-body`. The `html_nodes` function in `rvest` allows us to retrieve these nodes.

```
message.data <- thread %>% html_nodes(".message-body")
print(message.data[2])
```

```
## {xml_nodeset (1)}
## [1] <article class="message-body js-selectToQuote"><div itemprop="te
```

How to scrape a web page

Collecting messages

Next we use `html_text()` to extract the text from the HTML.

```
messages <- thread %>% html_nodes(".bbWrapper") %>%
  html_text() %>% str_squish()
messages[1]
```

```
## [1] "Hi all, I'm new to the forum and have been reading all of your
```

How to scrape a web page

Collecting messages

As expected, there are twenty messages.

```
print(length(messages))
```

```
## [1] 20
```

```
print(substr(messages[20], 1, 250)) # print a substring
```

```
## [1] "Furmama22 said: When he does need to go in the room, I'll go wi
```

How to scrape a web page

Getting user names

Next we collect the name of each user using the same logic. User information is found by parsing the `.message-userDetails` node.

```
users <- thread %>% html_nodes(".message-userDetails") %>%
  html_text() %>% str_trim()
print(length(users))
```

```
## [1] 20
```

```
class(users)
```

```
## [1] "character"
```

```
users[1]
```

```
## [1] "Furmama22\n\t\t\tTCS Member\n\t\t\tAdult Cat"
```

How to scrape a web page

Getting user names

Let's add some more elements to the pipe to extract the user name from this string. Note how the elements in the string returned in the previous chunk are separated by the newline symbol (`\n`).

```
users <- thread %>% html_nodes(".message-userDetails") %>%
  html_text() %>% str_trim() %>% str_split('\n')
class(users)
```

```
## [1] "list"
```

```
users[1]
```

```
## [[1]]
## [1] "Furmama22"          "\t\t\tTCS Member" "\t\t\tAdult Cat"
```

How to scrape a web page

Getting user names

The final step is to get the name from each list. This can be done by using the `map` command.

```
users <- thread %>% html_nodes(".message-userDetails") %>%
  html_text() %>% str_trim() %>% str_split('\n') %>% map(1)
class(users)
```

```
## [1] "list"
```

```
users[1:2]
```

```
## [[1]]
## [1] "Furmama22"
##
## [[2]]
## [1] "calicosrspecial"
```

How to scrape a web page

Collecting timestamps

Finally, we also want to get the time-stamp of each message. While the forum only displays dates, we can actually get the full timestamp. What's the problem here?

```
dates <- thread %>% html_nodes("time.u-dt")
print(dates[1])
```

```
## {xml_nodeset (1)}
## [1] <time class="u-dt" dir="auto" datetime="2020-12-22T11:26:12-0800
```

```
length(dates)
```

```
## [1] 27
```

How to scrape a web page

Collecting timestamps

I went back to the HTML and found this CSS selector `.u-concealed .u-dt` is selected instead. It returns the datetime for each post in the thread, along with the date time at the top indicating when the thread was created.

```
dates <- thread %>% html_nodes(".u-concealed .u-dt")
length(dates)
```

```
## [1] 21
```

```
dates[1]
```

```
## {xml_nodeset (1)}
## [1] <time class="u-dt" dir="auto" datetime="2020-12-22T11:26:12-0800
```

```
class(dates[1])
```

```
## [1] "xml_nodeset"
```

How to scrape a web page

Collecting timestamps

Each HTML node contains several different attributes related to the time. In this case we can select the `datetime` attribute using the `html_attr` function.

```
dates <- thread %>% html_nodes(".u-concealed .u-dt") %>% html_attr("dat
dates[1]
```

```
## [1] "2020-12-22T11:26:12-0800"
```

```
class(dates[1])
```

```
## [1] "character"
```

How to scrape a web page

Collecting timestamps

Finally, its often useful to clean up timestamps. We can do this using the `lubridate` package. In this case we extract the year, month, day, hour, minutes, and seconds, converted to EST. The result is a special type of object used to represent dates and times.

```
library(lubridate)
dates <- dates %>% ymd_hms(tz = "EST")
dates[1]
```

```
## [1] "2020-12-22 14:26:12 EST"
```

```
class(dates)
```

```
## [1] "POSIXct" "POSIXt"
```

How to scrape a web page

Putting it all together

```
length(users)
```

```
## [1] 20
```

```
class(users)
```

```
## [1] "list"
```

```
length(messages)
```

```
## [1] 20
```

```
class(messages)
```

```
## [1] "character"
```

```
length(dates)
```

```
## [1] 21
```

```
class(dates)
```

```
## [1] "POSIXct" "POSIXt"
```

How to scrape a web page

Putting it all together

```
data <- as_tibble(cbind(messages, unlist(users), dates[-1]))
colnames(data) <- c("message", "user", "timestamp")
head(data)
```

```
## # A tibble: 6 x 3
##   message                                                          use
##   <chr>                                                            <ch
## 1 "Hi all, I'm new to the forum and have been reading all of yo~ Fur
## 2 "Furmama22 said: Hi all, I'm new to the forum and have been r~ cal
## 3 "Thank you SO much for taking the time to reply. I really rea~ Fur
## 4 "I don't think I can add a thing to C calicosrspecial 's advi~ Mam
## 5 "Thanks so much for your thoughts and comments! And thank you~ Fur
## 6 "You certainly came to the right place! And, in my experience~ Mam
```

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_squish()
  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 can now easily run all the code to extract information using a single function call:

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

```
## # A tibble: 6 x 3
##   message                                                          use
##   <chr>                                                            <ch
## 1 "Hi all, I'm new to the forum and have been reading all of yo~ Fur
## 2 "Furmama22 said: Hi all, I'm new to the forum and have been r~ cal
## 3 "Thank you SO much for taking the time to reply. I really rea~ Fur
## 4 "I don't think I can add a thing to C calicosrspecial 's advi~ Mam
## 5 "Thanks so much for your thoughts and comments! And thank you~ Fur
## 6 "You certainly came to the right place! And, in my experience~ Mam
```

Next lecture

- Webscraping II
  - Systematically collecting data across multiple threads and pages