Lab5a_Scraping

Lab Purpose

This lab is designed to help you develop your scraping skills. It will also help us get data that we will use later when we explore text analysis.

Our focus will be on Emily Dickinson's poetry. In order to bring her poetry into R, we can webscrape the poems from Wikipedia. There is a separate Wikipedia page for each of Dickinson's poems—she wrote over 1,500! How can we efficiently search across all these pages to get the text from each poem into R?

We'll first web scrape a Wikipedia page that contains a *table* listing links to all her poems. Then we'll use data from that table to loop through each of the linked pages to scrape the *text* of the poems.

The packages for this lab include *tidyverse*, *rvest* (for general scraping), *robotstxt* (checking `paths_allowed()`), and *purrr* (to `pluck()` a single element from a list). If you expect to do a lot of scraping in the future, you should also check out the *polite* package.

Review

In the class notes/examples, we walked through the following steps to scrape and print Dickinson's *September's Baccalaureate* using html_text (and not html_table):

```
# 1. Identify page where poem is listed
sep_bac_url <- "https://en.wikisource.org/wiki/September%27s_Baccalaureate"

# 2. Confirm bots are allowed to access the page
paths_allowed(sep_bac_url)

# 3. Get poem text
sep_bac_text <- sep_bac_url %>%
```

```
  read_html() %>%
  # a. Get list of "div p" elements on the page
  html_elements("div p") %>%
  # b. `Pluck` poem from list and grab text
  pluck(1) %>%
  html_text()

# 4. Print poem
cat(sep_bac_text)
```

For this lab, we'll need to modify and repeat this process many times to scrape *all* (or at least some) of Emily Dickinson's poems available on Wikipedia!

(Realistic note: Ideally, you'd all scrape ALL the poems in this lab. However, this process takes a LONG time, and there are some details to figure out with poems that cause issues - more time than we have in lab. So, our REALISTIC goal is for each of you to scrape between 5-10-20 poems as practice scraping (getting 1 is the minimum, since you already have code for that!), and then I'll provide a script/the full data for our reference later. For the practice set due this week, you are scraping individual pages, not a set like this.)

1 - Algorithmic thinking

This Wikipedia page contains a table listing Emily Dickinson's poems. There is a separate Wikipedia page for (almost) every one of Dickinson's poems. Our goal is ultimately to create a dataframe with the title and text of every poem linked in the table. How can we efficiently search across all these pages to get the text from each poem into R? Let's walk through the process.

> part a - Skim Web Scraping 101, courtesy of rvest.tidyverse.org, focusing particularly on the section on Extracting Data (text, attributes, and tables). Feel free to take some notes below if desired.

Notes:

> part b - Revisit the List of Emily Dickinson's poems and recall how you scraped a single poem already (*September's Baccalaureate* above). Broadly speaking, what are the things you need to do to get to our final goal? Don't worry about the specific details yet—ignore the functions and arguments you'll need in R and don't worry about the particular order of steps for now.

You might call writing out these steps the "pseudocode". You're not coding, you're figuring out the steps you need to take. Later, we figure out the code to enact the steps.

Solution:

> part c - One of your steps likely involved scraping the table that lists Emily Dickinson's poems. Use the code chunk below to do that, creating a dataframe called `poem_table`. Be sure to check that bots are allowed first, and make sure the column names of your dataframe are user-friendly or "clean".

(Hint: remember `janitor::clean_names()` and/or you can rename the necessary columns yourself).

Solution:

Another important step is getting a list of all the URLs so we can iterate through each linked page to scrape the poem. There are a couple approaches we might take to do this: piecing the URLs together ourselves, or scraping the URLs from the table. Click through five or so poems in the table to see the pages that contain the text of each poem.

> part d - What do you notice about the URLs for the pages that contain the poem text? Can you identify a pattern in the URLs? Are there links that fall outside that pattern? How might we piece together the URLs ourselves? What did you learn from Web Scraping 101 that would allow us to scrape the URLs from the table directly?

Solution:

> part e - Choose one of the two approaches described before part d to create a data frame that contains all the links we need to iterate through and the link titles. Verify you only have full links (i.e., starting with "https:"), then join that dataframe with your `poem_tables` dataframe.

Solution:

2 - Poem Scraping

Ideally, as mentioned above, we would now iterate to scrape all the poems.

We have our table of poems, we have the corresponding links available, and we know how to scrape a poem from a single webpage (see example above where we scraped *September's Baccalaureate*).

The next major task is figuring out how to iterate through (all) the poems. Remember, the final product should be a dataframe with at least two columns: the title of the poem and the text of the poem. To do this, we will pre-allocate space in our data frame (a new column called `text`), and use a `for()` loop to iterate through the URLs and fill the `text` column with the poem text as we scrape each poem.

IMPORTANT:

You should develop and test your code on a small subset of pages (e.g., if really aiming for all of them, one URL, 5 URLs, 20 URLs, 50 URLs) and check for errors or oddities each time before scaling up. Given the number of poems and the delay time between hits to the site, it will take a (very) long time to run the code on the full set of ~1800 links, so don't do that until you are absolutely sure your code is producing the output you expect.

For our lab, if you can get 1 URL, 5 URLS, and then 10 URLS to work, that's great! You're seeing how the process would work. And that's plenty for the amount of time we have.

Some links will not work (links don't exist or links exist but poem has been removed). Typically, when an error is encountered, the code will break and quit without producing output—not ideal! Instead, we will use `tryCatch()` to produce alternative output when a link doesn't work. This will allow us to continue on to the next link without breaking the code. For more on `tryCatch()`, check out this chapter on Handling conditions.

Some potentially problematic or wonky poems:

- *A narrow Fellow in the Grass*
- *Alter! When the Hills do*
- *If I can stop one Heart from breaking*
- *I never lost as much but twice,*
- *The earth has many keys,*
- *The Himmaleh was known to stoop*

> Here is some code to get you started. Remember, we're only aiming to get to 10-20 poems as part of the lab.

If you do get it set up for all the poems, when you are sure everything finally works, replace `poem_table[seq_len(n_links), ]` with `poem_table` to be able to run the code for all poems (but don't run it yet!).

```
# Identify number of iterations (start with 1, 5, 10, etc.)
n_links <- _CODE_TO_SET_VALUE_

# Pre-allocate space in dataframe for poem text
poem_tibble <- poem_table[seq_len(n_links), ] %>%
  mutate(text = "")

# Iterate through links to grab text
for(i in seq_len(n_links)){

  # Identify URL
  link <- poem_tibble$_VARIABLE_CAPTURING_URL_[i]
```

```
  # Scrape poem title and text
  poem_tibble$text[i] <- tryCatch(

    # Return "Missing" instead of poem text when error is thrown
    error = function(cnd) {
      return("Missing")
    },

    # Scrape text otherwise
    _CODE_TO_SCRAPE_DATA_
  )
}
```

3 - Workflow

Yet again we've been working in an Rmd file to work through questions and answers. To practice the appropriate workflow, place your scraping code above in an R script called "scrape-poems-lab.R". Be sure to make the file reproducible, including loading any necessary packages at the top of the script. Use `write_csv()` to output a csv file called "dickinson-poems-partial.txt" (note the file extension!). If you have the code set up to run through ALL the poems, call it "dickinson-poems.txt".

I will demo the .R script file and provide the full data set later, as we need it.