Executing an API-based Scraping Project

How do I convert API requests into a dataset?

Building a Dataset using an API: Overview

- Three step process
  1. Get the necessary level of access to the API
  2. Create a template API request that grabs what you want
  3. Create a data structure/file containing all of the API requests you'll need to send, send them, and convert the results into a dataset

Data Requests Using an API

- **Step 1:** Get the necessary level of access to the API
  - Most APIs require "keys" or "tokens" or "secret phrase", etc.
    - To use these APIs, you will need to create an account with the service first, request an API key using your account, and then add the code it tells you to your GET query (e.g. http://server.com/api/query?token=abcd)
    - Examples: Facebook, Twitter, Glassdoor
  - Some APIs use implicit authentication, such as requiring you to access from a university IP address
    - Examples: Scopus, Web of Science
  - Some APIs allow open access without any authentication
    - Even so, sometimes you get increased data access with a token
    - Examples: Wikipedia, Google Books, the Star Wars API (https://swapi.co/)
- We'll be using **Google Books**
  - API documentation: https://developers.google.com/books/

Data Requests Using an API

- **Step 2:** Create a template API request that grabs what you want
  - Don't start in R. Start in Chrome.
  - Literally create an API request in the address bar of your browser.
  - Only move on once it looks like you're getting all of the variables you want out of it.
  - The output of an API can be in essentially *any* format, but some are more common.
    - If you're lucky
      - CSV: comma-separated values file
      - DAT: tab-delimited data file
    - More than likely
      - JSON: JavaScript object notation
  - Let's try one: https://www.googleapis.com/books/v1/volumes?q=i/o%20psychology

Typical Output from APIs

- JavaScript Object Notation (http://json.org)
  - Multi-level, hierarchically organized data, but not like you probably assume
  - Usually not human-friendly
  - Use Chrome extension: JSON Viewer (https://chrome.google.com/webstore/detail/json-viewer/gbmdgpbipfallnflgajpaliibnhdgobh?hl=en-US)

Data Requests Using an API

- **Step 3:** Create a data structure/file containing all of the API requests you'll need to send, send them, and convert the results into a dataset
  - You will usually need multiple API calls to get everything you want
  - Try to minimize the number of calls as much as possible
- From the Google Books API Documentation:
  - *You can paginate the volumes list by specifying two values in the parameters for the request:*
    - *startIndex - The position in the collection at which to start. The index of the first item is 0.*
    - *maxResults - The maximum number of results to return. The default is 10, and the maximum allowable value is 40.*

Let's Try It

- https://www.googleapis.com/books/v1/volumes?q=i/o%20psychology
  - Notice the URL encoding
  - Notice the 10 case return
  - Try to add startIndex and maxResults
- Let's say we want the title of every book considered to be "I/O Psychology" by Google
- What pattern will we eventually need?
  - Grab data 40 cases at a time, from 0 to the end
  - We know what case 0 looks like so, what does the end look like?
    - Let's try to figure out where the end is
  - So we will want to grab cases 40 at a time starting at 0, ending with 520
- You could do this by hand, or you could do in R/Python (let's try R)

38

A One-Slide Primer on R

- It's a statistical programming language
  - Basically everything in any programming language works with this format:
    returnValue = function(parameter1, parameter2)
  - *function* is a set of instructions that do something
    parameters are specific pieces of input to the function to change how it works
    returnValue is what information the method returns when it's done
  - Some functions have returnValues and some don't. Some functions just "do" things.
  - Everything must have a data type, such as *number* or *character* or *vector* or *list*.
- Example
  - numVec = c(1,2,3) # this creates a "vector" with 3 numbers
  - meanVec <- mean(numVec) # this calculates the mean of the vector values
  - print(meanVec) # this prints the value of meanVec where
    # you can see it

39

Remember to Iterate

- Check your data source theory and revise
  - Are these really all I/O psychology books?
  - Did you mean books written by I/O psychologists?
  - Did you mean books about I/O psychology topics?
  - What would have happened if the database changed while we were accessing it 40 cases at a time?
- When everything's ***final***, create streamlined "production" code
  This is impressive but is not production code:

```
library(jsonlite)
library(dplyr)
readBooks <- function(startIndex) {
  myOffset <- paste("https://www.googleapis.com/books/v1/volumes?q=i/o%20psychology&startIndex=",
  result <- fromJSON(myOffset)
  dyn.ctemp()
  titles <- result$items$volumeInfo$title
  df <- data.frame(myOffset,titles,stringsAsFactors=FALSE)
  print(startIndex)
  return(df)
}
books_df <- data.frame("offset"=character(), "titles"=character(), stringsAsFactors = FALSE)
indices <- seq(0,520,by=40)
for(startIndex in indices) books_df <- bind_rows(books_df, readBooks(startIndex))
write.csv(books_df, "../output/ko_books.csv")
```

40