Contents

1 week 7: apis continued

Goals:

- review MET API,
  - how to request, parse, and extract data
  - start to do a little data analysis of results
- examine concepts: authentication & pagination
  - NYT API
  - self guided exploration of NYPL API
- discuss article, start to brainstrom what we want to do

1.1 review: API requests, methods, calls (20m)

- discuss the anatomy of an API request
  - root, path, endpoint
  - making the call
- go over some useful methods for working with API data
  - dir(), json(), keys()
- using different paths: review documentation for the MET API:
  - search path
  - object path

```
base_url = "https://collectionapi.metmuseum.org"
path = "/public/collection/v1/search"
query = "?q=woman"

url = f'{base_url}{path}{query}'
r = requests.get(url)

type(r)
dir(r)

parsed = r.json()

parsed.keys()
```

1.2 looping through results to make many calls (20m)

- we are going to grab a bunch of these, create a dataset of works.
- let's start with the first fifty, saving them to a list:
  - get the IDs, make a list.
- write a loop that:
  - makes a call to the objects path
  - parses the json
  - saves to a relevant list

```
first_fifty = []
for item in ids:
    # passing the objectID variable into the URL
    base_url = 'https://collectionapi.metmuseum.org'
    path = '/public/collection/v1/objects/'
    url = f'{base_url}{path}{item}'
    # grabbing our response for that object
    response = requests.get(url)
    # parsing our response with json
    parsed = response.json()
    # appending the response to our new list
    first_fifty.append(parsed)
```

Dealing with errors: two ways: if statements and variables, with get()

```
for item in first_fifty:
  if item.get('artistDisplayName'):
    print(item['artistDisplayName'])

for item in first_fiftu:
  title = item.get('artistDisplayName')
  print(title)
```

Put it into a dataframe. Make a couple of charts - pie chart, bar chart, just for demonstration

```
# let's get a bunch of this data into lists

titles = []
names = []
genders = []
depts = []
countries = []
urls = []

for item in first_fifty:
    title = item.get('artistGender')
    titles.append(title)
    name = item.get('artistDisplayName')
    names.append(name)
    gender = item.get('artistGender')
    genders.append(gender)
    dept = item.get('department')
    depts.append(dept)
    country = item.get('country')
    countries.append(country)
    url = item.get('objectURL')
    urls.append(url)

import pandas as pd

df = pd.DataFrame({
  'title': titles,
  'name': names,
  'gender': genders,
  'department': depts,
  'country': countries,
  'link': urls
})

df.info()

df.value_counts('department')

df.department.value_counts().plot(kind = 'barh')

df.department.value_counts().plot(kind = 'pie')
```

1.3 authentication & pagination: NYTimes API (40m)

Creating API calls with keys

- requesting an API key
- adding the key

```
# key
key = 'HHhPAjnfEF2EmLtf9PdGsLEQZizR3iQu'

# query
query = 'migrant'

# base URL
url = f'https://api.nytimes.com/svc/search/v2/articlesearch.json?&q={query}&api-key={k

response = requests.get(url)

response # 200

type(response) # Response
```

Understanding our structure

- parse our data

- we find different structure of data
  - keep going deeper into the data using keys()

```
parsed = response.json()

parsed.keys() # dict_keys(['status', 'copyright', 'response'])

parsed['response'].keys() # dict_keys(['docs', 'meta'])

type(parsed['response']['meta']) # dict

parsed['response']['meta']

type(parsed['response']['docs']) # list

# The very first item in `docs` appears to be information about a
# single article.

parsed['response']['docs'][0]

parsed['response']['docs'][0].keys()

# If I wanted to proceed to get this data, I would write a loop and
# assign variables
```

Paginating through results

- We have 10 articles, because the NYTimes API only allows us to get 10 results at a time.
- We can paginate through the results by adding an &page= parameter, combined with a sleep() function, which allows us to insert pauses in our request (as to not overload the NYTimes servers).
- write a loop that goes through a range() of 5 pages, and does a search for each page.
- append the results for each page to a list called results
- check out our results, where is our data?

```
results = []
query = 'migrant'

for i in range(0, 5):
    url = f'https://api.nytimes.com/svc/search/v2/articlesearch.json?q={query}&page={i
    response = requests.get(url)
    parsed = response.json()
    articles = parsed['response']['docs']
    results.append(articles)
    sleep(6) # sleep at least 6 seconds not to overload the servers
```

In the end, we have a list of results, 5 lists to be exact (each list represents one page of articles). Each result (or page) contains information about 10 articles that matched our search.

```
results[0][0].keys() # for the very first item on the very first page.

results[0][0]['abstract']
```

BREAK

1.4 group challenge: the NYPL API

Walk them through the authentication process.

Challenge: use the search path, and figure out a way to get more information about the individual objects in the search path results.

1.5 discuss artivism article

What is "data artivism"?

- works that mobilize art and craft as tools of social contestation and (de)construction and as methods to engage with and visualize data
- not just about making things accessible that weren't before, not about representation through presence. But about changing *how* we see things.

Buidling on concept of "counterdata"

> The production of “counterdata” (D’Ignazio and Klein, 2020) on feminicide—a careful practice that involves researching, recording, and remembering the lives of women lost to violence (D’Ignazio et al., 2022)—has been a key practice for activists to visibilizar the issue. With their data, activists hope to move society from silence and indifference to outrage and action (Suárez Val, 2021). They also aim to contest the hegemonic visuality of feminicide: “an aesthetic order [. . .] made up of structures of visibility and invisibility that classify and separate individuals into grievable, political lives and culturally objectified lives” (Blaagaard, 2019: 253) that is dominated by often glamorized mainstream representations of violence against women, such as crime shows or news coverage.

The point of amassing this data is not only to make what is not often visualized more visible, but to also change the kind of visiblity that a topic usually gets: with gender-based violence, we have a lot of gratuitous representations in movies, the battered woman, for example, we also have crime shows.

There’s an example of a visualization project, "No estamos todas", that doesn’t just enumerate the number of deaths, but to remember them by their beauty. They "create an illustration for each woman to remember her in life, not as another crime statistic."

A possibility for your project, thinking about using your project for taking data and maybe presenting it in another way.

This kind of "data artivism" wants to represent data, differently to move people to action instead of just spectating.