

Webcraping Nonroad Activity Data with Selenium

Joseph Jakuta

Branch Chief, Air Quality Planning Branch, DOEE

September 2023



GOVERNMENT OF THE
DISTRICT OF COLUMBIA
MURIEL BOWSER, MAYOR

TAG THIS PRESENTATION @DOEE_DC

Nonroad Sectors Lacking Inventories



Commuter Trains

- Commuter train travel is in the EPA/LADCO inventory, but not idling at Ivy City Yard
- For 2020 estimated based on schedules, but 2023 we hope to account for more accurate activity
- Good news, MARC and VRE both have .yaml files and no need for Selenium



Helicopters

- EPA inventory only has emissions with one helipad in the District
- There is a reasonable level of helicopter traffic in the District, mostly hospitals and police
- Data can be seen online, but not straight html, yaml, xml, etc.
- We need to use Selenium

What is Selenium?

- “Selenium is an umbrella project for a range of tools and libraries that enable and support the automation of web browsers.”
- Goes beyond HTML parsers:
 - Explore Elements
 - Click Links, Buttons
 - Interact with Forms
 - Set Cookies
- Works with C#, Ruby, Java, **Python**, and Javascript
- Emulates **Chrome**, Firefox, Edge, Safari, and IE

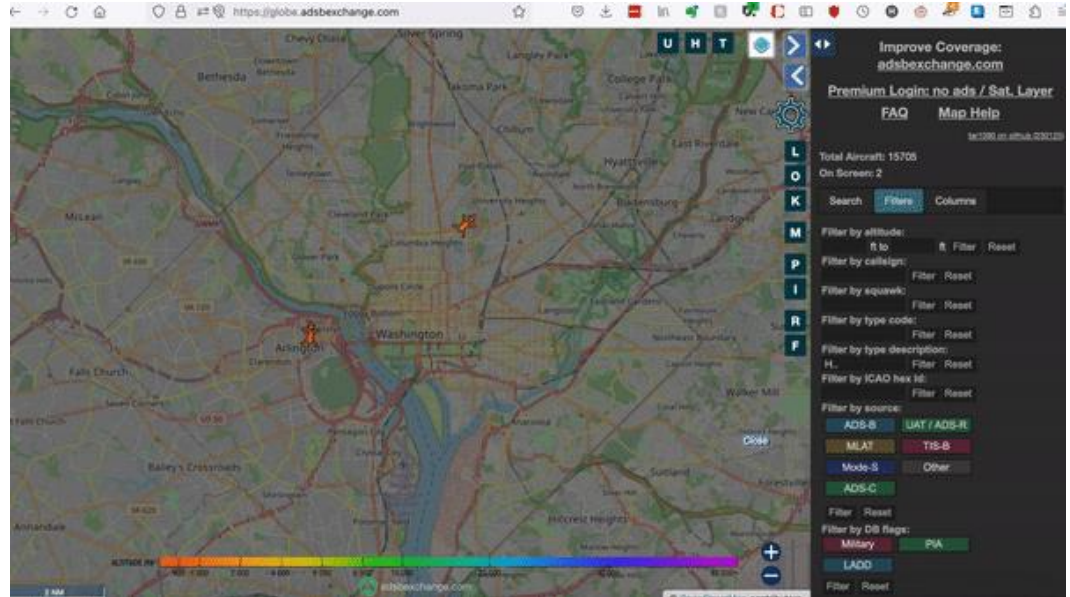
<https://www.selenium.dev/documentation/>

Website

Data from ADS-B Exchange:

<https://globe.adsbexchange.com>

“ADS-Bexchange.com offers a high fidelity, stable, and secure flight tracking service based on the world’s largest independent unfiltered ADS-B receiver network.”



Approach

- Libraries

- Selenium to access and manipulate the website
- BeautifulSoup to parse data
- SQLite to store data

- Notes

- Only saved helicopter data, though would work for airplanes
- Current focus activity data for 2023, emission calculations later
- Focus on “ground-level” emissions

Code Runs on
Server, Saves
to SQL DB,
Every Minute



Manual Script
to Dump DB
Contents to
.csv



Develop
Inventories
Based on
Activity:
Forthcoming

Code Snippets

1. Loading Libraries
2. Creating a Chrome Window
3. Executing Browser Actions
4. Saving Data

Note: This is not all of the code, only the highlights.

```
#Load Libraries
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from subprocess import CREATE_NO_WINDOW

from bs4 import BeautifulSoup
import sqlite3

#Create Chrome Window
chrome_options = Options()
chrome_options.add_argument("--headless")
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument("--window-size=1920,1030")
chrome_options.add_argument("--disable-dev-shm-usage")
chrome_options.binary_location = os.path.abspath(r"chrome-win\chrome.exe")

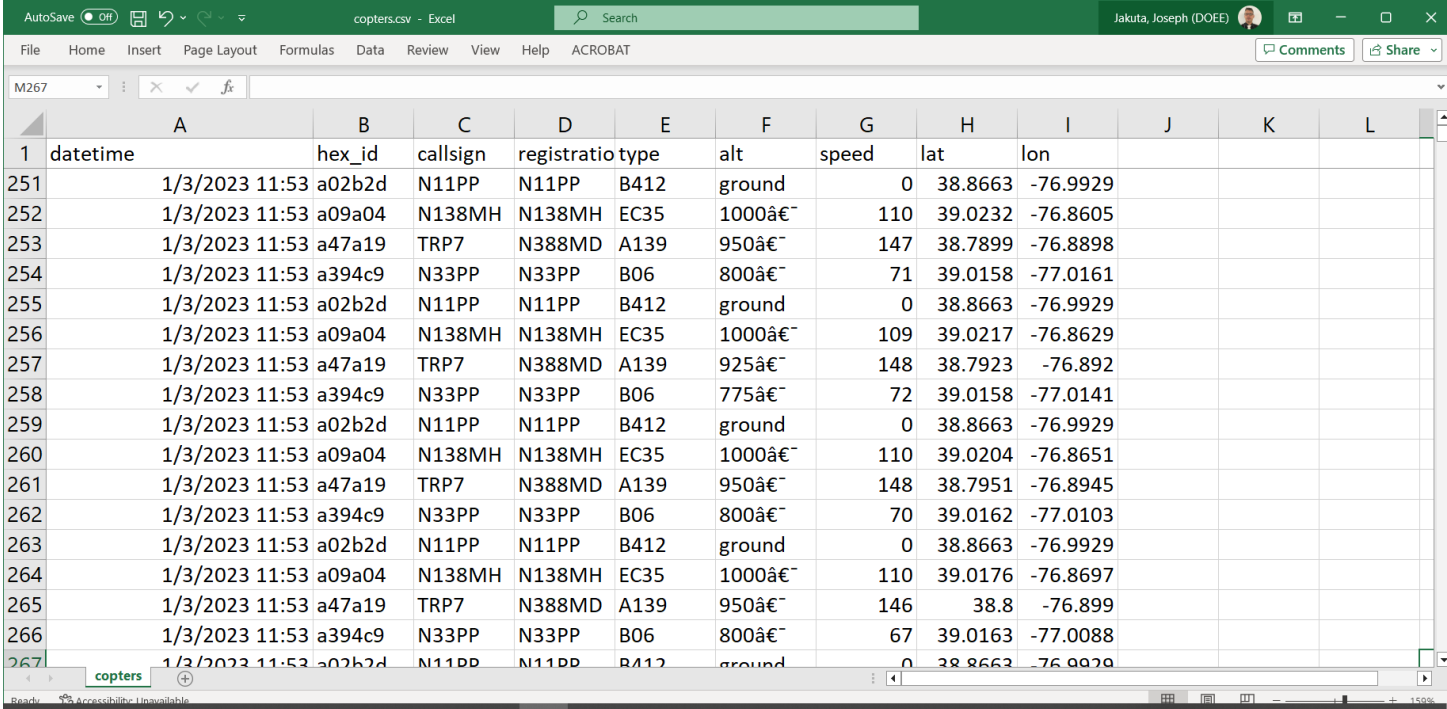
#Load the Chrome Service and URL
chrome_service = Service()
browserLoad = False
i = 0
while(not browserLoad and i < 10): #give the browser a few chances to load just in case of a problem
    try:
        browser = webdriver.Chrome(service=chrome_service, options=chrome_options)
        browserLoad = True
    except:
        browserLoad = False
        i = i+1
browser.get(url)

#perform some actions on the page to get the settings we want
browser.execute_script("arguments[0].click()",
                        browser.find_element(By.ID,"column_registration_cb"))
browser.execute_script("arguments[0].click()",
                        browser.find_element(By.ID,"column_flag_cb"))
browser.execute_script("arguments[0].click()",
                        browser.find_element(By.ID,"column_lat_cb"))
browser.execute_script("arguments[0].click()",
                        browser.find_element(By.ID,"column_lon_cb"))
browser.execute_script("arguments[0].click()",
                        browser.find_element(By.ID,"column_wd_cb"))
browser.execute_script("arguments[0].click()",
                        browser.find_element(By.ID,"column_ws_cb"))
browser.execute_script("arguments[0].value = 'H..',
                        browser.find_element(By.ID,"filters_description_input"))
browser.execute_script("arguments[0].click()",
                        browser.find_element(By.ID,"filters_description").find_element(By.XPATH, './button'))

#make sure everything has loaded before proceeding
time.sleep(10)
while(int(time.strftime("%M%S")) < 5955):
    #Parse the data
    soup = BeautifulSoup(browser.find_element(By.ID,"planesTable").get_property('outerHTML'), 'html.parser')
    for table in soup.find_all("table", id="planesTable"):
        for tr in table.find_all("tr"):
            tds = tr.find_all("td")
            if(tds[0].encode_contents().decode("utf-8") != 'Hex ID'):
                #Insert the data into SQL
                dbconn.execute("INSERT INTO copters (datetime, hex_id, callsign, registration, type, alt, speed,
                SELECT datetime('now', 'localtime'), ?, ?, ?, ?, ?, ?, ?)
                (tds[0].encode_contents().decode("utf-8"), tds[1].encode_contents().decode("utf-8"), tds[2].enco

            dbconn.commit()
        time.sleep(2)
    dbconn.close()
```

Data Set (1-Minute Snapshot)



	A	B	C	D	E	F	G	H	I	J	K	L
1	datetime	hex_id	callsign	registratio	type	alt	speed	lat	lon			
251	1/3/2023 11:53	a02b2d	N11PP	N11PP	B412	ground	0	38.8663	-76.9929			
252	1/3/2023 11:53	a09a04	N138MH	N138MH	EC35	1000â€	110	39.0232	-76.8605			
253	1/3/2023 11:53	a47a19	TRP7	N388MD	A139	950â€	147	38.7899	-76.8898			
254	1/3/2023 11:53	a394c9	N33PP	N33PP	B06	800â€	71	39.0158	-77.0161			
255	1/3/2023 11:53	a02b2d	N11PP	N11PP	B412	ground	0	38.8663	-76.9929			
256	1/3/2023 11:53	a09a04	N138MH	N138MH	EC35	1000â€	109	39.0217	-76.8629			
257	1/3/2023 11:53	a47a19	TRP7	N388MD	A139	925â€	148	38.7923	-76.892			
258	1/3/2023 11:53	a394c9	N33PP	N33PP	B06	775â€	72	39.0158	-77.0141			
259	1/3/2023 11:53	a02b2d	N11PP	N11PP	B412	ground	0	38.8663	-76.9929			
260	1/3/2023 11:53	a09a04	N138MH	N138MH	EC35	1000â€	110	39.0204	-76.8651			
261	1/3/2023 11:53	a47a19	TRP7	N388MD	A139	950â€	148	38.7951	-76.8945			
262	1/3/2023 11:53	a394c9	N33PP	N33PP	B06	800â€	70	39.0162	-77.0103			
263	1/3/2023 11:53	a02b2d	N11PP	N11PP	B412	ground	0	38.8663	-76.9929			
264	1/3/2023 11:53	a09a04	N138MH	N138MH	EC35	1000â€	110	39.0176	-76.8697			
265	1/3/2023 11:53	a47a19	TRP7	N388MD	A139	950â€	146	38.8	-76.899			
266	1/3/2023 11:53	a394c9	N33PP	N33PP	B06	800â€	67	39.0163	-77.0088			
267	1/3/2023 11:53	a02b2d	N11PP	N11PP	B412	ground	0	38.8663	-76.9929			

Other Thoughts

Challenges

- Need to Use Windows Server
Solution: Convert python script to Windows .exe
- Data Size: ~1.2GB per Year
Solution: Will create fresh SQL DB annually

Future Work

- Calculate helicopter emissions
- Collect airplane data
 - Not useful for NEI, DC has no airports, but could be useful for health impacts
 - Need to reevaluate data approach
- Explore Selenium technique for other sectors to build bottom-up inventories

Questions (Later...)

Email: joseph.jakuta@dc.gov

