09/01/2024

Correction du CSV obtenu depuis l'API hubeau.eaufrance

Projet PLANT

Auteur
CAPGEMINI ENGINEERING

1. Inversion Longitude/Latitude

Lors de l'utilisation de l'API sur le site hubeau.eaufrance : https://hubeau.eaufrance.fr/, nous nous sommes rendu compte que certaines stations avaient une inversion entre les longitudes et les latitudes. Nous avons, dans le tableau ci-dessous, toutes les stations modifiées:

Tableau 1 : Stations dont la longitude/latitude ont été inversée

	Libelle site	**Code site**	**Code station**
1.	Le Petit Gland à Any-Martin-Rieux	H0010003	H001000301
2.	L'Oise à Étréaupont	H0020011	H002001101
3.	Le Ton à Étréaupont	H0030002	H003000201
4.	L'Oise à Guise	H0040001	H004000101
5.	L'Oise à Proisy	H0040002	H004000201
6.	La Serre à Chaourse	H0100022	H010002201
7.	La Serre à Montigny-sous-Marle	H0100023	H010002301
8.	La Brune à Hary	H0120002	H012000201
9.	Le Vilpion à Marle	H0130001	H013000101
10.	Le ruisseau de Saint-Lambert à Suzanne	H1240003	H124000301
11.	La Foivre à Écordal	H1250002	H125000202
12.	Le ruisseau de Saulces à Auboncourt-Vauzelles	H1270001	H127000101
13.	Le ruisseau de Fayau à Aizelles	H1430002	H143000201
14.	Le ruisseau d'Arcombe à Maurs	O8264011	O826401101
15.	Le ruisseau de Planioles à Planioles	O8310002	O831000201
16.	Le ruisseau de Planioles à Figeac	O8310003	O831000301
17.	L'Aude à Cuxac-d'Aude [Déversoirs 1952]	Y1612056	Y161205602
18.	Vallat le Grand [Affluent Rive Gauche Arc] à Aix-en-Provence	Y4022014	Y402201401

2. Correction des altitudes :

Les altitudes fournies par l'API sont soit inconnues, soit erronées. Nous avons comparé ces résultats avec les altitudes disponibles sur le site : https://www.freemaptools.com/elevation-finder.htm . Nous avons constaté que certaines valeurs de l'API étaient proches des résultats du site pour quelques stations, tandis que pour d'autres, les différences étaient significatives. Cependant, une observation intéressante est que si nous effectuons la conversion en millimètres, les résultats se rapprochent davantage. Cette constatation nous amène à conclure que l'API fournit des valeurs d'altitude en mètres pour certaines stations et en millimètres pour d'autres.

Exemples :

Tableau 2 : Comparaison entre les altitudes de quelques stations obtenues par l'API hubeau et par elevation_finder

Code_station	Logitude / Latitude	Altitude API	Altitude elevation finder	Commentaires
1015000101	16.175997699, -61.69319975	218.0	233.3 m	Résultats proches (API en m)
4000000101	-21.376618691, 55.615699903	31.2	45.0 m	
1120000201	16.057036348, -61.56608824	11830.0	19.4 m	API en mm ?
2501000101	14.716331664, -61.048843648	240550.0	255.1 m	

De plus, 47 % des stations ne disposent pas d'informations d'altitude, rendant l'utilisation de cette colonne très délicate dans nos études. Par conséquent, nous avons remplacé toutes les altitudes par les résultats obtenus via l'API OpenTopoData : https://www.opentopodata.org/. Nous n'avons pas opté pour le simple remplissage des données inconnues en raison des problèmes liés aux unités de mesure, comme expliqué ci-dessus, ainsi que la différence d'environ 15 m entre les résultats de Hubeau.eaufrance et OpenTopoData, ce qui pourrait être significatif pour les stations situées à des niveaux proches.

Le choix de l'API OpenTopoData a été effectué après plusieurs tests. Nous avons notamment essayé OpenCageGeocode (https://opencagedata.com/dashboard?chartdays=14#geocoding), qui fonctionne mais fournit des résultats différents d'Elevation Finder. De même, des divergences ont été constatées avec l'API https://serpapi.com/dashboard et quelques autres.

Veuillez trouver en pièce jointe le fichier CSV contenant les données après les modifications mentionnées précédemment.

Ci-dessous, vous trouverez le code utilisé pour le remplissage des altitudes :

```
import csv
import requests
import time

def get_elevation_open_topo_data_with_retry(latitude, longitude, dataset="aster30m", max_retries=3):
    url = f"https://api.opentopodata.org/v1/{dataset}?locations={latitude},{longitude}"
    for _ in range(max_retries):
        try:
            response = requests.get(url)
            response.raise_for_status()
            data = response.json()
            elevation = data['results'][0]['elevation']
            return elevation
        except requests.exceptions.HTTPError as e:
            if response.status_code == 429:
                print(f"Rate limit exceeded. Retrying after 2 seconds...")
                time.sleep(2)
                continue
            else:
                print(f"HTTP error: {e}")
                break
        except Exception as e:
            print(f"Error getting elevation: {e}")
            break
    return None

def remplacer_toutes_altitudes(fichier_entree, fichier_sortie, fichier_erreur):
    k = 0
    erreurs = []
    with open(fichier_entree, 'r', newline='', encoding='utf-8') as fichier_entree:
        lecteur_csv = csv.DictReader(fichier_entree)
        lignes = list(lecteur_csv)

    for ligne in lignes:
        latitude_str = ligne['latitude_station']
        longitude_str = ligne['longitude_station']

        if latitude_str and longitude_str:
            latitude = float(latitude_str)
            longitude = float(longitude_str)

            if latitude is not None and longitude is not None:
                altitude = get_elevation_open_topo_data_with_retry(latitude, longitude)

                if altitude is not None:
                    ligne['altitude_ref_alti_station'] = str(altitude)
                    k += 1
                    print(k)
                else:
                    erreurs.append({'latitude': latitude_str, 'longitude': longitude_str})
            else:
                print("alt not modif")

    with open(fichier_sortie, 'w', newline='', encoding='utf-8') as fichier_sortie:
        entetes = lecteur_csv.fieldnames
        ecrivain_csv = csv.DictWriter(fichier_sortie, fieldnames=entetes)
        ecrivain_csv.writeheader()
        ecrivain_csv.writerows(lignes)

    if erreurs:
        with open(fichier_erreur, 'w', newline='', encoding='utf-8') as fichier_erreur:
            entetes_erreur = ['latitude', 'longitude']
            ecrivain_erreur = csv.DictWriter(fichier_erreur, fieldnames=entetes_erreur)
            ecrivain_erreur.writeheader()
            ecrivain_erreur.writerows(erreurs)
```

```
    print(f"Le fichier '{fichier_sortie}' a été modifié avec succès.")

# Remplacez le chemin du fichier d'entrée, de sortie et d'erreur en conséquence
fichier_entree = 'chemin du fichier d'entrée'
fichier_sortie = 'chemin du fichier de sortie'
fichier_erreur = 'chemin du fichier d'erreur'

remplacer_toutes_altitudes(fichier_entree, fichier_sortie, fichier_erreur)
```