Tutorial EcoCounter-Daten

In diesem Tutorial erfahren Sie, wie Sie Daten der EcoCounter-Fahrradzählstellen aus Baden-Württemberg abrufen und analysieren können. Diese Daten umfassen tägliche Fahrradzählungen, die für verschiedene Analyse- und Visualisierungszwecke verwendet werden können.

1. Datenquelle

Die EcoCounter-Daten für Baden-Württemberg sind über die folgende URL zugänglich:

https://mobidata-bw.de/daten/eco-counter/v2/fahrradzaehler_tageswerten_yyyy.json.gz

Hierbei steht yyyy für das Jahr, für das die Daten abgerufen werden sollen. Zum Beispiel erhalten Sie die Daten für das Jahr 2020 durch den folgenden Link:

https://mobidata-bw.de/daten/eco-counter/v2/fahrradzaehler_tageswerten_2020.json.gz

2. Datei entpacken

Die heruntergeladene Datei ist im GZip-Format (.gz) komprimiert. Um diese Datei zu entpacken, können Sie ein Tool wie 7-Zip verwenden.

3. Importieren in JupyterLab

Um die entpackte Datei in JupyterLab zu importieren, können Sie die Drag-and-Drop-Funktion verwenden. Öffnen Sie dazu JupyterLab in Ihrem Webbrowser und ziehen Sie die Datei per Drag & Drop in den JupyterLab-Datei-Browser (linker Rand).

Alternativ können Sie die Datei auch mit Python importieren, indem Sie die folgenden Schritte ausführen:

```
import pandas as pd

data = pd.read_csv('fahrradzaehler_tageswerten_2020.csv') #Dateiname ggf. um Pfad ergänzen
```

4. Position der Stationen als CSV ausgeben

Um die Positionen der Stationen in einem CSV-Format auszugeben, können Sie die Informationen aus der Datenquelle extrahieren und in eine CSV-Datei speichern. Hier ist ein Beispiel, wie das aussehen könnte:

```
import pandas as pd
```

```
# Name der JSON-Datei und der gewünschten CSV-Datei
input_file = 'fahrradzaehler_tageswerten_2020.json'
output_file = 'bike_count_locations.csv'

# Einlesen der JSON-Datei in ein DataFrame
df = pd.read_json(input_file)

# Auswahl der relevanten Spalten für QGIS
df_locations = df[['domain_name', 'domain_id', 'counter_site',
'counter_site_id', 'latitude', 'longitude']]

# Entfernen von doppelten Einträgen
df_locations = df_locations.drop_duplicates()

# Speichern der Daten in eine CSV-Datei
df_locations.to_csv(output_file, index=False, encoding='utf-8')

print(f"CSV-Datei '{output_file}' wurde erfolgreich erstellt.")
```

5. Abfrage von Werten für eine Station an einem Tag

Um die Werte für eine bestimmte Station an einem bestimmten Tag abzufragen, können Sie einen Filter auf das DataFrame anwenden. Hier ist ein Beispiel:

```
import pandas as pd

# Name der JSON-Datei und ID des gewünschten Standorts
input_file = 'fahrradzaehler_tageswerten_2020.json'
counter_site_id = 100003358  # Beispiel für die gewünschte
'counter_site_id'
date_to_filter = "2020-01-01"  # Das gewünschte Datum im Format
"YYYY-MM-DD"

# Einlesen der JSON-Datei in ein DataFrame
df = pd.read_json(input_file)
```

```
# Auswahl des gewünschten Standorts anhand der 'counter_site_id'
df_filtered = df[df['counter_site_id'] == counter_site_id]


# Normalisieren der 'channels'-Daten
channels_expanded =
pd.json_normalize(df_filtered['channels'].explode())


# Filter auf das gewünschte Datum anwenden
channels_expanded =
channels_expanded[channels_expanded['iso_timestamp'].str.startswith(
date_to_filter)]


# Ausgabe der gefilterten Daten
print(f"Zählwerte für Standort {counter_site_id} am
{date_to_filter}:")
print(channels_expanded[['iso_timestamp', 'direction', 'counts']])
```

Autor: Markus Jackenkroll