example_script

September 14, 2022

1 Create electricity and heat demand profiles for buildings in Germany

This jupyter notebook can be used to create and plot electricity and heat demand curves for a specitic building in Germany using the supplementary dataset for the paper "Open modeling of electricity and heat demand curves for all residential buildings in Germany".

The data needs to be provided by a local pgSQL database which incldes the published backup file from zenodo.

After restoring the backup file, the data is stored in different schemas: society, openstreetmap and demand. Different tables have to be combined to create the final demand time series for heat and electricity. In the following, the querys to select demand curves are presented.

[ ]:
```
import pandas as pd
import numpy as np
import geopandas as gpd
from datetime import datetime
import os
from sqlalchemy import create_engine
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go
from matplotlib import pyplot as plt
import matplotlib
import folium

def engine(db_config):
    """Engine for local database."""

    return create_engine(
        f"postgresql+psycopg2://{db_config['POSTGRES_USER']}:"
        f"{db_config['POSTGRES_PASSWORD']}@{db_config['HOST']}:"
        f"{db_config['PORT']}/{db_config['POSTGRES_DB']}",
        echo=False,
    )
```

1.1 Set connection parameters to restored database

[ ]:
```
db_config = {
    "POSTGRES_USER": "postgres",
    "POSTGRES_PASSWORD": "postgres",
    "HOST": "localhost",
    "PORT": "5432",
    "POSTGRES_DB": "egon-data",
}
```

1.2 Choose index of building

[ ]:
```
building_id = 842872
```

[ ]:
```
gdf = gpd.read_postgis(
    f"""
        SELECT * FROM openstreetmap.osm_buildings_residential
        WHERE id = {building_id}
    """,
    geom_col = 'geom_building',
    con = engine(db_config),
    )
```

[ ]:
```
m = folium.Map(
    location=[gdf.geom_building.centroid.to_crs(4326).y, gdf.geom_building.
 ↪centroid.to_crs(4326).x],
    tiles="cartodbpositron",
    zoom_start=20,
    control_scale=True,
    prefer_canvas=True
)

folium.features.GeoJson(
    gdf,
    name="Selected building",
    style_function=lambda x: {"fillColor": "red", "color": "000", "weight": 0.5}
).add_to(m)

m
```

1.3 Select electricity demand profile for building

[ ]:
```
df_elec = pd.read_sql(
    f"""
        SELECT sum(elem * dem.factor_2035)/1000 AS demand_profile
            FROM
```

```
                demand.egon_household_electricity_profile_of_buildings as␣
↪prof_buildings,
                demand.iee_household_load_profiles as prof,
                unnest(prof.load_in_wh) WITH ORDINALITY x(elem, rn),
                demand.egon_household_electricity_profile_in_census_cell dem
            WHERE
                prof_buildings.profile_id = prof.type AND
                prof_buildings.building_id = {building_id} AND
                prof_buildings.cell_id = dem.cell_id
            GROUP BY rn
            ORDER BY rn;
    """,
        engine(db_config),
    )

df_elec.index = pd.date_range(
        datetime(2011, 1, 1, 0), periods=8760, freq="H"
    )
```

1.4 Plot electricity demand over the year in hourly resolution

[ ]:
```
fig = px.line(df_elec, labels={'variable': 'demand', 'value':␣
↪'electricity_demand'})
fig.update_yaxes(title_text="Electricity demand in kWh", row=1, col=1)
fig.update_xaxes(title_text="time")
fig.show()
```

1.5 Select heat demand profile for building

[ ]:
```
df_heat = pd.read_sql(
    f"""
    SELECT (b.demand/f.count * UNNEST(e.idp) * d.daily_demand_share)*1000 AS␣
↪demand_profile
    FROM
    (SELECT * FROM demand.egon_heat_timeseries_selected_profiles
    ,
    UNNEST(selected_idp_profiles) WITH ORDINALITY as selected_idp) a

    JOIN demand.egon_peta_heat b
    ON b.zensus_population_id = a.zensus_population_id
    JOIN boundaries.egon_map_zensus_climate_zones c
    ON c.zensus_population_id = a.zensus_population_id
    JOIN demand.egon_daily_heat_demand_per_climate_zone d
    ON (c.climate_zone = d.climate_zone AND d.day_of_year = ordinality)
    JOIN demand.egon_heat_idp_pool e
    ON selected_idp = e.index
```

```
    JOIN (SELECT zensus_population_id, COUNT(building_id)
           FROM demand.egon_heat_timeseries_selected_profiles
          GROUP BY zensus_population_id
    ) f
    ON f.zensus_population_id = a.zensus_population_id



    WHERE a.building_id = {building_id}
    AND b.scenario = 'eGon2035'
    AND b.sector = 'residential'
    ;
    """,
        engine(db_config),
    )

df_heat.index = pd.date_range(
        datetime(2011, 1, 1, 0), periods=8760, freq="H"
    )
```

1.6 Plot heat demand over the year in hourly resolution

[ ]:
```
fig = px.line(df_heat,labels={'variable': 'demand', 'value': 'heat_demand'})
fig.update_yaxes(title_text="Heat demand in kWh", row=1, col=1)
fig.update_xaxes(title_text="time")
fig.show()
```

1.7 Plot electricity and heat demand over the year in hourly resolution

[ ]:
```
df = pd.DataFrame(data={'electricity': df_elec.demand_profile, 'heat': df_heat.
 ↪demand_profile})

fig = px.line(df,labels={'variable': 'demand'})
fig.update_yaxes(title_text="Demand in kWh", row=1, col=1)
fig.update_xaxes(title_text="time")
fig.show()
```

[ ]:
```

```