Getting Started - PyPSA Eur

Useful links

PyPSA-Eur:

- PyPSA-Eur's website
  - PyPSA-Eur's github repository
- PyPSA-Eur introductory paper
  Programming environment:
- Tutorials: Conda + data science with Python

---

1. Introduction

> "PyPSA-Eur is an open model dataset of the European energy system at the transmission network level that covers the full ENTSO-E area.
> It covers demand and supply for all energy sectors."
> (PyPSA-Eur's website)

It is a workflow for modeling energy networks; first **building the database**, then solving **operation and expansion optimizations**.

One can choose which energy sector(s) to consider (electricity, transport, heating, biomass, etc.). Considering electricity:

> "The electricity system representation contains alternating current lines at and above 220 kV voltage level and all high voltage direct current lines, substations, an open database of conventional power plants, time series for electrical demand and variable renewable generator availability, geographic potentials for the expansion of wind and solar power.
> The model is suitable both for **operational studies** and generation and transmission **expansion planning** studies."

In particular, one can leverage PyPSA-Eur database-building capacities for creating **digital twins** of real electrical networks.

PyPSA-Eur relies on two **Python** based tools:
- Snakemake, "a tool to create **reproducible and scalable** data analyses." Through a unique configuration file, one can specify **all the parameters** to consider for building the database along with optimization parameters.

- PyPSA, an energy system modeling Python-library.

2. Installation

For using PyPSA-Eur, one should use *conda*.

A. Installing Conda

Conda is a "package, dependency, and environment management" tool. It can be installed through:

- Anaconda distribution (all in one)
- Miniconda (customizable, lighter than anaconda)
- Miniforge (no license restrictions)
  If getting started, one should use Anaconda or Miniconda.

B. Setting up PyPSA-Eur environment

1. Open conda in the terminal (*terminal* -> select *conda* in the upper right selection panel).
2. Using *cd* prompt, move to the desired folder for saving the files.
`cd C:/Users/…`
3. Clone PyPSA-Eur's github repository.

`git clone https://github.com/PyPSA/pypsa-eur.git`
4. Create a *conda* environment dedicated to PyPSA-Eur (or give the absolute path of the `environment.yaml` file downloaded in the from the github repository).
`conda env create -f pypsa-eur/envs/environment.yaml`

3. Running Snakemake

One should configure the database and solver parameters before running the snakemake process. In the *config* folder (of the github repository), either:
- Create a blank *.yaml* file named *config.yaml* (create a *.txt* file and change the extension to *.yaml).*
- Duplicate the *default.congi.yaml* file, renaming it to *config.yaml*.

Then custom the *config.yaml* file to your need (see the role of this file described hereafter). Alternatively, here is a proposed *config.yaml* file with pre-selected major parameters to play with.

The *default.config.yaml* file contains all the parameters, along with their default values. Later, when running the snakemake process, it will look for a *config.yaml* file for overwriting default parameters (for this run only). Thus, the *config.yaml* file can be tailored to one's needs, with **any subset of parameters from the default file**.

One can find all the information on configuration parameters in PyPSA-Eur's documentation. You can also refer to the proposed *config.yaml* file for a non-exhaustive list of major parameters, with comments.

For running the snakemake process, which creates / updates the databases and performs optimization in one go, one should:
1. Open conda in the terminal (*terminal* -> select *conda* in the upper right selection panel).
2. Activate PyPSA-Eur's environment.
`conda activate pypsa-eur`
3. Move to the pypsa-eur directory using the `cd C:/Users/…` command.
4. Call the snakemake process using either `snakemake -call solve_elec_networks` for **electrical networks only** or `snakemake -call all` for a multi-sector optimization.

One can first add `-n` in the end of the command line for testing the command with a **dry-run** (it gives a list of all the *rules*, i.e. sub functions, it will perform). One should add `--cores all` for allowing the process to fully use the computer's resources.

A "list of jobs" is updated in the terminal along the process. When finished, one can find a *.nc* file (multiple ones in case of multi-scenarios configuration) in the *results/{series_name}/{run_name}* folder. This file is a PyPSA one that stores the final optimized network.

Bonus - MATPOWER like converter program

A python program enabling to extract a networks' electrical data built by PyPSA-Eur is proposed.

It extracts data from the resulting *.nc* PyPSA network and converts it into a MATPOWER like (i.e. similar to the PTI format) data structure, in the form of *.csv* files. The resulting data structure is well described in the file's introduction.