PyPSA-Eur Documentation of Energie Studie 2050+

Release 0.10.0

Tom Brown (KIT, TUB, FIAS), Jonas Hoersch (KIT, FIAS), Fabian Hofmann (TUB, FIAS), Fabian Neumann (TUB, KIT), Marta Victoria (Aarhus University), Lisa Zeyen (KIT, TUB), Thomas Gilon (Climact), Vincent Laguna (Climact), Benoit Martin (Climact), Dimitri Krings (Climact)

Sep 19, 2024

CONTENTS

CHAPTER

ONE

PYPSA-EUR: A SECTOR-COUPLED OPEN OPTIMISATION MODEL OF THE EUROPEAN ENERGY SYSTEM

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. From version v0.8.0, PyPSA-Eur includes all the features from PyPSA-Eur-Sec, which is now deprecated.

1.1 Energie Studie 2050+ for VEKA and VLAIO

This version of PyPSA-Eur is an extended version designed by Climact for the *Flemish Energy and Climate Agency* (VEKA, Belgium) and the *Flanders Innovation & Entrepreneurship* (VLAIO, Belgium) in the context of the *Energie Studie 2050+* (see *Energie Studie 2050+ for VEKA and VLAIO* for further details).

See also:

Detailed chapter about modifications made by Climact : *Energie Studie 2050+ for VEKA and VLAIO*

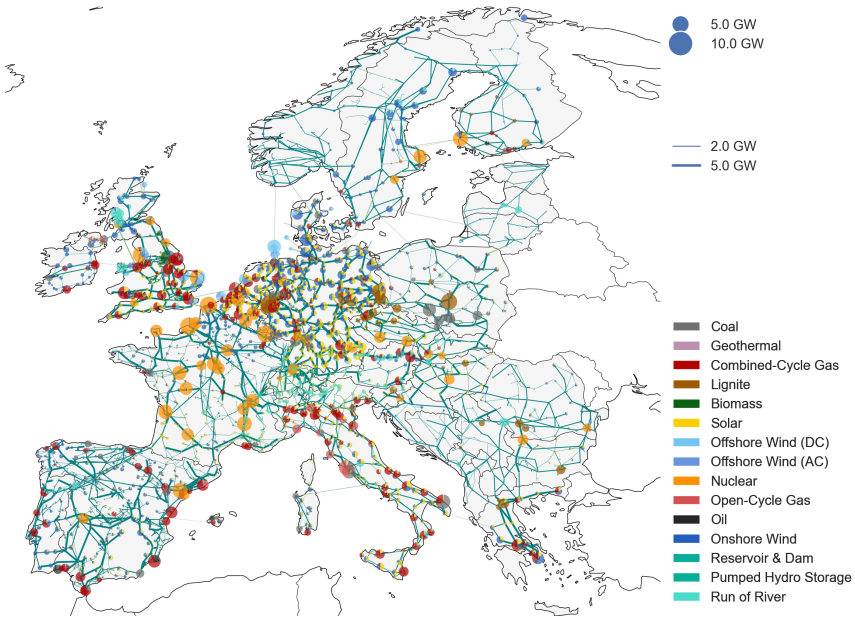
See also:

Web interface to explore the data online

1.2 Electricity System

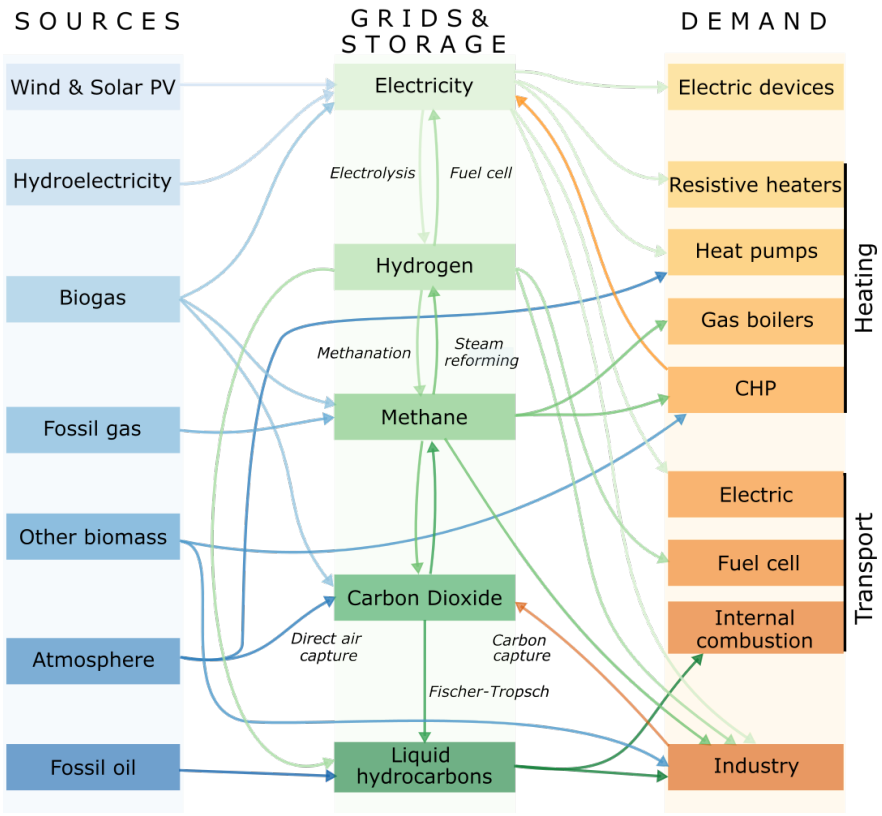
The electricity system representation contains alternating current lines at and above 220 kV voltage level and all high voltage direct current lines, substations, an open database of conventional power plants, time series for electrical demand and variable renewable generator availability, geographic potentials for the expansion of wind and solar power.

The model is suitable both for operational studies and generation and transmission expansion planning studies. The continental scope and highly resolved spatial scale enables a proper description of the long-range smoothing effects for renewable power generation and their varying resource availability.



1.3 Sector-Coupled Energy System

A sector-coupled extension (previously known as **PyPSA-Eur-Sec**, which is now deprecated) adds demand and supply for the following sectors: transport, space and water heating, biomass, energy consumption in the agriculture, industry and industrial feedstocks, carbon management, carbon capture and usage/sequestration. This completes the energy system and includes all greenhouse gas emitters except waste management, agriculture, forestry and land use. The diagram below gives an overview of the sectors and the links between them:



Note: You can find showcases of the model's capabilities in the Supplementary Materials of the Joule paper The potential role of a hydrogen network in Europe, the Supplementary Materials of another paper in Joule with a description of the industry sector, or in a 2021 presentation at EMP-E. The sector-coupled extension of PyPSA-Eur was initially described in the paper Synergies of sector coupling and transmission reinforcement in a cost-optimised, highly renewable European energy system (2018) but it differs by being based on the higher resolution electricity transmission model PyPSA-Eur rather than a one-node-per-country model, and by including biomass, industry, industrial feedstocks, aviation, shipping, better carbon management, carbon capture and usage/sequestration, and gas networks.

1.4 About

PyPSA-Eur is designed to be imported into the open energy system modelling framework PyPSA for which documentation is available as well. However, since the workflow is modular, it should be easy to adapt the data workflow to other modelling frameworks.

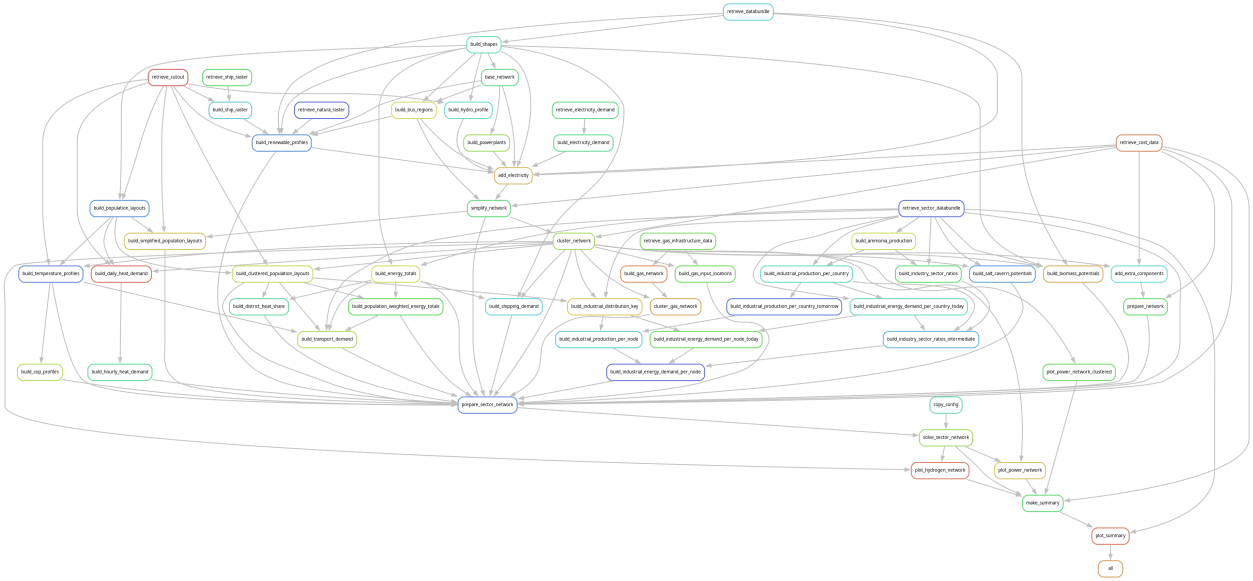
The restriction to freely available and open data encourages the open exchange of model data developments and eases the comparison of model results. It provides a full, automated software pipeline to assemble the load-flow-ready model from the original datasets, which enables easy replacement and improvement of the individual parts.

Warning: PyPSA-Eur is under active development and has several *Limitations* which you should understand before using the model. The Github repository issues collect known topics we are working on. Please feel free to help or make suggestions.

This project is currently maintained by the Department of Digital Transformation in Energy Systems at the Technische

Universität Berlin. Previous versions were developed within the IAI at the Karlsruhe Institute of Technology (KIT) which was funded by the Helmholtz Association, and by the Renewable Energy Group at FIAS to carry out simulations for the CoNDyNet project, financed by the German Federal Ministry for Education and Research (BMBF) as part of the Stromnetze Research Initiative.

1.5 Workflow



Note: The graph above was generated using `snakemake --rulegraph -F | sed -n "/digraph/,/}/p" | dot -Tpng -o workflow.png`

1.6 Learning Energy System Modelling

If you are (relatively) new to energy system modelling and optimisation and plan to use PyPSA-Eur, the following resources are one way to get started in addition to reading this documentation.

- Documentation of PyPSA, the package for modelling energy systems which PyPSA-Eur uses under the hood.
- Course on Energy Systems given at Technical University of Berlin by Prof. Dr. Tom Brown.
- Course on Data Science for Energy System Modelling given at Technical University of Berlin by Dr. Fabian Neumann.

1.7 Citing PyPSA-Eur

If you use PyPSA-Eur for your research, we would appreciate it if you would cite one of the following papers:

For electricity-only studies:

```
@article{PyPSAEur,
    author = "Jonas Hoersch and Fabian Hofmann and David Schlachtberger and Tom Brown",
    title = "PyPSA-Eur: An open optimisation model of the European transmission system",
    journal = "Energy Strategy Reviews",
    volume = "22",
    pages = "207--215",
    year = "2018",
    doi = "10.1016/j.esr.2018.08.012",
    eprint = "1806.01613"
}
```

For sector-coupling studies:

```
@misc{PyPSAEurSec,
    author = "Fabian Neumann and Elisabeth Zeyen and Marta Victoria and Tom Brown",
    title = "The potential role of a hydrogen network in Europe",
    journal "Joule",
    volume = "7",
    pages = "1--25"
    year = "2023",
    eprint = "2207.05816",
    doi = "10.1016/j.joule.2023.06.016",
}
```

For sector-coupling studies with pathway optimisation:

```
@article{SpeedTechnological2022,
    title = "Speed of technological transformations required in {Europe} to achieve␣
↪different climate goals",
    author = "Marta Victoria and Elisabeth Zeyen and Tom Brown",
    journal = "Joule",
    volume = "6",
    number = "5",
    pages = "1066--1086",
    year = "2022",
    doi = "10.1016/j.joule.2022.04.016",
    eprint = "2109.09563",
}
```

If you want to cite a specific PyPSA-Eur version, each release of PyPSA-Eur is stored on Zenodo with a release-specific DOI:

1.8 Operating Systems

The PyPSA-Eur workflow is continuously tested for Linux, macOS and Windows (WSL only).

CHAPTER

TWO

ENERGIE STUDIE 2050+ FOR VEKA AND VLAIO

2.1 Overview

This chapter aims to give a broad overview of the use of PyPSA-Eur in the context of the *Energie Studie 2050+* for the *Flemish Energy and Climate Agency* (VEKA, Belgium).

See also:

Web interface to explore the data online

2.1.1 High level description

PyPSA-Eur is an open model dataset of the European energy system at the transmission network level that covers the full ENTSO-E area. Written in Python, it optimizes the energy system with high temporal and spatial resolution for a large spectrum of technologies and energy vectors. It covers demand and supply for all energy sectors.

General description

PyPSA-Eur optimizes energy systems with

- a temporal granularity up to 1 hour (which represents a maximum of 8760 time steps each year) ;
- a spatial granularity up to ENTSO-E transmission lines (which represents a maximum of 8800 electrical buses) ;
- a technology list reaching up to 80 different technologies (see *Technological assumptions*).

PyPSA-Eur allows connecting different energy vectors (referred to as energy carriers in PyPSA-Eur) and technologies using them, hence allowing to tackle inter-sectorial and inter-carriers connections.

The system is optimized over the energy system cost (including OPEX and annualized CAPEX), for different consecutive planning horizons. The optimization is done over the sizing and dispatch of energy production and storage units and of transmission infrastructures to meet an inelastic demand for each of the time frames and geographical locations considered.

The cost optimization of the energy system for a given year is performed under different constraints, including an annual CO2 emission constraint for the given year and renewable potentials.

Optimization details

For each planning horizon, PyPSA-Eur provides the cost optimal energy dispatch of each asset at each node and for each time frame given an hourly demand at each node of each energy carrier, as well as the cost optimal sizing of production, storage and transmission units needed for such dispatch (see Power System Optimization)

The optimization hence minimizes the annual total cost of the energy system for each planning horizon, which is defined as :

$$c = \sum_{n,s} CAPEX_{n,s} + \sum_{l} CAPEX_{l} + \sum_{t} w_t \cdot \left(\sum_{n,s} OPEX_{n,s,t} \right)$$

where :

- $CAPEX_{n,s}$ is the annualized investment cost of units at node *n* for generator or storage asset *s*
- $CAPEX_{l}$ is the annualized investment cost of infrastructure on line *l*
- $OPEX_{n,s,t}$ is the operational cost of units at node *n*, for generator or storage asset *s* at time frame *t*
- w_t is the weighting of time *t* in the objective function (e.g. multiple hours)

If *myopic* optimization is configured, the solver does not optimize the energy system over a continuous trajectory but rather planning horizon by planning horizon. For each planning horizon, the capacity installed in the previous planning horizon is taken into account and phased out assets are removed.

Given the spatial and temporal resolution used, the use of a commercial solver is required to compute the solution in a reasonable amount of time. Therefore, a Gurobi license, which is the standard choice in the PyPSA community, has been bought.

Optimization constraints

The energy system is optimized under various constrains, among which :

- A carbon budget constraint, representing how much CO2 can be emitted over the operation of the energy system for the considered planning horizons ;
- A potential constraint, representing the maximal capacity that can be installed per technology and per node. Currently, renewables are the only technologies for which this potential constraint applies.

2.1.2 Main limitations

The current study has been developed on two successive versions of PyPSA-Eur (v0.8.0 and v0.10.0) (https://github.com/PyPSA/pypsa-eur/releases/tag/v0.10.0). Please refer to the release notes for further details on fixes and improvements since then. A limitations list is also maintained by the PyPSA-Eur team. During the development of this project, the following attention points have been identified :

- For industry, the production of different materials per country is assumed to remain constant and no industry demand elasticity is included in the modelled. Hence, no Demand Side Management is modeled for the industry.
- Energy carrier transmission/transport:
  - Hydrogen and electricity are by default regionalized (i.e. considered at for each nodes) and transmission infrastructure are modeled.
  - Methane and CO2 are by default regionalized without transport infrastructures. We recommend to include these infrastructures.

  - Solid biomass is by default modeled as a single equivalent EU node. The energy vector can be regionalized with or without transport between nodes. Transport of solid biomass between nodes does not include capital costs but only marginal costs and maximum nominal power between two nodes. The current configuration consider regionalized topology and transport.
  - Heat is regionalized but excluding district heating, heat is not transported.
  - Ammonia and oil are by default modeled as a single equivalent EU node, but can be regionalized (without transport infrastructure). We recommand to regionalize these carriers.
  - Coal, lignite and methanol are by default modeled as a single equivalent EU node, but can be regionalized. We recommand to keep the EU node.
  - Uranium is not regionalized and a single equivalent EU node is considered.
- Some technologies are not yet supported by the model, including:
  - Geothermal energy for heat or electricity production ;
  - Industrial heat pumps ;
  - Low TRL storage technologies (Flow batteries, Compressed Air Energy Storage, gravitary storage system, Pumped Thermal Energy Storage, etc) ;
- Exports are not taken into account in the current version of the model. The only way to model them is exogenous.
- Hydrogen can be:
  - used as a feedstock/energy carrier for industry and as an energy carrier for different processes ;
  - produced and is not imported ;
  - transported in a gaseous form through pipelines ;
  - used as liquefied hydrogen or hydrogen for shipping demand.

2.2 Configurations

PyPSA-Eur is able to provide the energy supply of an energy system given :

- The configuration of the system by year-varying parameters (such as carbon budget or primary route share in steel production) and fixed parameters (such as maximum potential per renewable technologies or charging power of EVs) ;
- The list of technologies used ;
- Techno-economic parameters (such as investment costs, efficiency, FOM, VOM, lifetime, discount rate, etc.).

The quality of the optimization results depends on the configuration quality, as :

- The evolution trajectory of year-dependent parameters has an impact on the energy demand (by dispatching the total demand over different vectors). Fixed parameters have an impact on some technologies (through potential, minimum capacity factor, etc.) ;
- The addition of a technology might lead to an energy system significantly cheaper throughout the optimization, due to otherwise non-existing or uninteresting interactions between technologies ;
- Some technologies might not be considered in the cost-optimal system because of too high CAPEX/OPEX, or might be massively installed because of too optimistic costs.

The way PyPSA deals with those different topics is explained in the following sections.

2.2.1 Technological assumptions

PyPSA-Eur optimization of the energy system is done by computing the cost-optimal sizing of each technology per geographical location.

A technology can be used for

- Generation of energy using energy carrier(s) to produce other energy carrier(s) :
  - i.e. Using coal to produce electricity, CO2 and captured CO2 for a coal powerplant with CC ;
  - i.e. Using hydrogen and captured CO2 to produce synthetic oil for the Fischer-Tropsh process;
- Storage of energy under a specific energy vector:
  - i.e. Centralized Thermal Energy Storage;
  - i.e. Hydrogen underground storage;
- Transmission of energy vector between two geographical locations:
  - i.e. AC and DC lines;
  - i.e. Methane pipelines;
  - i.e. CO2 pipelines.

Some technologies are added to the system only if an energy sector is considered in the optimization. An exhaustive list is given here below, sorted by module and with each energy carriers the technology uses.

Base technologies

Technology name	Carrier 1	Carrier 2	Carrier 3	Carrier 4
AC lines	Elec	•	•	•
Allam power cycle	Gas	Elec	CO2 CC	•
Ammonia storage	NH3	•	•	•
Ammonia cracker	Elec	NH3	H2	•
Battery storage	Elec battery	•	•	•
CCGT	Gas	Elec	CO2	•
CCGT CC	Gas	Elec	CO2	CO2 CC
CCGT H2	H2	Elec	•	•

continues on next page

Table 1 – continued from previous page

Technology name	Carrier 1	Carrier 2	Carrier 3	Carrier 4
CH4 (g) pipeline	Gas	•	•	•
CO2 pipeline	CO2 CC	•	•	•
CO2 sequestration	CO2 CC	•	•	•
Coal	Coal	Elec	CO2	•
Coal CC	Coal	Elec	CO2 CC	CO2
DC links	Elec (DC)	Elec	•	•
Electricity distribution grid	Elec	Elec (LV)	•	•
Electricity grid connection	Elec	•	•	•
Electrolysis	Elec	H2	(Central Heat)	•
Fuel cell	H2	Elec	(Central Heat)	•
H2 (g) pipeline	H2	•	•	•
H2 (g) pipeline repurposed	H2	•	•	•
Haber-Bosch	Elec	H2	NH3	•
Helmeth	Elec	CH4	CO2	•
Home battery storage	Elec battery	•	•	•
Hydro	Hydro	Elec	•	•
Hydrogen storage overground	H2	•	•	•

continues on next page

Table 1 – continued from previous page

Technology name	Carrier 1	Carrier 2	Carrier 3	Carrier 4
Hydrogen storage underground	H2	•	•	•
Lignite	Lignite	Elec	•	•
Methanation/Sabatier	H2	Gas	CO2	•
Nuclear	Uranium	Elec	•	•
OCGT	Gas	Elec	CO2	•
OCGT CC	Gas	Elec	CO2	CO2 CC
OCGT H2	H2	Elec	•	•
Offwind-AC	Wind	Elec	•	•
Offwind-DC	Wind	Elec	•	•
Onwind	Wind	Elec	•	•
PHS	Hydro	Elec	•	•
RoR	Hydro	Elec	•	•
SMR	Gas	H2	CO2	•
SMR CC	Gas	H2	CO2 CC	CO2
Solar utility	Solar	Elec	•	•
Solar rooftop	Solar	Elec	•	•
Oil	Oil	Elec	CO2	•

Heat technologies

Technology name	Carrier 1	Carrier 2	Carrier 3	Carrier 4
Central air-sourced heat pump	Elec	Heat buses	•	•
Central gas boiler	Gas	Heat buses	•	•
Central gas CHP	Gas	Elec	Heat buses	•
Central gas CHP CC	Gas	Elec	Heat buses	CO2 CC
Central ground-sourced heat pump	Elec	Heat buses	•	•
Central resistive heater	Elec	Heat buses	•	•
Central solar thermal	Heat buses	•	•	•
Central solid biomass CHP	Biomass	Elec	Heat	•
Central solid biomass CHP CC	Biomass	Elec	Heat	CO2 CC
Central water tank storage	Heat storage	•	•	•
Decentral air-sourced heat pump	Elec	Heat buses	•	•
Decentral gas boiler	Gas	Heat buses	•	•
Decentral ground-sourced heat pump	Elec	Heat buses	•	•
Decentral resistive heater	Elec	Heat buses	•	•
Decentral solar thermal	Heat buses	•	•	•
Decentral water tank storage	Heat storage	•	•	•
Direct air capture	CO2	CO2 CC	Elec	Urban heat
Micro CHP	Gas	Elec	Heat	•
Retrofitting	Retrofitting	Heat buses		
Water tank charger	Heat storage	Heat buses	•	•
Water tank discharger	Heat storage	Heat buses	•	•

Biomass technologies

Technology name	Carrier 1	Carrier 2	Carrier 3	Carrier 4
Biogas	Biogas	•	•	•
Biogas upgrading	Biogas	Gas	•	•
Biomass boiler	Biomass	Heat	•	•
BioSNG	Biomass	Gas	•	•
BioSNG CC	Biomass	Gas	CO2 CC	•
BtL	Biomass	Oil	•	•
BtL CC	Biomass	Oil	CO2 CC	•
Central solid biomass CHP	Biomass	Elec	Heat	•
Central solid biomass CHP CC	Biomass	Elec	Heat	CO2 CC
Solid Biomass	Biomass	•	•	•

Biomass and Heat technologies

Technology name	Carrier 1	Carrier 2	Carrier 3	Carrier 4
Central solid biomass CHP	Biomass	Elec	Heat	•
Central solid biomass CHP CC	Biomass	Elec	Heat	CO2 CC

Industry technologies

Technology name	Carrier 1	Carrier 2	Carrier 3	Carrier 4
Ammonia load for ind	Ammonia	•	•	•
Biomass for ind	Biomass	Biomass for Ind	•	•
Biomass for ind CC	Biomass	Biomass for Ind	CO2 CC	•
Decentral oil boiler	Oil	Heat	•	•
Fischer-Tropsch	H2	Oil	(Central Heat)	
Gas for ind	Gas	Gas for Ind	•	•
Gas for ind CC	Gas	Gas for Ind	CO2 CC	
H2 liquefaction	H2	H2 liquid	•	•
Methanol load for ind	MeOH	•	•	•
Methanolisation	H2	MeOH	Elec	•
Oil boilers	oil	heat_buses (rural + urban decentral)		
Process CO2 emissions	CO2 process	•	•	•
Process CO2 emissions CC	CO2 process	CO2 CC	•	•
Transport ship Methanol	H2	MeOH	Elec	•

Transport technologies

Technology name	Carrier 1	Carrier 2	Carrier 3	Carrier 4
V2G	EV	Elec	•	•
DSM	Elec	EV	•	•

2.2.2 Techno-economic parameters

The default definition of the technologies in PyPSA is done by retrieving data from a cost database and formatting it into the metrics used by PyPSA-Eur, namely :

- Annualized Capital cost (€/MW/year)
- Marginal cost (EUR/MWh)
- Lifetime (years)
- Efficiency(ies) (MWhout/MWhin)
- CO2 intensity (tCO2/MWhout)
- Potential (MWhmax)
- Carrier(s)

The cost database (https://github.com/pypsa/technology-data) has a granularity of up to 5 years and is mostly based on the Danish Energy Agency (DEA) forecasts (March 2018 - August 2023). The version v0.6.2 (https://github.com/PyPSA/technology-data/tree/v0.6.2) has been thoroughly reviewed by Climact and UGent to produce tailor-made files specific to each scenario considered.

It must be noted nonetheless that for some technologies, some techno-economic parameters are set from the configuration file instead of the cost database. Those values have been reviewed as well (see *Configuration file*).

2.2.3 Configuration file

PyPSA-Eur optimization is mostly based on the choice of the technologies used and the techno-economic parameters.

Some additional parameters can nonetheless be set from a separate configuration file. Those parameters can be grouped under different categories :

- On/off technology use : Levers (de)activating some technologies in PyPSA optimization
  - i.e. Conventional technologies to consider in future planning horizons;
  - i.e. Use of micro-CHP, solid biomass to liquid, etc.;
  - i.e. Considering distribution electric and/or gas networks;
- Technology parameters : techno-economic parameters that were not set from the cost database or that alter technologies
  - i.e. Potentials and correction factors for renewables;
  - i.e. Heat pump sink temperature;
- Demand-related parameters : share between different energy carriers of a given demand. They can be fixed over the explored time horizons or year-dependent
  - i.e. Share of primary route in steel production;
  - i.e. Share of EV/ICE/FC vehicles for land transport compared to today's demand;
  - i.e. Share of HVC routes compared to today's demand;
- Simulation parameters : parameters impacting the optimization constraints and energy system definition
  - i.e. Temporal scale for the system optimization
  - i.e. Carbon budget per year (how much CO2 can be emitted annually);
  - i.e. Authorized expansion of AC/DC transmission lines (in terms of cost or transmission capacity);

- i.e. Regionalized/copperplated ammonia at EU scale;
- i.e. Emission pricing and sequestration costs per tCO2;
- i.e. Locations where hydrogen storage is allowed;

Those additional parameters default values can be modified to match expert's best estimate.

Spatio-temporal specifications

PyPSA is technically able to define the energy supply down to a resolution of 1 hour and down to the spatial resolution of ENTSO-E transmission network. However, practically speaking, such a fine resolution (8760h on one year for ~8800 nodes) is not feasible due to the huge computational burden linked to the optimization of such an energy system.

The system is hence clustered to a smaller number of equivalent nodes (i.e. clusters), small enough to allow acceptable runtimes but large enough to ensure a detailed representation of the energy system (power demand, renewable power generation, transmission infrastructures, etc.).

As mentioned in [1], we need to be especially aware of the implications of those hypotheses. Model outputs are strongly influenced by network resolution. This is why we chose to take 37 clustered nodes into account while considering 181 renewables generation sites (onshore and offshore wind as well as utility-scale solar PV technologies). This gives a better estimation of the load factors for renewables without significantly increasing the computation time.

Temporal resolution has also been explored during the preliminary phase of the project. Two resolution techniques were proposed : time aggregation and time segmentation. Time aggregation averages timesteps on a given resolution (e.g.: 3h aggregation). Time segmentation use the *tsam* package (https://github.com/FZJ-IEK3-VSA/tsam). This package looks for typical periods using machine learning algorithms. While having an impact on the computation time, we preferred a 3h time aggregation to be as close as possible to profiles. This choice also eases the interpretation of results.

More details about the spatial resolution are given in Section *Spatial resolution*.

2.3 Model modifications

To better match client needs, various modifications have been added to PyPSA-Eur. To implement this, two approaches have been considered :

1. when feasible, the request is generalized and submitted to the community as pull request (see *Submitted pull requests*) ;
2. if not, a custom modification is added to the model (see *Custom modifications*).

2.3.1 Submitted pull requests

Here is the list of pull requests shared with the community. Depending on the pull request, the modification has been integrated or not to the official repository of PyPSA-Eur.

- Define methanol energy demand for industry (https://github.com/PyPSA/pypsa-eur/pull/1068) ;
- Improve run_gurobi to wait for available token (https://github.com/PyPSA/linopy/pull/281) ;
- Fix non steel related coal demand during transition (using sector_ratios_fraction_future) (https://github.com/PyPSA/pypsa-eur/pull/1047) ;
- Add calculate_nodal_supply_energy in make summary (https://github.com/PyPSA/pypsa-eur/pull/1046) ;
- Fix gas network retrofit in brownfield (https://github.com/PyPSA/pypsa-eur/pull/1036) ;

- Improve *agg_p_nom_limits* configuration (https://github.com/PyPSA/pypsa-eur/pull/1023) ;
- Fix small issues in *add_land_use_constraint_m* (https://github.com/PyPSA/pypsa-eur/pull/1022) ;
- Clarify suffix usage in add existing baseyear (https://github.com/PyPSA/pypsa-eur/pull/1017) ;
- Fix custom busmap read in cluster network (https://github.com/PyPSA/pypsa-eur/pull/1008) ;
- Fix typo (https://github.com/PyPSA/pypsa-eur/pull/1005, https://github.com/PyPSA/pypsa-eur/pull/1045) ;
- Fix fill missing in industry sector ratios intermediate (https://github.com/PyPSA/pypsa-eur/pull/1004) ;
- Fix index for existing capacities in add_existing_baseyear (https://github.com/PyPSA/pypsa-eur/pull/992) ;
- Fix grouping year reference in add_land_use_constraint_m (https://github.com/PyPSA/pypsa-eur/pull/991) ;
- Fix error with symbol of buses in simplify_network (https://github.com/PyPSA/pypsa-eur/pull/987) ;
- Fix type error in cluster_network with "m" configuration (https://github.com/PyPSA/pypsa-eur/pull/986) ;
- Fix duplicated years in *add_land_use_constraint_m* (https://github.com/PyPSA/pypsa-eur/pull/968) ;
- Add decommissioning of renewables assets (https://github.com/PyPSA/pypsa-eur/pull/959) ;
- Add warning when negative bev availability profile values (https://github.com/PyPSA/pypsa-eur/pull/858) ;
- Fix typo in buses definition for oil boilers in *add_industry* in *prepare_sectors_networks* (https://github.com/PyPSA/pypsa-eur/pull/812) ;
- Fix nodal fraction with distributed generators (https://github.com/PyPSA/pypsa-eur/pull/798) ;
- Add option for SMR CC (https://github.com/PyPSA/pypsa-eur/pull/757) ;
- Add rule to update IRENA renewables capacities (https://github.com/PyPSA/pypsa-eur/pull/756) ;
- Improve Gurobi usage for *linopy* package (https://github.com/PyPSA/linopy/pull/162) ;
- Raised issue for *snakemake* package to better manage Gurobi licenses (https://github.com/snakemake/snakemake/issues/1801) ;
- Raised issue for *pulp* package to better manage Gurobi licenses (https://github.com/coin-or/pulp/issues/571).

2.3.2 Custom modifications

When no suitable for a pull request, the customisation is added on our fork of the model. Techno-economic assumptions have been thoroughly reviewed as described in *Techno-economic parameters*.

- Add project specific configurations ;
- Develop a Streamlit app available at https://climact-veka-2050plus.streamlit.app/ ;
- Develop a data management pipeline to produce files required by the app ;
- Define a set of configurations and techno-economic assumptions, needed for scenario management ;
- Add custom configuration to model Flanders as a node ;
- Add custom configuration to model nuclear powerplants ;
- Fix PV shares in Belgium to reflect current imbalance between regions ;
- Split HVC primary route into three types of routes (NSC, NSC CC and MTO) to include VLAIO scenarios for Flanders ;
- Split DRI route of steel in two routes (DRI CH4 and DRI H2), needed for scenario management of industry. DRI CH4 is model using MIDREX process ;
- Fix BEV dsm restriction time to take time aggregation and time zones into account ;

- Add emission prices for each planning horizon ;
- Update IRENA data to 2023 ;
- Update status of TYNDP links to the latest version available ;
- Limit offshore wind capacity in Belgium to a realistic value ;
- Add a proxy to model distribution costs of gas boilers ;
- Update OCGT, CCGT and coal CC efficiencies to include electricity demand of CC ;
- Add CC and H2 gas turbines ;
- Add hydrogen import terminals ;
- Add a marginal costs for gas and hydrogen pipelines to model operating costs ;
- Add a rule to produce map with capacities ;
- Add a rule to manage multiple set of costs, needed for scenario management ;
- Generalize *agg_p_nom_minmax* constraint to handle nuclear and all types of offshore wind ;
- Fix config provider to work as expected with reference scenario ;
- Stick to version 0.5.11 of https://github.com/PyPSA/powerplantmatching to ensure stability ;
- Fix various minor issues.

2.4 Starter kit

To start using the model and its dependencies, multiple scripts and tool should be used.

1. **Get the model**
    - Clone the git repository from GitHub.
    - Install your environment using *envs* folder (latest data were produced with *envs/environment.aws_r6a.12xlarge.yaml*). A detailed installation guide is available in the PyPSA-Eur documentation.
    - Optimisation has been done using Gurobi (commercial solver). HiGHS is a good open source alternative. Other alternatives are listed in the PyPSA-Eur documentation.
2. **Run the model**
    - Use snakemake using the following command

```
snakemake -call all --configfile config/config.veka.yaml -n
```

    - Model can be ran locally or on a remote server to improve performances (should work on a AWS EC2 r6a.12xlarge).
    - Edit the configurations files to improve the scenarios : *config/config.veka.yaml* and *config/scenarios.veka.yaml*.
3. **Extract data from the PyPSA-Eur outputs**
    - Copy desired files from *results* and *resources* folders to *analysis/{run}* folder. If you use a remote server, use SFTP to download data from the server.
    - Configure the correct run versions in *scripts/graph_extraction_main.py* for scenarios, sensitivities and reference runs.

- Run the data extraction pipeline (locally).
- The output data (*analysis/{run}/graph_extraction_st*) are created in *analysis* folder and should be copied in the *app/assets/data* folder if you want to add them to the streamlit.

4. **Ship the data in streamlit**
    - Streamlit platform is free to use when the data are available on Github publicly. Currently our website is linked to *main-v10* branch on the Github of Climact.
    - To use the new data, reconfigure *app/st_common.py* with the new runs paths (change *scenario_dict* values).
    - Merge the updated data with the GitHub branch *main-v10* to ship the data online.
    - Streamlit will automatically update the webpage. If you want to be sure that cached data are updated, you can reboot the app on the Streamlit dashboard.
    - The webpage can be debugged locally using:

```
streamlit run app/VEKA_2050+.py
```

    - Since Streamlit can also be run as a container, we have added *app/Dockerfile_demo* as an example.

5. **The documentation is shared online through Readthedocs.**
    - Pushing new updates on the *main-v10* branch on GitHub will also automatically update the page.
    - Documentation is stored in the *doc* folder.
    - The documentation can be tested locally using:

```
cd doc
make clean
make html
open _build/html/index.html
```

    - The documentation is also configured to generate a PDF file using:

```
cd doc
make clean
# Need twice for LaTeX references
make latexpdf; make latexpdf
open _build/latex/PyPSA-Eur.pdf
```

CHAPTER

THREE

GETTING STARTED

3.1 Introduction

Note: Find the introductory slides here.

Warning: The video only introduces the electricity-only part of PyPSA-Eur.

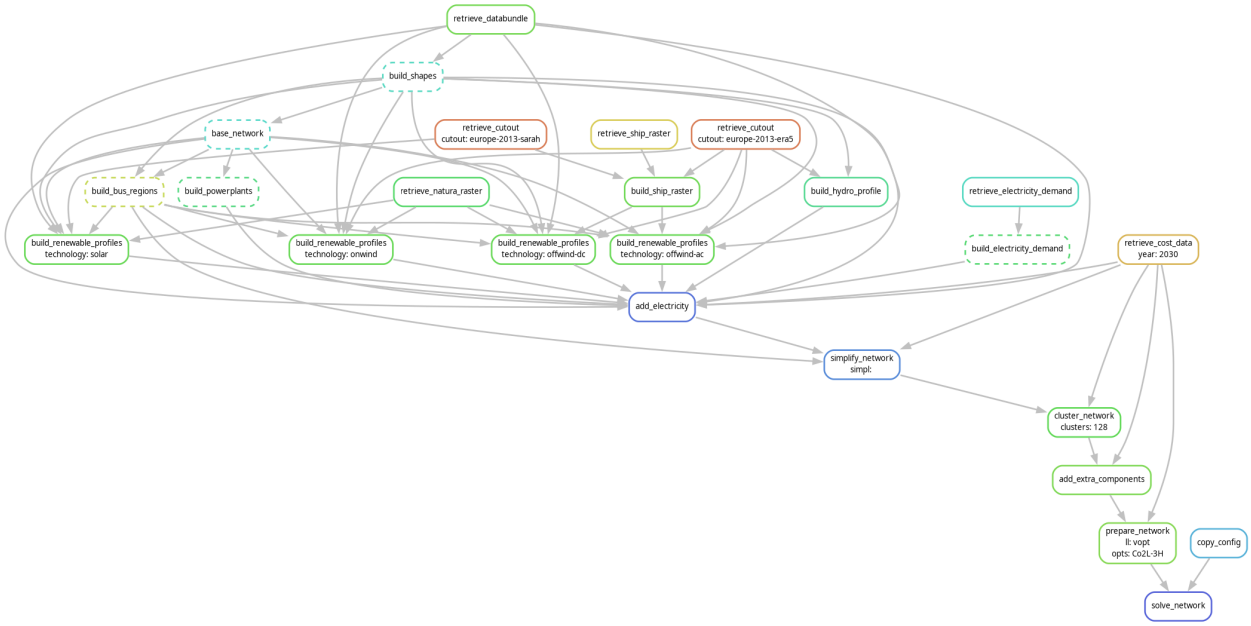
3.1.1 Workflow

The generation of the model is controlled by the open workflow management system Snakemake. In a nutshell, the `Snakefile` declares for each script in the `scripts` directory a rule which describes which files the scripts consume and produce (their corresponding input and output files). The `snakemake` tool then runs the scripts in the correct order according to the rules' input and output dependencies. Moreover, `snakemake` will track what parts of the workflow have to be regenerated when files or scripts were modified.

For instance, an invocation to

```
.../pypsa-eur % snakemake -call results/networks/elec_s_128_ec_lvopt_Co2L-3H.nc
```

follows this dependency graph



to solve an electricity system model.

The **blocks** represent the individual rules which are required to create the file referenced in the command above. The **arrows** indicate the outputs from preceding rules which another rule takes as input data.

Note: The dependency graph was generated using `snakemake --dag results/networks/elec_s_128_ec_lvopt_Co2L-3H.nc -F | sed -n "/digraph/,/}/p" | dot -Tpng -o doc/img/intro-workflow.png`

For the use of `snakemake`, it makes sense to familiarize yourself quickly with the basic tutorial and then read carefully through the documentation of the command line interface, noting the arguments `-j`, `-c`, `-f`, `-F`, `-n`, `-r`, `--dag` and `-t` in particular.

3.1.2 Scenarios, Configuration and Modification

It is easy to run PyPSA-Eur for multiple scenarios using the wildcards feature of `snakemake`. Wildcards allow to generalise a rule to produce all files that follow a regular expression pattern, which defines a particular scenario. One can think of a wildcard as a parameter that shows up in the input/output file names and thereby determines which rules to run, what data to retrieve and what files to produce. Details are explained in *Wildcards* and *scenario*.

The model also has several further configuration options collected in the `config/config.default.yaml` file located in the root directory, which that are not part of the scenarios. Options are explained in *Configuration*.

3.1.3 Folder Structure

- `scripts`: Includes all the Python scripts executed by the `snakemake` rules.
- `rules`: Includes all the `snakemake` rules loaded in the `Snakefile`.
- `envs`: Includes all the `conda` environment specifications to run the workflow.
- `data`: Includes input data that is not produced by any `snakemake` rule.
- `cutouts`: Stores raw weather data cutouts from `atlite`.
- `resources`: Stores intermediate results of the workflow which can be picked up again by subsequent rules.
- `results`: Stores the solved PyPSA network data, summary files and plots.
- `logs`: Stores log files.
- `benchmarks`: Stores `snakemake` benchmarks.
- `doc`: Includes the documentation of PyPSA-Eur.
- `graphics`: Includes some graphics for the documentation of PyPSA-Eur.

3.1.4 System Requirements

Building the model with the scripts in this repository runs on a regular computer. But optimising for investment and operation decisions across many scenarios requires a strong interior-point solver like Gurobi or CPLEX with more memory. Open-source solvers like *HiGHS <https://highs.dev>* can also be used for smaller problems.

3.2 Installation

The subsequently described installation steps are demonstrated as shell commands, where the path before the `%` sign denotes the directory in which the commands following the `%` should be entered.

3.2.1 Clone the Repository

First of all, clone the PyPSA-Eur repository using the version control system `git` in the command line.

```
/some/other/path % cd /some/path

/some/path % git clone https://github.com/PyPSA/pypsa-eur.git
```

3.2.2 Install Python Dependencies

PyPSA-Eur relies on a set of other Python packages to function. We recommend using the package manager mamba to install them and manage your environments. For instructions for your operating system follow the `mamba` installation guide. You can also use `conda` equivalently.

The package requirements are curated in the envs/environment.yaml file. The environment can be installed and activated using

```
.../pypsa-eur % mamba env create -f envs/environment.yaml

.../pypsa-eur % mamba activate pypsa-eur
```

Note: The equivalent commands for `conda` would be

```
.../pypsa-eur % conda env create -f envs/environment.yaml

.../pypsa-eur % conda activate pypsa-eur
```

3.2.3 Install a Solver

PyPSA passes the PyPSA-Eur network model to an external solver for performing the optimisation. PyPSA is known to work with the free software

- HiGHS
- Cbc
- GLPK (WinGLKP)
- Ipopt

and the non-free, commercial software (for some of which free academic licenses are available)

- Gurobi
- CPLEX
- FICO Xpress Solver

For installation instructions of these solvers for your operating system, follow the links above. Commercial solvers such as Gurobi and CPLEX currently significantly outperform open-source solvers for large-scale problems, and it might be the case that you can only retrieve solutions by using a commercial solver. Nevertheless, you can still use open-source solvers for smaller problems.

See also:

Instructions how to install a solver in the documentation of PyPSA

Note: The rules *cluster_network* and *simplify_network* solve a mixed-integer quadratic optimisation problem for clustering. The open-source solvers HiGHS, Cbc and GlPK cannot handle this. A fallback to SCIP is implemented in this case. For an open-source solver setup install in your `conda` environment on OSX/Linux. To install the default solver Gurobi, run

```
mamba activate pypsa-eur
mamba install -c gurobi gurobi
```

Additionally, you need to setup your Gurobi license.

3.2.4 Handling Configuration Files

PyPSA-Eur has several configuration options that users can specify in a `config/config.yaml` file. The default configuration `config/config.default.yaml` is maintained in the repository. More details on the configuration options are in *Configuration*.

You can also use `snakemake` to specify another file, e.g. `config/config.mymodifications.yaml`, to update the settings of the `config/config.yaml`.

```
.../pypsa-eur % snakemake -call --configfile config/config.mymodifications.yaml
```

3.3 Tutorial: Electricity-Only

Note: If you have not done it yet, follow the *Installation* steps first.

In this tutorial, we will build a heavily simplified power system model for Belgium. But before getting started with **PyPSA-Eur** it makes sense to be familiar with its general modelling framework PyPSA.

Running the tutorial requires limited computational resources compared to the full model, which allows the user to explore most of its functionalities on a local machine. The tutorial will cover examples on how to configure and customise the PyPSA-Eur model and run the `snakemake` workflow step by step from network creation to the solved network. The configuration for the tutorial is located at `config/test/config.electricity.yaml`. It includes parts deviating from the default config file `config/config.default.yaml`. To run the tutorial with this configuration, execute

```
snakemake -call results/test-elec/networks/elec_s_6_ec_lcopt_Co2L-24H.nc --configfile␣
↪config/test/config.electricity.yaml
```

This configuration is set to download a reduced data set via the rules *retrieve_databundle*, `retrieve_natura_raster`, `retrieve_cutout`. For more information on the data dependencies of PyPSA-Eur, continue reading *Retrieving Data*.

3.3.1 How to configure runs?

The model can be adapted to only include selected countries (e.g. Belgium) instead of all European countries to limit the spatial scope.

```
countries: ['BE']
```

Likewise, the example's temporal scope can be restricted (e.g. to a single week).

```
snapshots:
  start: "2013-03-01"
  end: "2013-03-08"
```

It is also possible to allow less or more carbon-dioxide emissions. Here, we limit the emissions of Belgium to 100 Mt per year.

```
electricity:
  co2limit: 100.e+6
```

PyPSA-Eur also includes a database of existing conventional powerplants. We can select which types of existing powerplants we like to be extendable:

```
  extendable_carriers:
    Generator: [OCGT]
    StorageUnit: [battery]
    Store: [H2]
    Link: [H2 pipeline]
```

To accurately model the temporal and spatial availability of renewables such as wind and solar energy, we rely on historical weather data. It is advisable to adapt the required range of coordinates to the selection of countries.

```
atlite:
  default_cutout: be-03-2013-era5
  cutouts:
    be-03-2013-era5:
      module: era5
      x: [4., 15.]
      y: [46., 56.]
      time: ["2013-03-01", "2013-03-08"]
```

We can also decide which weather data source should be used to calculate potentials and capacity factor time-series for each carrier. For example, we may want to use the ERA-5 dataset for solar and not the default SARAH-2 dataset.

```
  solar:
    cutout: be-03-2013-era5
```

Finally, it is possible to pick a solver. For instance, this tutorial uses the open-source solver GLPK.

```
  solver:
    name: glpk
    options: "glpk-default"
```

Note, that `config/test/config.electricity.yaml` only includes changes relative to the default configuration. There are many more configuration options, which are documented at *Configuration*.

3.3.2 How to use `snakemake` rules?

Open a terminal, go into the PyPSA-Eur directory, and activate the `pypsa-eur` environment with

```
mamba activate pypsa-eur
```

Let's say based on the modifications above we would like to solve a very simplified model clustered down to 6 buses and every 24 hours aggregated to one snapshot. The command

```
snakemake -call results/test-elec/networks/elec_s_6_ec_lcopt_Co2L-24H.nc --configfile␣
↪config/test/config.electricity.yaml
```

orders `snakemake` to run the rule *`solve_network`* that produces the solved network and stores it in `results/networks` with the name `elec_s_6_ec_lcopt_Co2L-24H.nc`:

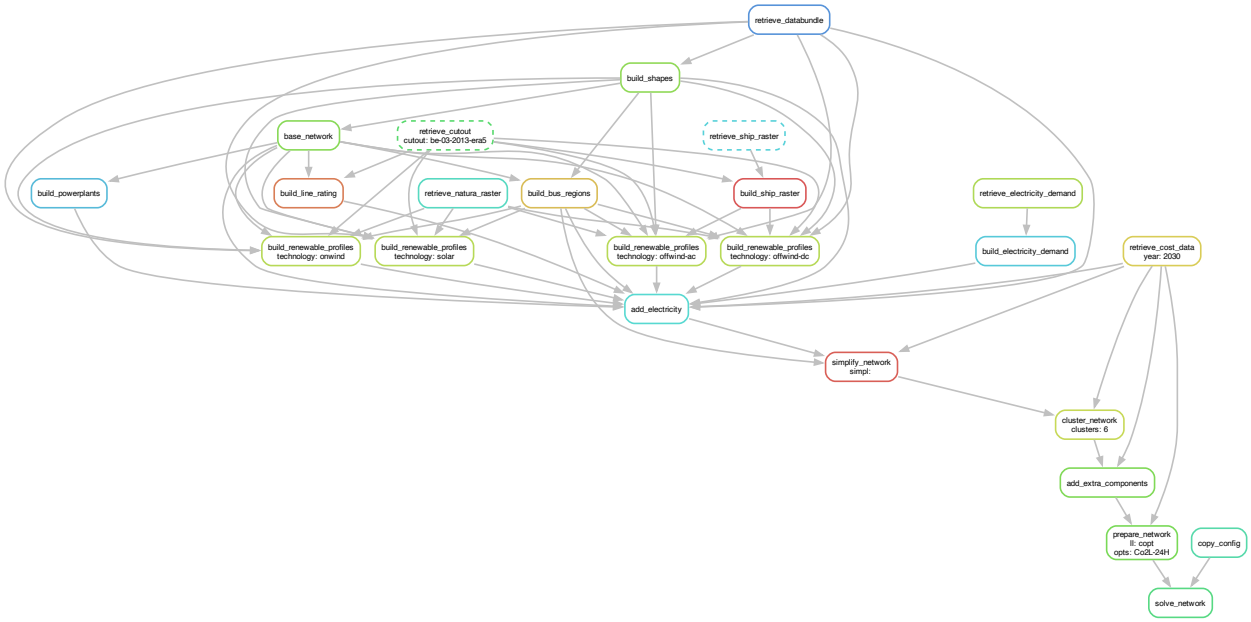
```
rule solve_network:
    params:
        solving=config_provider("solving"),
```

(continues on next page)

(continued from previous page)

```
        foresight=config_provider("foresight"),
        planning_horizons=config_provider("scenario", "planning_horizons"),
        co2_sequestration_potential=config_provider(
            "sector", "co2_sequestration_potential", default=200
        ),
        custom_extra_functionality=input_custom_extra_functionality,
    input:
        network=resources("networks/elec_s{simpl}_{clusters}_ec_l{ll}_{opts}.nc"),
    output:
        network=RESULTS + "networks/elec_s{simpl}_{clusters}_ec_l{ll}_{opts}.nc",
        config=RESULTS + "configs/config.elec_s{simpl}_{clusters}_ec_l{ll}_{opts}.yaml",
    log:
        solver=normpath(
            RESULTS
            + "logs/solve_network/elec_s{simpl}_{clusters}_ec_l{ll}_{opts}_solver.log"
        ),
        python=RESULTS
        + "logs/solve_network/elec_s{simpl}_{clusters}_ec_l{ll}_{opts}_python.log",
    benchmark:
        (RESULTS + "benchmarks/solve_network/elec_s{simpl}_{clusters}_ec_l{ll}_{opts}")
    threads: solver_threads
    resources:
        mem_mb=memory,
        runtime=config_provider("solving", "runtime", default="6h"),
    shadow:
        "shallow"
    conda:
        "../envs/environment.yaml"
    script:
        "../scripts/solve_network.py"
```

This triggers a workflow of multiple preceding jobs that depend on each rule's inputs and outputs:



In the terminal, this will show up as a list of jobs to be run:

```
Building DAG of jobs...
Job stats:
job                            count
---------------------------  -------
add_electricity                    1
add_extra_components               1
base_network                       1
build_bus_regions                  1
build_electricity_demand           1
build_line_rating                  1
build_powerplants                  1
build_renewable_profiles           4
build_shapes                       1
build_ship_raster                  1
cluster_network                    1
copy_config                        1
prepare_network                    1
retrieve_cost_data                 1
retrieve_databundle                1
retrieve_electricity_demand        1
retrieve_natura_raster             1
simplify_network                   1
solve_network                      1
total                             22
```

`snakemake` then runs these jobs in the correct order.

A job (here `simplify_network`) will display its attributes and normally some logs below this block:

```
[Mon Feb 19 17:06:17 2024]
rule simplify_network:
    input: resources/test/networks/elec.nc, data/costs_2030.csv, resources/test/regions_
↪onshore.geojson, resources/test/regions_offshore.geojson
    output: resources/test/networks/elec_s.nc, resources/test/regions_onshore_elec_s.
↪geojson, resources/test/regions_offshore_elec_s.geojson, resources/test/busmap_elec_s.
↪csv, resources/test/connection_costs_s.csv
    log: logs/test-elec/simplify_network/elec_s.log
    jobid: 4
    benchmark: benchmarks/test-elec/simplify_network/elec_s
    reason: Missing output files: resources/test/regions_offshore_elec_s.geojson,␣
↪resources/test/busmap_elec_s.csv, resources/test/regions_onshore_elec_s.geojson,␣
↪resources/test/networks/elec_s.nc; Input files updated by another job: resources/test/
↪regions_offshore.geojson, resources/test/networks/elec.nc, resources/test/regions_
↪onshore.geojson, data/costs_2030.csv
    wildcards: simpl=
    resources: tmpdir=/tmp, mem_mb=12000, mem_mib=11445
```

Once the whole worktree is finished, it should state so in the terminal.

You will notice that many intermediate stages are saved, namely the outputs of each individual `snakemake` rule.

You can produce any output file occurring in the `Snakefile` by running

```
snakemake -call <output file>
```

For example, you can explore the evolution of the PyPSA networks by running

1. `snakemake resources/networks/base.nc -call --configfile config/test/config.electricity.yaml`
2. `snakemake resources/networks/elec.nc -call --configfile config/test/config.electricity.yaml`
3. `snakemake resources/networks/elec_s.nc -call --configfile config/test/config.electricity.yaml`
4. `snakemake resources/networks/elec_s_6.nc -call --configfile config/test/config.electricity.yaml`
5. `snakemake resources/networks/elec_s_6_ec_lcopt_Co2L-24H.nc -call --configfile config/test/config.electricity.yaml`

To run all combinations of wildcard values provided in the `config/config.yaml` under `scenario:`, you can use the collection rule `solve_elec_networks`.

```
snakemake -call solve_elec_networks --configfile config/test/config.electricity.yaml
```

If you now feel confident and want to tackle runs with larger temporal and spatial scope, clean-up the repository and after modifying the `config/config.yaml` file target the collection rule `solve_elec_networks` again without providing the test configuration file.

```
snakemake -call purge
snakemake -call solve_elec_networks
```

Note: It is good practice to perform a dry-run using the option *-n*, before you commit to a run:

```
snakemake -call solve_elec_networks -n
```

3.3.3 How to analyse results?

The solved networks can be analysed just like any other PyPSA network (e.g. in Jupyter Notebooks).

```
import pypsa

n = pypsa.Network("results/networks/elec_s_6_ec_lcopt_Co2L-24H.nc")
```

For inspiration, read the examples section in the PyPSA documentation.

3.4 Tutorial: Sector-Coupled

Note: If you have not done it yet, follow the *Installation* steps first.

Also, checkout the tutorial for electricity-only systems first at *Tutorial: Electricity-Only*.

In this tutorial, we will add further sectors to the electricity-only model from *Tutorial: Electricity-Only*, namely industry, transport, and buildings. This requires processing of a few more raw data sources.

The sector-coupling code can be run as an overnight / greenfield scenario or with multi-horizon investment with myopic foresight. Pathway analysis with perfect foresight is under development. See also the documentation on *Foresight Options*.

3.4.1 Overnight Scenarios

Configuration

The default configuration file (`config/config.default.yaml`) is set up for running overnight scenarios. Running a sector-coupled model unlocks many further configuration options. In the example below, we say that the gas network should be added and spatially resolved. We also say that the existing gas network may be retrofitted to transport hydrogen instead.

```
sector:
  gas_network: true
  H2_retrofit: true
```

Documentation for all options will be added successively to *Configuration*.

Scenarios can be defined like for electricity-only studies, but with additional wildcard options.

```
scenario:
  ll:
  - v1.5
  clusters:
  - 5
  sector_opts:
```

(continues on next page)

(continued from previous page)

```
  - CO2L0-24h-T-H-B-I-A-dist1
  planning_horizons:
  - 2030
```

For allowed wildcard values, refer to *Wildcards*.

Execution

To run an overnight / greenfiled scenario with the specifications above, run

```
snakemake -call all --configfile config/test/config.overnight.yaml
```

which will result in the following jobs `snakemake` wants to run, some of which were already included in the electricity-only tutorial:

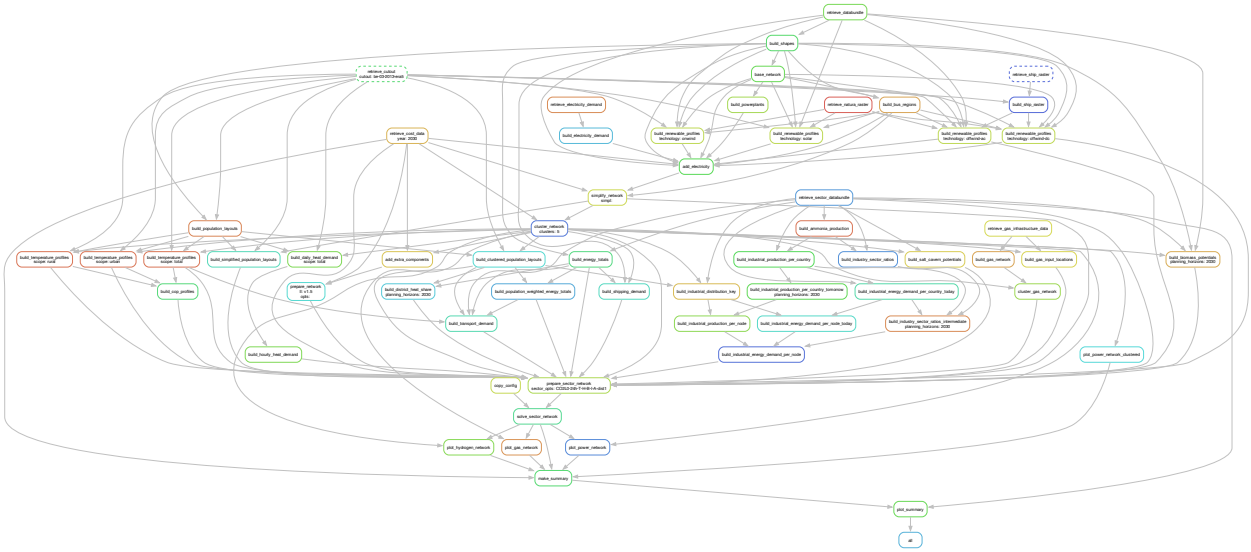
```
job                                                 count
------------------------------------------------  -------
add_electricity                                         1
add_extra_components                                    1
all                                                     1
base_network                                            1
build_ammonia_production                                1
build_biomass_potentials                                1
build_bus_regions                                       1
build_clustered_population_layouts                      1
build_cop_profiles                                      1
build_daily_heat_demand                                 1
build_district_heat_share                               1
build_electricity_demand                                1
build_energy_totals                                     1
build_gas_input_locations                               1
build_gas_network                                       1
build_hourly_heat_demand                                1
build_industrial_distribution_key                       1
build_industrial_energy_demand_per_country_today        1
build_industrial_energy_demand_per_node                 1
build_industrial_energy_demand_per_node_today           1
build_industrial_production_per_country                 1
build_industrial_production_per_country_tomorrow        1
build_industrial_production_per_node                    1
build_industry_sector_ratios                            1
build_industry_sector_ratios_intermediate               1
build_population_layouts                                1
build_population_weighted_energy_totals                 1
build_powerplants                                       1
build_renewable_profiles                                4
build_salt_cavern_potentials                            1
build_shapes                                            1
build_ship_raster                                       1
build_shipping_demand                                   1
build_simplified_population_layouts                     1
build_temperature_profiles                              3
```

(continues on next page)

(continued from previous page)

build_transport_demand	1
cluster_gas_network	1
cluster_network	1
copy_config	1
make_summary	1
plot_gas_network	1
plot_hydrogen_network	1
plot_power_network	1
plot_power_network_clustered	1
plot_summary	1
prepare_network	1
prepare_sector_network	1
retrieve_cost_data	1
retrieve_databundle	1
retrieve_electricity_demand	1
retrieve_gas_infrastructure_data	1
retrieve_natura_raster	1
retrieve_sector_databundle	1
simplify_network	1
solve_sector_network	1
total	60

This covers the retrieval of additional raw data from online resources and preprocessing data about the transport, industry, and heating sectors as well as additional rules about geological storage and sequestration potentials, gas infrastructure, and biomass potentials. The collection rule `all` will also generate summary CSV files and plots after the network has been solved successfully.



3.4.2 Myopic Foresight Scenarios

Configuration

To activate the myopic foresight mode, set

```
foresight: myopic
```

Scenarios can be defined like for electricity-only studies, but with additional wildcard options. For the myopic foresight mode, the {planning_horizons} wildcard defines the sequence of investment horizons.

```
scenario:
  ll:
  - v1.5
  clusters:
  - 5
  sector_opts:
  - 24h-T-H-B-I-A-dist1
  planning_horizons:
  - 2030
  - 2040
  - 2050
```

For allowed wildcard values, refer to *Wildcards*.

In the myopic foresight mode, you can tweak for instance exogenously given transition paths, like the one for the share of primary steel production we change below:

```
industry:
  St_primary_fraction:
    2030: 0.6
    2040: 0.5
    2050: 0.4
```

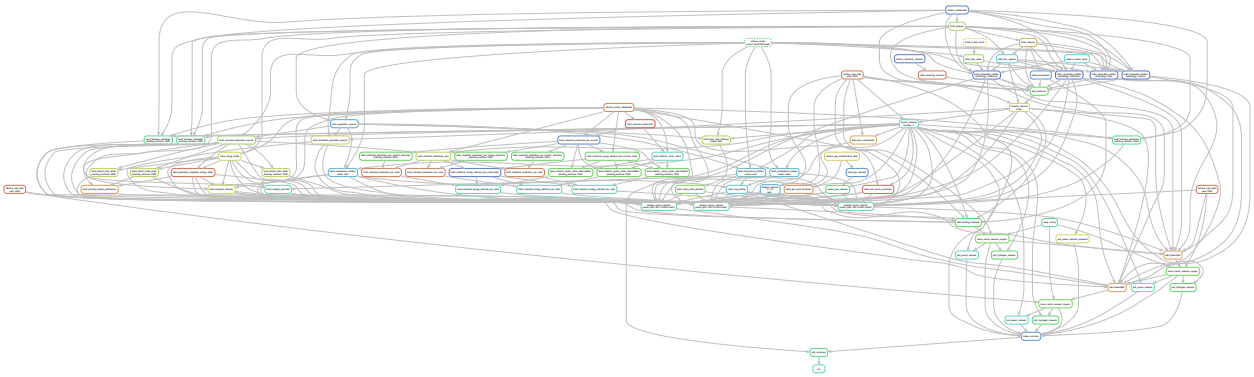
Documentation for all options will be added successively to *Configuration*.

Execution

To run a myopic foresight scenario with the specifications above, run

```
snakemake -call all --configfile config/test/config.myopic.yaml
```

which will result in additional jobs `snakemake` wants to run, which translates to the following workflow diagram which nicely outlines how the sequential pathway optimisation with myopic foresight is implemented in the workflow:



3.4.3 Scaling-Up

If you now feel confident and want to tackle runs with larger temporal, technological and spatial scope, clean-up the repository and after modifying the `config/config.yaml` file target the collection rule `all` again without providing the test configuration file.

```
snakemake -call purge
snakemake -call all
```

Note: It is good practice to perform a dry-run using the option *-n*, before you commit to a run:

```
snakemake -call all -n
```

CHAPTER

FOUR

CONFIGURATION

4.1 Wildcards

It is easy to run PyPSA-Eur for multiple scenarios using the wildcards feature of `snakemake`. Wildcards allow to generalise a rule to produce all files that follow a regular expression pattern which e.g. defines one particular scenario. One can think of a wildcard as a parameter that shows up in the input/output file names of the `Snakefile` and thereby determines which rules to run, what data to retrieve and what files to produce.

Note: Detailed explanations of how wildcards work in `snakemake` can be found in the relevant section of the documentation.

4.1.1 The `{cutout}` wildcard

The `{cutout}` wildcard facilitates running the rule *build_cutout* for all cutout configurations specified under `atlite: cutouts:`. These cutouts will be stored in a folder specified by `{cutout}`.

4.1.2 The `{technology}` wildcard

The `{technology}` wildcard specifies for which renewable energy technology to produce availability time series and potentials using the rule *build_renewable_profiles*. It can take the values `onwind`, `offwind-ac`, `offwind-dc`, and `solar` but **not** `hydro` (since hydroelectric plant profiles are created by a different rule).

4.1.3 The `{simpl}` wildcard

The `{simpl}` wildcard specifies number of buses a detailed network model should be pre-clustered to in the rule *simplify_network* (before *cluster_network*).

4.1.4 The {clusters} wildcard

The {clusters} wildcard specifies the number of buses a detailed network model should be reduced to in the rule *cluster_network*. The number of clusters must be lower than the total number of nodes and higher than the number of countries. However, a country counts twice if it has two asynchronous subnetworks (e.g. Denmark or Italy).

If an *m* is placed behind the number of clusters (e.g. 100m), generators are only moved to the clustered buses but not aggregated by carrier; i.e. the clustered bus may have more than one e.g. wind generator.

4.1.5 The {ll} wildcard

The {ll} wildcard specifies what limits on line expansion are set for the optimisation model. It is handled in the rule *prepare_network*.

The wildcard, in general, consists of two parts:

1. The first part can be v (for setting a limit on line volume) or c (for setting a limit on line cost)
2. The second part can be opt or a float bigger than one (e.g. 1.25).
   (a) If opt is chosen line expansion is optimised according to its capital cost (where the choice v only considers overhead costs for HVDC transmission lines, while c uses more accurate costs distinguishing between overhead and underwater sections and including inverter pairs).
   (b) v1.25 will limit the total volume of line expansion to 25 % of currently installed capacities weighted by individual line lengths; investment costs are neglected.
   (c) c1.25 will allow to build a transmission network that costs no more than 25 % more than the current system.

4.1.6 The {opts} wildcard

The {opts} wildcard is used for electricity-only studies. It triggers optional constraints, which are activated in either *prepare_network* or the *solve_network* step. It may hold multiple triggers separated by -, i.e. Co2L-3H contains the Co2L trigger and the 3H switch. There are currently:

Trigger	Description	Definition	Status
nH; i.e. 2H-6H	Resample the time-resolution by averaging over every n snapshots	prepare_network: average_every_nhours() and its caller)	In active use
nSEG; e.g. 4380SEG	Apply time series segmentation with tsam package to n adjacent snapshots of varying lengths based on capacity factors of varying renewables, hydro inflow and load.	prepare_network: apply_time_segmentatio	In active use
Co2L	Add an overall absolute carbon-dioxide emissions limit configured in electricity: co2limit. If a float is appended an overall emission limit relative to the emission level given in electricity: co2base is added (e.g. Co2L0.05 limits emissisions to 5% of what is given in electricity: co2base)	prepare_network: add_co2limit() and its caller	In active use
Ep	Add cost for a carbon-dioxide price configured in costs: emission_prices: co2 to marginal_cost of generators (other emission types listed in network.carriers possible as well)	prepare_network: add_emission_prices(and its caller	In active use
Ept	Add monthly cost for a carbon-dioxide price based on historical values built by the rule build_monthly_prices	In active use	
CCL	Add minimum and maximum levels of generator nominal capacity per carrier for individual countries. These can be specified in the file linked at electricity: agg_p_nom_limits in the configuration. File defaults to data/agg_p_nom_minmax.csv.	solve_network	In active use
EQ	Require each country or node to on average produce a minimal share of its total consumption itself. Example: EQ0.5c demands each country to produce on average at least 50% of its consumption; EQ0.5 demands each node to produce on average at least 50% of its consumption.	solve_network	In active use
ATK	Require each node to be autarkic. Example: ATK removes all lines and links. ATKc removes all cross-border lines and links.	prepare_network	In active use
BAU	Add a per-carrier minimal overall capacity; i.e. at least 40GW of OCGT in Europe; configured in electricity: BAU_mincapacities	solve_network: add_opts_constraints(	Untested
SAFE	Add a capacity reserve margin of a certain fraction above the peak demand to which renewable generators and storage do *not* contribute. Ignores network.	solve_network add_opts_constraints(	Untested
carrier+{c|p|m}fa	Alter the capital cost (c), installable potential (p) or marginal costs (m) of a carrier by a factor. Example: solar+c0.5 reduces the capital cost of solar to 50% of original values.	prepare_network	In active use
CH4L	Add an overall absolute gas limit. If configured in electricity: gaslimit it is given in MWh thermal, if a float is appended, the overall gaslimit is assumed to be given in	prepare_network: add_gaslimit()	In active use

4.1.7 The {sector_opts} wildcard

Warning:

More comprehensive documentation for this wildcard will be added soon. To really understand the options here, look in scripts/prepare_sector_network.py

Co2Lx specifies the CO2 target in x% of the 1990 values; default will give default (5%); # Co2L0p25 will give 25% CO2 emissions; Co2Lm0p05 will give 5% negative emissions # xH is the temporal resolution; 3H is 3-hourly, i.e. one snapshot every 3 hours # single letters are sectors: T for land transport, H for building heating, # B for biomass supply, I for industry, shipping and aviation, # A for agriculture, forestry and fishing # solar+c0.5 reduces the capital cost of solar to 50% of reference value # solar+p3 multiplies the available installable potential by factor 3 # seq400 sets the potential of CO2 sequestration to 400 Mt CO2 per year # dist{n} includes distribution grids with investment cost of n times cost in data/costs.csv # for myopic/perfect foresight cb states the carbon budget in GtCO2 (cumulative # emissions throughout the transition path in the timeframe determined by the # planning_horizons), be:beta decay; ex:exponential decay # cb40ex0 distributes a carbon budget of 40 GtCO2 following an exponential # decay with initial growth rate 0

The {sector_opts} wildcard is only used for sector-coupling studies.

Trigger	Description	Definition	Status
nH	i.e. 2H-6H	Resample the time-resolution by averaging over every n snapshots, prepare_network: average_every_nhours() and its caller)	In active use
Co2L	Add an overall absolute carbon-dioxide emissions limit configured in electricity: co2limit. If a float is appended an overall emission limit relative to the emission level given in electricity: co2base is added (e.g. Co2L0.05 limits emissions to 5% of what is given in electricity: co2base)	prepare_network: add_co2limit() and its caller	In active use
carrier+{c|p|m}fa	Alter the capital cost (c), installable potential (p) or marginal costs (m) of a carrier by a factor. Example: solar+c0.5 reduces the capital cost of solar to 50% of original values.	prepare_network	In active use
T	Add land transport sector		In active use
H	Add heating sector		In active use
B	Add biomass		In active use
I	Add industry sector		In active use
A	Add agriculture sector		In active use
dist``+``n	Add distribution grid with investment costs of n times costs in resources/costs_{cost_year}.csv		In active use
seq``+``n	Sets the CO2 sequestration potential to n Mt CO2 per year		In active use

4.1.8 The {scope} wildcard

Takes values `residential`, `urban`, `total`.

4.1.9 The {planning_horizons} wildcard

Warning: More comprehensive documentation for this wildcard will be added soon.

The {planning_horizons} wildcard is only used for sector-coupling studies. It takes years as values, e.g. 2020, 2030, 2040, 2050.

4.2 Configuration

PyPSA-Eur has several configuration options which are documented in this section and are collected in a `config/config.yaml` file. This file defines deviations from the default configuration (`config/config.default.yaml`); confer installation instructions at *Handling Configuration Files*.

4.2.1 Top-level configuration

"Private" refers to local, machine-specific settings or data meant for personal use, not to be shared. "Remote" indicates the address of a server used for data exchange, often for clusters and data pushing/pulling.

	Unit	Values	Description
version	–	0.x.x	Version of PyPSA-Eur. Descriptive only.
tutorial	bool	{true, false}	Switch to retrieve the tutorial data set instead of the full data set.
logging			
– level	–	Any of {'INFO', 'WARNING', 'ERROR'}	Restrict console outputs to all infos, warning or errors only
– format	–		Custom format for log messages. See LogRecord attributes.
private			
– keys			
– – entsoe_api	–		Optionally specify the ENTSO-E API key. See the guidelines to get ENTSO-E API key
remote			
– ssh	–		Optionally specify the SSH of a remote cluster to be synchronized.
– path	–		Optionally specify the file path within the remote cluster to be synchronized.

4.2.2 `run`

It is common conduct to analyse energy system optimisation models for **multiple scenarios** for a variety of reasons, e.g. assessing their sensitivity towards changing the temporal and/or geographical resolution or investigating how investment changes as more ambitious greenhouse-gas emission reduction targets are applied.

The `run` section is used for running and storing scenarios with different configurations which are not covered by *Wildcards*. It determines the path at which resources, networks and results are stored. Therefore the user can run different configurations within the same directory. If a run with a non-empty name should use cutouts shared across runs, set `shared_cutouts` to *true*.

	Unit	Values	Description
name	–	str/list	Specify a name for your run. Results will be stored under this name. If `scenario: enable:` is set to `true`, the name must contain a subset of scenario names defined in `scenario: file:`. If the name is 'all', all defined scenarios will be run.
prefix	–	str	Prefix for the run name which is used as a top-layer directory name in the results and resources folders.
scenarios			
– enable	bool	{true, false}	Switch to select whether workflow should generate scenarios based on `file`.
– file	str		Path to the scenario yaml file. The scenario file contains config overrides for each scenario. In order to be taken account, `run: scenarios` has to be set to `true` and `run: name` has to be a subset of top level keys given in the scenario file. In order to automatically create a *scenario.yaml* file based on a combination of settings, alter and use the `config/create_scenarios.py` script in the `config` directory.
disable_progressbar	bool	{true, false}	Switch to select whether progressbar should be disabled.
shared_resources	bool/str		Switch to select whether resources should be shared across runs. If a string is passed, this is used as a subdirectory name for shared resources. If set to 'base', only resources before creating the elec.nc file are shared.
shared_cutouts	bool	{true, false}	Switch to select whether cutouts should be shared across runs.

4.2.3 foresight

	Unit	Values	Description
foresight	string	{overnight, myopic, perfect}	See *Foresight Options* for detail explanations.

Note: If you use myopic or perfect foresight, the planning horizon in *The {planning_horizons} wildcard* in scenario has to be set.

4.2.4 scenario

The `scenario` section is an extraordinary section of the config file that is strongly connected to the *Wildcards* and is designed to facilitate running multiple scenarios through a single command

```
# for electricity-only studies
snakemake -call solve_elec_networks

# for sector-coupling studies
snakemake -call solve_sector_networks
```

For each wildcard, a **list of values** is provided. The rule `solve_all_elec_networks` will trigger the rules for creating `results/networks/elec_s{simpl}_{clusters}_ec_l{ll}_{opts}.nc` for **all combinations** of the provided wildcard values as defined by Python's itertools.product(...) function that snakemake's expand(...) function uses.

An exemplary dependency graph (starting from the simplification rules) then looks like this:

	Unit	Values	Description
simpl	–	cf. *The {simpl} wildcard*	List of `{simpl}` wildcards to run.
clusters	–	cf. *The {clusters} wildcard*	List of `{clusters}` wildcards to run.
ll	–	cf. *The {ll} wildcard*	List of `{ll}` wildcards to run.
opts	–	cf. *The {opts} wildcard*	List of `{opts}` wildcards to run.
sector_opts	–	cf. *The {sector_opts} wildcard*	List of `{sector_opts}` wildcards to run.
planning_horizons	–	cf. *The {planning_horizons} wildcard*	List of `{planning_horizon}` wildcards to run.

4.2.5 countries

	Unit	Values	Description
countries	–	Subset of {'AL', 'AT', 'BA', 'BE', 'BG', 'CH', 'CZ', 'DE', 'DK', 'EE', 'ES', 'FI', 'FR', 'GB', 'GR', 'HR', 'HU', 'IE', 'IT', 'LT', 'LU', 'LV', 'ME', 'MK', 'NL', 'NO', 'PL', 'PT', 'RO', 'RS', 'SE', 'SI', 'SK'}	European countries defined by their Two-letter country codes (ISO 3166-1) which should be included in the energy system model.

4.2.6 snapshots

Specifies the temporal range to build an energy system model for as arguments to pandas.date_range

	Unit	Values	Description
start	–	str or datetime-like; e.g. YYYY-MM-DD	Left bound of date range
end	–	str or datetime-like; e.g. YYYY-MM-DD	Right bound of date range
inclusive	–	One of {'neither', 'both', 'left', 'right'}	Make the time interval closed to the `left`, `right`, or both sides `both` or neither side `None`.

4.2.7 enable

Switches for some rules and optional features.

	Unit	Values	Description
enable	str or bool	{auto, true, false}	Switch to include (true) or exclude (false) the retrieve_* rules of snakemake into the workflow; 'auto' sets true|false based on availability of an internet connection to prevent issues with snakemake failing due to lack of internet connection.
prepare_links_p_nom	bool	{true, false}	Switch to retrieve current HVDC projects from Wikipedia
retrieve_databundle	bool	{true, false}	Switch to retrieve databundle from zenodo via the rule *retrieve_databundle* or whether to keep a custom databundle located in the corresponding folder.
retrieve_sector_databundle	bool	{true, false}	Switch to retrieve sector databundle from zenodo via the rule `retrieve_sector_databundle` or whether to keep a custom databundle located in the corresponding folder.
retrieve_cost_data	bool	{true, false}	Switch to retrieve technology cost data from technology-data repository.
build_cutout	bool	{true, false}	Switch to enable the building of cutouts via the rule *build_cutout*.
retrieve_irena	bool	{true, false}	Switch to enable the retrieval of `existing_capacities` from IRENASTAT with *retrieve_irena*.
retrieve_cutout	bool	{true, false}	Switch to enable the retrieval of cutouts from zenodo with `retrieve_cutout`.
build_natura_raster	bool	{true, false}	Switch to enable the creation of the raster `natura.tiff` via the rule *build_natura_raster*.
retrieve_natura_raster	bool	{true, false}	Switch to enable the retrieval of `natura.tiff` from zenodo with `retrieve_natura_raster`.
custom_busmap	bool	{true, false}	Switch to enable the use of custom busmaps in rule *cluster_network*. If activated the rule looks for provided busmaps at `data/custom_busmap_elec_s{simpl}_{clusters}.csv` which should have the same format as `resources/busmap_elec_s{simpl}_{clusters}.csv`, i.e. the index should contain the buses of `networks/elec_s{simpl}.nc`.
drop_leap_day	bool	{true, false}	Switch to drop February 29 from all time-dependent data in leap years

4.2.8 co2 budget

	Unit	Values	Description
co2_budget	–	Dictionary with planning horizons as keys.	CO2 budget as a fraction of 1990 emissions. Overwritten if CO2Lx or cb are set in {sector_opts} wildcard"doc/configtables/othertoplevel.csv

Note: this parameter is over-ridden if CO2Lx or cb is set in sector_opts.

4.2.9 electricity

	Unit	Values	Description
voltages	kV	Any subset of {220., 300., 380.}	Voltage levels to consider
gaslimit_enable	bool	true or false	Add an overall absolute gas limit configured in electricity: gaslimit.
gaslimit	MWhth	float or false	Global gas usage limit
co2limit_enable	bool	true or false	Add an overall absolute carbon-dioxide emissions limit configured in electricity: co2limit.
co2limit	t_{CO_2-eq}	float	Cap on total annual system carbon dioxide emissions
co2base	t_{CO_2-eq}	float	Reference value of total annual system carbon dioxide emissions if relative emission reduction target is specified in {opts} wildcard.
operational_reserve			Settings for reserve requirements following GenX
– activate	bool	true or false	Whether to take operational reserve requirements into account during optimisation
– epsilon_load	–	float	share of total load
– epsilon_vres	–	float	share of total renewable supply
– contingency	MW	float	fixed reserve capacity
max_hours			
– battery	h	float	Maximum state of charge capacity of the battery in terms of hours at full output capacity p_nom. Cf. PyPSA documentation.
– H2	h	float	Maximum state of charge capacity of the hydrogen storage in terms of hours at full output capacity p_nom. Cf. PyPSA documentation.
extendable_carriers			

continues on next page

Table 1 – continued from previous page

	Unit	Values	Description
– Generator	–	Any extendable carrier	Defines existing or non-existing conventional and renewable power plants to be extendable during the optimization. Conventional generators can only be built/expanded where already existent today. If a listed conventional carrier is not included in the `conventional_carriers` list, the lower limit of the capacity expansion is set to 0.
– StorageUnit	–	Any subset of {'battery','H2'}	Adds extendable storage units (battery and/or hydrogen) at every node/bus after clustering without capacity limits and with zero initial capacity.
– Store	–	Any subset of {'battery','H2'}	Adds extendable storage units (battery and/or hydrogen) at every node/bus after clustering without capacity limits and with zero initial capacity.
– Link	–	Any subset of {'H2 pipeline'}	Adds extendable links (H2 pipelines only) at every connection where there are lines or HVDC links without capacity limits and with zero initial capacity. Hydrogen pipelines require hydrogen storage to be modelled as `Store`.
powerplants_filter	–	use pandas.query strings here, e.g. `Country not in ['Germany']`	Filter query for the default powerplant database.
custom_powerplants	–	use pandas.query strings here, e.g. `Country in ['Germany']`	Filter query for the custom powerplant database.
everywhere_powerplants	–	Any subset of {nuclear, oil, OCGT, CCGT, coal, lignite, geothermal, biomass}	List of conventional power plants to add to every node in the model with zero initial capacity. To be used in combination with `extendable_carriers` to allow for building conventional powerplants irrespective of existing locations.
conventional_carriers	–	Any subset of {nuclear, oil, OCGT, CCGT, coal, lignite, geothermal, biomass}	List of conventional power plants to include in the model from `resources/powerplants.csv`. If an included carrier is also listed in `extendable_carriers`, the capacity is taken as a lower bound.
renewable_carriers	–	Any subset of {solar, onwind, offwind-ac, offwind-dc, hydro}	List of renewable generators to include in the model.
estimate_renewable_capacities			

continues on next page

Table 1 – continued from previous page

	Unit	Values	Description
– enable		bool	Activate routine to estimate renewable capacities in rule *add_electricity*. This option should not be used in combination with pathway planning `foresight: myopic` or `foresight: perfect` as renewable capacities are added differently in *add_existing_baseyear*.
– from_opsd	–	bool	Add renewable capacities from OPSD database. The value is depreciated but still can be used.
– year	–	bool	Renewable capacities are based on existing capacities reported by IRENA (IRENASTAT) for the specified year
– expansion_limit	–	float or false	Artificially limit maximum IRENA capacities to a factor. For example, an `expansion_limit: 1.1` means 110% of capacities . If false are chosen, the estimated renewable potentials determine by the workflow are used.
– technology_mapping			Mapping between PyPSA-Eur and powerplantmatching technology names
– – Offshore	–	Any subset of {offwind-ac, offwind-dc}	List of PyPSA-Eur carriers that is considered as (IRENA, OPSD) onshore technology.
– – Offshore	–	{onwind}	List of PyPSA-Eur carriers that is considered as (IRENA, OPSD) offshore technology.
– – PV	–	{solar}	List of PyPSA-Eur carriers that is considered as (IRENA, OPSD) PV technology.
autarky			
– enable	bool	true or false	Require each node to be autarkic by removing all lines and links.
– by_country	bool	true or false	Require each country to be autarkic by removing all cross-border lines and links. `electricity: autarky` must be enabled.

4.2.10 `atlite`

Define and specify the `atlite.Cutout` used for calculating renewable potentials and time-series. All options except for `features` are directly used as cutout parameters.

	Unit	Values	Description
default_cutout	–	str	Defines a default cutout.
nprocesses	–	int	Number of parallel processes in cutout preparation
show_progress	bool	true/false	Whether progressbar for atlite conversion processes should be shown. False saves time.
cutouts			
– {name}	–	Convention is to name cutouts like `<region>-<year>-<sourc` (e.g. `europe-2013-era5`).	Name of the cutout netcdf file. The user may specify multiple cutouts under configuration `atlite: cutouts:`. Reference is used in configuration `renewable: {technology}: cutout:`. The cutout `base` may be used to automatically calculate temporal and spatial bounds of the network.
– – module	–	Subset of {'era5','sarah'}	Source of the reanalysis weather dataset (e.g. ERA5 or SARAH-2)
– – x	°	Float interval within [-180, 180]	Range of longitudes to download weather data for. If not defined, it defaults to the spatial bounds of all bus shapes.
– – y	°	Float interval within [-90, 90]	Range of latitudes to download weather data for. If not defined, it defaults to the spatial bounds of all bus shapes.
– – dx	°	Larger than 0.25	Grid resolution for longitude
– – dy	°	Larger than 0.25	Grid resolution for latitude
– – time		Time interval within ['1979', '2018'] (with valid pandas date time strings)	Time span to download weather data for. If not defined, it defaults to the time interval spanned by the snapshots.
– – features		String or list of strings with valid cutout features ('influx', 'wind').	When freshly building a cutout, retrieve data only for those features. If not defined, it defaults to all available features.

4.2.11 `renewable`

onwind

	Unit	Values	Description
cutout	–	Should be a folder listed in the configuration `atlite: cutouts:` (e.g. 'europe-2013-era5') or reference an existing folder in the directory `cutouts`. Source module must be ERA5.	Specifies the directory where the relevant weather data ist stored.
resource			
– method	–	Must be 'wind'	A superordinate technology type.
– turbine	–	One of turbine types included in atlite. Can be a string or a dictionary with years as keys which denote the year another turbine model becomes available.	Specifies the turbine type and its characteristic power curve.
capacity_per_sqkm	MW/km	float	Allowable density of wind turbine placement.
corine			
– grid_codes	–	Any subset of the CORINE Land Cover code list	Specifies areas according to CORINE Land Cover codes which are generally eligible for wind turbine placement.
– distance	m	float	Distance to keep from areas specified in `distance_grid_codes`
– distance_grid_codes	–	Any subset of the CORINE Land Cover code list	Specifies areas according to CORINE Land Cover codes to which wind turbines must maintain a distance specified in the setting `distance`.
luisa			
– grid_codes	–	Any subset of the LUISA Base Map codes in Annex 1	Specifies areas according to the LUISA Base Map codes which are generally eligible for wind turbine placement.
– distance	m	float	Distance to keep from areas specified in `distance_grid_codes`
– distance_grid_codes	–	Any subset of the LUISA Base Map codes in Annex 1	Specifies areas according to the LUISA Base Map codes to which wind turbines must maintain a distance specified in the setting `distance`.
natura	bool	{true, false}	Switch to exclude Natura 2000 natural protection areas. Area is excluded if `true`.
clip_p_max_pu	p.u.	float	To avoid too small values in the renewables` per-unit availability time series values below this threshold are set to zero.
correction_factor	–	float	Correction factor for capacity factor time series.
excluder_resolution	m	float	Resolution on which to perform geographical elibility analysis.

Note: Notes on `capacity_per_sqkm`. ScholzPhd Tab 4.3.1: 10MW/km^2 and assuming 30% fraction of the already restricted area is available for installation of wind generators due to competing land use and likely public acceptance issues.

Note: The default choice for corine `grid_codes` was based on Scholz, Y. (2012). Renewable energy based electricity supply at low costs development of the REMix model and application for Europe. (p.42 / p.28)

`offwind-ac`

	Unit	Values	Description
cutout	–	Should be a folder listed in the configuration `atlite: cutouts:` (e.g. 'europe-2013-era5') or reference an existing folder in the directory `cutouts`. Source module must be ERA5.	Specifies the directory where the relevant weather data ist stored.
resource			
– method	–	Must be 'wind'	A superordinate technology type.
– turbine	–	One of turbine types included in atlite. Can be a string or a dictionary with years as keys which denote the year another turbine model becomes available.	Specifies the turbine type and its characteristic power curve.
capacity_per_sqkm	MW/km	float	Allowable density of wind turbine placement.
correction_factor	–	float	Correction factor for capacity factor time series.
excluder_resolution	m	float	Resolution on which to perform geographical elibility analysis.
corine	–	Any *realistic* subset of the CORINE Land Cover code list	Specifies areas according to CORINE Land Cover codes which are generally eligible for AC-connected offshore wind turbine placement.
luisa	–	Any subset of the LUISA Base Map codes in Annex 1	Specifies areas according to the LUISA Base Map codes which are generally eligible for AC-connected offshore wind turbine placement.
natura	bool	{true, false}	Switch to exclude Natura 2000 natural protection areas. Area is excluded if `true`.
ship_threshold	–	float	Ship density threshold from which areas are excluded.
max_depth	m	float	Maximum sea water depth at which wind turbines can be build. Maritime areas with deeper waters are excluded in the process of calculating the AC-connected offshore wind potential.
min_shore_distance	m	float	Minimum distance to the shore below which wind turbines cannot be build. Such areas close to the shore are excluded in the process of calculating the AC-connected offshore wind potential.
max_shore_distance	m	float	Maximum distance to the shore above which wind turbines cannot be build. Such areas close to the shore are excluded in the process of calculating the AC-connected offshore wind potential.
clip_p_max_pu	p.u.	float	To avoid too small values in the renewables` per-unit availability time series values below this threshold are set to zero.

Note: Notes on `capacity_per_sqkm`. ScholzPhd Tab 4.3.1: 10MW/km^2 and assuming 20% fraction of the already restricted area is available for installation of wind generators due to competing land use and likely public acceptance issues.

Note: Notes on `correction_factor`. Correction due to proxy for wake losses from 10.1016/j.energy.2018.08.153 until done more rigorously in #153

`offwind-dc`

	Unit	Values	Description
cutout	–	Should be a folder listed in the configuration `atlite: cutouts:` (e.g. 'europe-2013-era5') or reference an existing folder in the directory `cutouts`. Source module must be ERA5.	Specifies the directory where the relevant weather data ist stored.
resource			
– method	–	Must be 'wind'	A superordinate technology type.
– turbine	–	One of turbine types included in atlite. Can be a string or a dictionary with years as keys which denote the year another turbine model becomes available.	Specifies the turbine type and its characteristic power curve.
capacity_per_sqkm	MW/km	float	Allowable density of wind turbine placement.
correction_factor	–	float	Correction factor for capacity factor time series.
excluder_resolution	m	float	Resolution on which to perform geographical elibility analysis.
corine	–	Any *realistic* subset of the CORINE Land Cover code list	Specifies areas according to CORINE Land Cover codes which are generally eligible for AC-connected offshore wind turbine placement.
luisa	–	Any subset of the LUISA Base Map codes in Annex 1	Specifies areas according to the LUISA Base Map codes which are generally eligible for DC-connected offshore wind turbine placement.
natura	bool	{true, false}	Switch to exclude Natura 2000 natural protection areas. Area is excluded if `true`.
ship_threshold	–	float	Ship density threshold from which areas are excluded.
max_depth	m	float	Maximum sea water depth at which wind turbines can be build. Maritime areas with deeper waters are excluded in the process of calculating the AC-connected offshore wind potential.
min_shore_distance	m	float	Minimum distance to the shore below which wind turbines cannot be build.
max_shore_distance	m	float	Maximum distance to the shore above which wind turbines cannot be build.
clip_p_max_pu	p.u.	float	To avoid too small values in the renewables` per-unit availability time series values below this threshold are set to zero.

Note: both `offwind-ac` and `offwind-dc` have the same assumption on `capacity_per_sqkm` and

correction_factor.

solar

	Unit	Values	Description
cutout	–	Should be a folder listed in the configuration `atlite: cutouts:` (e.g. 'europe-2013-era5') or reference an existing folder in the directory `cutouts`. Source module can be ERA5 or SARAH-2.	Specifies the directory where the relevant weather data ist stored that is specified at `atlite/cutouts` configuration. Both `sarah` and `era5` work.
resource			
– method	–	Must be 'pv'	A superordinate technology type.
– panel	–	One of {'Csi', 'CdTe', 'KANENA'} as defined in atlite. Can be a string or a dictionary with years as keys which denote the year another turbine model becomes available.	Specifies the solar panel technology and its characteristic attributes.
– orientation			
– – slope	°	Realistically any angle in [0., 90.]	Specifies the tilt angle (or slope) of the solar panel. A slope of zero corresponds to the face of the panel aiming directly overhead. A positive tilt angle steers the panel towards the equator.
– – azimuth	°	Any angle in [0., 360.]	Specifies the azimuth orientation of the solar panel. South corresponds to 180.°.
capacity_per_sqkm	MW/km	float	Allowable density of solar panel placement.
correction_factor	–	float	A correction factor for the capacity factor (availability) time series.
corine	–	Any subset of the CORINE Land Cover code list	Specifies areas according to CORINE Land Cover codes which are generally eligible for solar panel placement.
luisa	–	Any subset of the LUISA Base Map codes in Annex 1	Specifies areas according to the LUISA Base Map codes which are generally eligible for solar panel placement.
natura	bool	{true, false}	Switch to exclude Natura 2000 natural protection areas. Area is excluded if `true`.
clip_p_max_pu	p.u.	float	To avoid too small values in the renewables` per-unit availability time series values below this threshold are set to zero.
excluder_resolution	m	float	Resolution on which to perform geographical elibility analysis.

Note: Notes on `capacity_per_sqkm`. ScholzPhd Tab 4.3.1: 170 MW/km^2 and assuming 1% of the area can be used for solar PV panels. Correction factor determined by comparing uncorrected area-weighted full-load hours

to those published in Supplementary Data to Pietzcker, Robert Carl, et al. "Using the sun to decarbonize the power sector – The economic potential of photovoltaics and concentrating solar power." Applied Energy 135 (2014): 704-720. This correction factor of 0.854337 may be in order if using reanalysis data. for discussion refer to this <issue https://github.com/PyPSA/pypsa-eur/issues/285>

hydro

	Unit	Values	Description
cutout	–	Must be 'europe-2013-era5'	Specifies the directory where the relevant weather data ist stored.
carriers	–	Any subset of {'ror', 'PHS', 'hydro'}	Specifies the types of hydro power plants to build per-unit availability time series for. 'ror' stands for run-of-river plants, 'PHS' represents pumped-hydro storage, and 'hydro' stands for hydroelectric dams.
PHS_max_hours	h	float	Maximum state of charge capacity of the pumped-hydro storage (PHS) in terms of hours at full output capacity `p_nom`. Cf. PyPSA documentation.
hydro_max_hours	h	Any of {float, 'energy_capacity_totals_by_cou 'estimate_by_large_installations'	Maximum state of charge capacity of the pumped-hydro storage (PHS) in terms of hours at full output capacity `p_nom` or heuristically determined. Cf. PyPSA documentation.
flatten_dispatch	bool	{true, false}	Consider an upper limit for the hydro dispatch. The limit is given by the average capacity factor plus the buffer given in `flatten_dispatch_buffer`
flatten_dispatch_buffer	–	float	If `flatten_dispatch` is true, specify the value added above the average capacity factor.
clip_min_inflow	MW	float	To avoid too small values in the inflow time series, values below this threshold are set to zero.
eia_norm_year	–	Year in EIA hydro generation dataset; or False to disable	To specify a specific year by which hydro inflow is normed that deviates from the snapshots' year
eia_correct_by_capacity	–	boolean	Correct EIA annual hydro generation data by installed capacity.
eia_approximate_missing	–	boolean	Approximate hydro generation data for years not included in EIA dataset through a regression based on annual runoff.

4.2.12 conventional

Define additional generator attribute for conventional carrier types. If a scalar value is given it is applied to all generators. However if a string starting with "data/" is given, the value is interpreted as a path to a csv file with country specific values. Then, the values are read in and applied to all generators of the given carrier in the given country. Note that the value(s) overwrite the existing values.

	Unit	Values	Description
unit_commitment	bool	{true, false}	Allow the overwrite of ramp_limit_up, ramp_limit_start_up, ramp_limit_shut_down, p_min_pu, min_up_time, min_down_time, and start_up_cost of conventional generators. Refer to the CSV file „unit_commitment.csv“.
dynamic_fuel_price	bool	{true, false}	Consider the monthly fluctuating fuel prices for each conventional generator. Refer to the CSV file "data/validation/monthly_fuel_price.csv".
{name}	–	string	For any carrier/technology overwrite attributes as listed below.
– {attribute}	–	string or float	For any attribute, can specify a float or reference to a file path to a CSV file giving floats for each country (2-letter code).

4.2.13 `lines`

	Unit	Values	Description
types	–	Values should specify a line type in PyPSA. Keys should specify the corresponding voltage level (e.g. 220., 300. and 380. kV)	Specifies line types to assume for the different voltage levels of the ENTSO-E grid extraction. Should normally handle voltage levels 220, 300, and 380 kV
s_max_pu	–	Value in [0.,1.]	Correction factor for line capacities (`s_nom`) to approximate $N-1$ security and reserve capacity for reactive power flows
s_nom_max	MW	float	Global upper limit for the maximum capacity of each extendable line.
max_extension	MW	float	Upper limit for the extended capacity of each extendable line.
length_factor	–	float	Correction factor to account for the fact that buses are *not* connected by lines through air-line distance.
under_construction	–	One of {'zero': set capacity to zero, 'remove': remove completely, 'keep': keep with full capacity}	Specifies how to handle lines which are currently under construction.
reconnect_crimea	–	true or false	Whether to reconnect Crimea to the Ukrainian grid
dynamic_line_rating			
– activate	bool	true or false	Whether to take dynamic line rating into account
– cutout	–	Should be a folder listed in the configuration `atlite: cutouts:` (e.g. 'europe-2013-era5') or reference an existing folder in the directory `cutouts`. Source module must be ERA5.	Specifies the directory where the relevant weather data ist stored.
– correction_factor	–	float	Factor to compensate for overestimation of wind speeds in hourly averaged wind data
– max_voltage_difference	deg	float	Maximum voltage angle difference in degrees or 'false' to disable
– max_line_rating	–	float	Maximum line rating relative to nominal capacity without DLR, e.g. 1.3 or 'false' to disable

4.2.14 `links`

	Unit	Values	Description
p_max_pu	–	Value in [0.,1.]	Correction factor for link capacities `p_nom`.
p_nom_max	MW	float	Global upper limit for the maximum capacity of each extendable DC link.
max_extension	MW	float	Upper limit for the extended capacity of each extendable DC link.
include_tyndp	bool	{'true', 'false'}	Specifies whether to add HVDC link projects from the TYNDP 2018 which are at least in permitting.
file_tyndp	str		Path to the links tyndp file.
file_parameter_corrections	str		Path to the parameter correction file of `data/entsoegridkit`.
under_construction	–	One of {'zero': set capacity to zero, 'remove': remove completely, 'keep': keep with full capacity}	Specifies how to handle lines which are currently under construction.

4.2.15 `transformers`

	Unit	Values	Description
x	p.u.	float	Series reactance (per unit, using `s_nom` as base power of the transformer. Overwritten if `type` is specified.
s_nom	MVA	float	Limit of apparent power which can pass through branch. Overwritten if `type` is specified.
type	–	A transformer type in PyPSA.	Specifies transformer types to assume for the transformers of the ENTSO-E grid extraction.

4.2.16 load

	Unit	Values	Description
interpolate_limit	hours	integer	Maximum gap size (consecutive nans) which interpolated linearly.
time_shift_for_large_gaps	string	string	Periods which are used for copying time-slices in order to fill large gaps of nans. Have to be valid `pandas` period strings.
manual_adjustments	bool	{true, false}	Whether to adjust the load data manually according to the function in `manual_adjustment()`.
scaling_factor	–	float	Global correction factor for the load time series.
fixed_year	–	Year or False	To specify a fixed year for the load time series that deviates from the snapshots' year
supplement_synthetic	bool	{true, false}	Whether to supplement missing data for selected time period should be supplemented by synthetic data from https://zenodo.org/record/10820928.

4.2.17 energy

Note: Only used for sector-coupling studies.

	Unit	Values	Description
energy_totals_year	–	{1990,1995,2000,2005,2010,	The year for the sector energy use. The year must be avaliable in the Eurostat report
base_emissions_year	–	YYYY; e.g. 1990	The base year for the sector emissions. See European Environment Agency (EEA).
emissions	–	{CO2, All greenhouse gases - (CO2 equivalent)}	Specify which sectoral emissions are taken into account. Data derived from EEA. Currently only CO2 is implemented.

4.2.18 biomass

Note: Only used for sector-coupling studies.

	Unit	Values	Description
year	–	{2010, 2020, 2030, 2040, 2050}	Year for which to retrieve biomass potential according to the assumptions of the JRC ENSPRESO .
scenario	–	{"ENS_Low", "ENS_Med", "ENS_High"}	Scenario for which to retrieve biomass potential. The scenario definition can be seen in ENSPRESO_BIOMASS
classes			
– solid biomass	–	Array of biomass comodity	The comodity that are included as solid biomass
– not included	–	Array of biomass comodity	The comodity that are not included as a biomass potential
– biogas	–	Array of biomass comodity	The comodity that are included as biogas

The list of available biomass is given by the category in ENSPRESO_BIOMASS, namely:

- Agricultural waste
- Manure solid, liquid
- Residues from landscape care
- Bioethanol barley, wheat, grain maize, oats, other cereals and rye
- Sugar from sugar beet
- Miscanthus, switchgrass, RCG
- Willow
- Poplar
- Sunflower, soya seed
- Rape seed
- Fuelwood residues
- FuelwoodRW
- C&P_RW
- Secondary Forestry residues - woodchips
- Sawdust
- Municipal waste
- Sludge

4.2.19 solar_thermal

Note: Only used for sector-coupling studies.

	Unit	Values	Description
clearsky_model	–	{'simple', 'enhanced'}	Type of clearsky model for diffuse irradiation
orientation	–	{units of degrees, 'latitude_optimal'}	Panel orientation with slope and azimuth
– azimuth	float	units of degrees	The angle between the North and the sun with panels on the local horizon
– slope	float	units of degrees	The angle between the ground and the panels

4.2.20 existing_capacities

Note: Only used for sector-coupling studies. The value for grouping years are only used in myopic or perfect foresight scenarios.

	Unit	Values	Description
grouping_years_power	–	A list of years	Intervals to group existing capacities for power
grouping_years_heat	–	A list of years below 2020	Intervals to group existing capacities for heat
threshold_capacity	MW	float	Capacities generators and links of below threshold are removed during add_existing_capacities
default_heating_lifetime	years	int	Default lifetime for heating technologies
conventional_carriers	–	Any subset of {uranium, coal, lignite, oil}	List of conventional power plants to include in the sectoral network

4.2.21 sector

Note: Only used for sector-coupling studies.

	Unit	Values	Description
transport	–	{true, false}	Flag to include transport sector.
heating	–	{true, false}	Flag to include heating sector.
biomass	–	{true, false}	Flag to include biomass sector.

continues on next page

Table 2 – continued from previous page

	Unit	Values	Description
industry	–	{true, false}	Flag to include industry sector.
agriculture	–	{true, false}	Flag to include agriculture sector.
district_heating	–		prepare_sector_network.py
– potential	–	float	maximum fraction of urban demand which can be supplied by district heating
– progress	–	Dictionary with planning horizons as keys.	Increase of today's district heating demand to potential maximum district heating share. Progress = 0 means today's district heating share. Progress = 1 means maximum fraction of urban demand is supplied by district heating
– district_heating_loss	–	float	Share increase in district heat demand in urban central due to heat losses
cluster_heat_buses	–	{true, false}	Cluster residential and service heat buses in prepare_sector_network.py to one to save memory.
bev_dsm_restriction _value	–	float	Adds a lower state of charge (SOC) limit for battery electric vehicles (BEV) to manage its own energy demand (DSM). Located in build_transport_demand.py. Set to 0 for no restriction on BEV DSM
bev_dsm_restriction _time	–	float	Time at which SOC of BEV has to be dsm_restriction_value
transport_heating _deadband_upper	°C	float	The maximum temperature in the vehicle. At higher temperatures, the energy required for cooling in the vehicle increases.
transport_heating _deadband_lower	°C	float	The minimum temperature in the vehicle. At lower temperatures, the energy required for heating in the vehicle increases.
ICE_lower_degree_factor	–	float	Share increase in energy demand in internal combustion engine (ICE) for each degree difference between the cold environment and the minimum temperature.
ICE_upper_degree_factor	–	float	Share increase in energy demand in internal combustion engine (ICE) for each degree difference between the hot environment and the maximum temperature.
EV_lower_degree_factor	–	float	Share increase in energy demand in electric vehicles (EV) for each degree difference between the cold environment and the minimum temperature.
EV_upper_degree_factor	–	float	Share increase in energy demand in electric vehicles (EV) for each degree difference between the hot environment and the maximum temperature.
bev_dsm	–	{true, false}	Add the option for battery electric vehicles (BEV) to participate in demand-side management (DSM)

continues on next page

Table 2 – continued from previous page

	Unit	Values	Description
bev_availability	–	float	The share for battery electric vehicles (BEV) that are able to do demand side management (DSM)
bev_energy	–	float	The average size of battery electric vehicles (BEV) in MWh
bev_charge_efficiency	–	float	Battery electric vehicles (BEV) charge and discharge efficiency
bev_plug_to_wheel _efficiency	km/kWh	float	The distance battery electric vehicles (BEV) can travel in km per kWh of energy charge in battery. Base value comes from Tesla Model S
bev_charge_rate	MWh	float	The power consumption for one electric vehicle (EV) in MWh. Value derived from 3-phase charger with 11 kW.
bev_avail_max	–	float	The maximum share plugged-in availability for passenger electric vehicles.
bev_avail_mean	–	float	The average share plugged-in availability for passenger electric vehicles.
v2g	–	{true, false}	Allows feed-in to grid from EV battery
land_transport_fuel_cell _share	–	Dictionary with planning horizons as keys.	The share of vehicles that uses fuel cells in a given year
land_transport_electric _share	–	Dictionary with planning horizons as keys.	The share of vehicles that uses electric vehicles (EV) in a given year
land_transport_ice _share	–	Dictionary with planning horizons as keys.	The share of vehicles that uses internal combustion engines (ICE) in a given year. What is not EV or FCEV is oil-fuelled ICE.
transport_fuel_cell _efficiency	–	float	The H2 conversion efficiencies of fuel cells in transport
transport_internal _combustion_efficiency	–	float	The oil conversion efficiencies of internal combustion engine (ICE) in transport
agriculture_machinery _electric_share	–	float	The share for agricultural machinery that uses electricity
agriculture_machinery _oil_share	–	float	The share for agricultural machinery that uses oil
agriculture_machinery _fuel_efficiency	–	float	The efficiency of electric-powered machinery in the conversion of electricity to meet agricultural needs.
agriculture_machinery _electric_efficiency	–	float	The efficiency of oil-powered machinery in the conversion of oil to meet agricultural needs.
Mwh_MeOH_per_MWh_H2	LHV	float	The energy amount of the produced methanol per energy amount of hydrogen. From DECHEMA (2017), page 64.
MWh_MeOH_per_tCO2	LHV	float	The energy amount of the produced methanol per ton of CO2. From DECHEMA (2017), page 66.
MWh_MeOH_per_MWh_e	LHV	float	The energy amount of the produced methanol per energy amount of electricity. From DECHEMA (2017), page 64.

continues on next page

Table 2 – continued from previous page

	Unit	Values	Description
shipping_hydrogen _liquefaction	–	{true, false}	Whether to include liquefaction costs for hydrogen demand in shipping.
shipping_hydrogen_share	–	Dictionary with planning horizons as keys.	The share of ships powered by hydrogen in a given year
shipping_methanol_share	–	Dictionary with planning horizons as keys.	The share of ships powered by methanol in a given year
shipping_oil_share	–	Dictionary with planning horizons as keys.	The share of ships powered by oil in a given year
shipping_methanol _efficiency	–	float	The efficiency of methanol-powered ships in the conversion of methanol to meet shipping needs (propulsion). The efficiency increase from oil can be 10-15% higher according to the IEA
shipping_oil_efficiency	–	float	The efficiency of oil-powered ships in the conversion of oil to meet shipping needs (propulsion). Base value derived from 2011
aviation_demand_factor	–	float	The proportion of demand for aviation compared to today's consumption
HVC_demand_factor	–	float	The proportion of demand for high-value chemicals compared to today's consumption
time_dep_hp_cop	–	{true, false}	Consider the time dependent coefficient of performance (COP) of the heat pump
heat_pump_sink_T	°C	float	The temperature heat sink used in heat pumps based on DTU / large area radiators. The value is conservatively high to cover hot water and space heating in poorly-insulated buildings
reduce_space_heat _exogenously	–	{true, false}	Influence on space heating demand by a certain factor (applied before losses in district heating).
reduce_space_heat _exogenously_factor	–	Dictionary with planning horizons as keys.	A positive factor can mean renovation or demolition of a building. If the factor is negative, it can mean an increase in floor area, increased thermal comfort, population growth. The default factors are determined by the Eurocalc Homes and buildings decarbonization scenario
retrofitting			
– retro_endogen	–	{true, false}	Add retrofitting as an endogenous system which co-optimise space heat savings.
– cost_factor	–	float	Weight costs for building renovation
– interest_rate	–	float	The interest rate for investment in building components
– annualise_cost	–	{true, false}	Annualise the investment costs of retrofitting

continues on next page

Table 2 – continued from previous page

	Unit	Values	Description
– tax_weighting	–	{true, false}	Weight the costs of retrofitting depending on taxes in countries
– construction_index	–	{true, false}	Weight the costs of retrofitting depending on labour/material costs per country
tes			
– central	–	{true, false}	Add option for storing thermal energy in large water pits associated with district heating systems (TES)
– decentral	–	{true, false}	Add option for storing thermal energy in large water pits associated with individual thermal energy storage (TES)
tes_tau			The time constant used to calculate the decay of thermal energy in thermal energy storage (TES): 1- $e^{-1/24t}$.
– decentral	days	float	The time constant in decentralized thermal energy storage (TES)
– central	days	float	The time constant in centralized thermal energy storage (TES)
boilers	–	{true, false}	Add option for transforming gas into heat using gas boilers
resistive_heaters	–	{true, false}	Add option for transforming electricity into heat using resistive heaters (independently from gas boilers)
oil_boilers	–	{true, false}	Add option for transforming oil into heat using boilers
biomass_boiler	–	{true, false}	Add option for transforming biomass into heat using boilers
overdimension_individual_heating	–	float	Add option for overdimensioning individual heating systems by a certain factor. This allows them to cover heat demand peaks e.g. 10% higher than those in the data with a setting of 1.1.
chp	–	{true, false}	Add option for using Combined Heat and Power (CHP)
micro_chp	–	{true, false}	Add option for using Combined Heat and Power (CHP) for decentral areas.
solar_thermal	–	{true, false}	Add option for using solar thermal to generate heat.
solar_cf_correction	–	float	The correction factor for the value provided by the solar thermal profile calculations
marginal_cost_storage	currency/M	float	The marginal cost of discharging batteries in distributed grids
methanation	–	{true, false}	Add option for transforming hydrogen and CO2 into methane using methanation.
coal_cc	–	{true, false}	Add option for coal CHPs with carbon capture
dac	–	{true, false}	Add option for Direct Air Capture (DAC)
co2_vent	–	{true, false}	Add option for vent out CO2 from storages to the atmosphere.

continues on next page

Table 2 – continued from previous page

	Unit	Values	Description
allam_cycle	–	{true, false}	Add option to include Allam cycle gas power plants
hydrogen_fuel_cell	–	{true, false}	Add option to include hydrogen fuel cell for re-electrification. Assuming OCGT technology costs
hydrogen_turbine	–	{true, false}	Add option to include hydrogen turbine for re-electrification. Add both OCGT and CCGT technology costs.
CC_turbine	–	{true, false}	Add option to include carbon capture turbine for re-electrification. Add both OCGT and CCGT technology costs.
SMR	–	{true, false}	Add option for transforming natural gas into hydrogen and CO2 using Steam Methane Reforming (SMR)
SMR CC	–	{true, false}	Add option for transforming natural gas into hydrogen and CO2 using Steam Methane Reforming (SMR) and Carbon Capture (CC)
regional_methanol_demand	–	{true, false}	Spatially resolve methanol demand. Set to true if regional CO2 constraints needed.
regional_oil_demand	–	{true, false}	Spatially resolve oil demand. Set to true if regional CO2 constraints needed.
regional_co2 _sequestration_potential			
– enable	–	{true, false}	Add option for regionally-resolved geological carbon dioxide sequestration potentials based on CO2StoP.
– attribute	–	string or list	Name (or list of names) of the attribute(s) for the sequestration potential
– include_onshore	–	{true, false}	Add options for including onshore sequestration potentials
– min_size	Gt	float	Any sites with lower potential than this value will be excluded
– max_size	Gt	float	The maximum sequestration potential for any one site.
– years_of_storage	years	float	The years until potential exhausted at optimised annual rate
co2_sequestration_potential	MtCO2/	float	The potential of sequestering CO2 in Europe per year
co2_sequestration_cost	currency/tC	float	The cost of sequestering a ton of CO2
co2_sequestration_lifetime	years	int	The lifetime of a CO2 sequestration site
co2_spatial	–	{true, false}	Add option to spatially resolve carrier representing stored carbon dioxide. This allows for more detailed modelling of CCUTS, e.g. regarding the capturing of industrial process emissions, usage as feedstock for electrofuels, transport of carbon dioxide, and geological sequestration sites.

continues on next page

Table 2 – continued from previous page

	Unit	Values	Description
co2network	–	{true, false}	Add option for planning a new carbon dioxide transmission network
co2_network_cost_factor	p.u.	float	The cost factor for the capital cost of the carbon dioxide transmission network
cc_fraction	–	float	The default fraction of CO2 captured with post-combustion capture
hydrogen_underground _storage	–	{true, false}	Add options for storing hydrogen underground. Storage potential depends regionally.
hydrogen_underground _storage_locations		{onshore, nearshore, offshore}	The location where hydrogen underground storage can be located. Onshore, nearshore, offshore means it must be located more than 50 km away from the sea, within 50 km of the sea, or within the sea itself respectively.
ammonia	–	{true, false, regional}	Add ammonia as a carrrier. It can be either true (copperplated NH3), false (no NH3 carrier) or "regional" (regionalised NH3 without network)
min_part_load_fischer _tropsch	per unit of p_nom	float	The minimum unit dispatch (`p_min_pu`) for the Fischer-Tropsch process
min_part_load _methanolisation	per unit of p_nom	float	The minimum unit dispatch (`p_min_pu`) for the methanolisation process
use_fischer_tropsch _waste_heat	–	{true, false}	Add option for using waste heat of Fischer Tropsch in district heating networks
use_fuel_cell_waste_heat	–	{true, false}	Add option for using waste heat of fuel cells in district heating networks
use_electrolysis_waste _heat	–	{true, false}	Add option for using waste heat of electrolysis in district heating networks
electricity_transmission _grid	–	{true, false}	Switch for enabling/disabling the electricity transmission grid.
electricity_distribution _grid	–	{true, false}	Add a simplified representation of the exchange capacity between transmission and distribution grid level through a link.
electricity_distribution _grid_cost_factor			Multiplies the investment cost of the electricity distribution grid
electricity_grid _connection	–	{true, false}	Add the cost of electricity grid connection for onshore wind and solar
transmission_efficiency			Section to specify transmission losses or compression energy demands of bidirectional links. Splits them into two capacity-linked unidirectional links.
– {carrier}	–	str	The carrier of the link.
– – efficiency_static	p.u.	float	Length-independent transmission efficiency.

continues on next page

Table 2 – continued from previous page

	Unit	Values	Description
– – efficiency_per_1000km	p.u. per 1000 km	float	Length-dependent transmission efficiency ($eta^{text{length}}$)
– – compression_per_1000km	p.u. per 1000 km	float	Length-dependent electricity demand for compression ($eta cdot text{length}$) implemented as multi-link to local electricity bus.
H2_network	–	{true, false}	Add option for new hydrogen pipelines
gas_network	–	{true, false}	Add existing natural gas infrastructure, incl. LNG terminals, production and entry-points. The existing gas network is added with a lossless transport model. A length-weighted k-edge augmentation algorithm can be run to add new candidate gas pipelines such that all regions of the model can be connected to the gas network. When activated, all the gas demands are regionally disaggregated as well.
H2_retrofit	–	{true, false}	Add option for retrofiting existing pipelines to transport hydrogen.
H2_retrofit_capacity_per_CH4	–	float	The ratio for H2 capacity per original CH4 capacity of retrofitted pipelines. The European Hydrogen Backbone (April, 2020) p.15 60% of original natural gas capacity could be used in cost-optimal case as H2 capacity.
H2_import	–	{true, false}	Add option to import H2 from outside of EU
gas_network_connectivity_upgrade	–	float	The number of desired edge connectivity (k) in the length-weighted k-edge augmentation algorithm used for the gas network
gas_distribution_grid	–	{true, false}	Add a gas distribution grid
gas_distribution_grid_cost_factor			Multiplier for the investment cost of the gas distribution grid
biomass_spatial	–	{true, false}	Add option for resolving biomass demand regionally
biomass_transport	–	{true, false}	Add option for transporting solid biomass between nodes
biogas_upgrading_cc	–	{true, false}	Add option to capture CO2 from biomass upgrading
conventional_generation			Add a more detailed description of conventional carriers. Any power generation requires the consumption of fuel from nodes representing that fuel.
biomass_to_liquid	–	{true, false}	Add option for transforming solid biomass into liquid fuel with the same properties as oil

continues on next page

Table 2 – continued from previous page

	Unit	Values	Description
biosng	–	{true, false}	Add option for transforming solid biomass into synthesis gas with the same properties as natural gas
limit_max_growth			
– enable	–	{true, false}	Add option to limit the maximum growth of a carrier
– factor	p.u.	float	The maximum growth factor of a carrier (e.g. 1.3 allows 30% larger than max historic growth)
– max_growth			
– – {carrier}	GW	float	The historic maximum growth of a carrier
– max_relative_growth			
– – {carrier}	p.u.	float	The historic maximum relative growth of a carrier

4.2.22 industry

Note: Only used for sector-coupling studies.

4.2.23 costs

	Unit	Values	Description
year	–	YYYY; e.g. '2030'	Year for which to retrieve cost assumptions of resources/costs.csv.
version	–	vX.X.X or <user>/<repo>/vX.X.X; e.g. 'v0.5.0'	Version of technology-data repository to use. If this string is of the form <user>/<repo>/<version> then costs are instead retrieved from github.com/<user>/<repo> at the <version> tag.
rooftop_share	–	float	Share of rooftop PV when calculating capital cost of solar (joint rooftop and utility-scale PV).
social_discountrate	p.u.	float	Social discount rate to compare costs in different investment periods. 0.02 corresponds to a social discount rate of 2%.
fill_values	–	float	Default values if not specified for a technology in resources/costs.csv.
capital_cost	EUR/M	Keys should be in the 'technology' column of resources/costs.csv. Values can be any float.	For the given technologies, assumptions about their capital investment costs are set to the corresponding value. Optional; overwrites cost assumptions from resources/costs.csv.
marginal_cost	EUR/M	Keys should be in the 'technology' column of resources/costs.csv. Values can be any float.	For the given technologies, assumptions about their marginal operating costs are set to the corresponding value. Optional; overwrites cost assumptions from resources/costs.csv.
emission_prices			Specify exogenous prices for emission types listed in network.carriers to marginal costs.
– enable	bool	true or false	Add cost for a carbon-dioxide price configured in costs: emission_prices: co2 to marginal_cost of generators (other emission types listed in network.carriers possible as well)
– co2	EUR/t	float	Exogenous price of carbon-dioxide added to the marginal costs of fossil-fuelled generators according to their carbon intensity. Added through the keyword Ep in the {opts} wildcard only in the rule prepare_network`.
– co2_monthly_price	bool	true or false	Add monthly cost for a carbon-dioxide price based on historical values built by the rule build_monthly_prices

Note: rooftop_share: are based on the potentials, assuming (0.1 kW/m2 and 10 m2/person)

4.2.24 clustering

	Unit	Values	Description
focus_weights			Optionally specify the focus weights for the clustering of countries. For instance: *DE: 0.8* will distribute 80% of all nodes to Germany and 20% to the rest of the countries.
simplify_network			
– to_substations	bool	{'true','false'}	Aggregates all nodes without power injection (positive or negative, i.e. demand or generation) to electrically closest ones
– algorithm	str	One of {'kmeans', 'hac', 'modularity'}	
– feature	str	Str in the format 'carrier1+carrier2+...+carrierN-X', where CarrierI can be from {'solar', 'onwind', 'offwind', 'ror'} and X is one of {'cap', 'time'}.	
– exclude_carriers	list	List of Str like ['solar', 'onwind'] or empy list []	List of carriers which will not be aggregated. If empty, all carriers will be aggregated.
– remove stubs	bool	{'true','false'}	Controls whether radial parts of the network should be recursively aggregated. Defaults to true.
– remove_stubs_across_borders	bool	{'true','false'}	Controls whether radial parts of the network should be recursively aggregated across borders. Defaults to true.
cluster_network			
– algorithm	str	One of {'kmeans', 'hac'}	
– feature	str	Str in the format 'carrier1+carrier2+...+carrierN-X', where CarrierI can be from {'solar', 'onwind', 'offwind', 'ror'} and X is one of {'cap', 'time'}.	
– exclude_carriers	list	List of Str like ['solar', 'onwind'] or empy list []	List of carriers which will not be aggregated. If empty, all carriers will be aggregated.
– consider_efficiency_classes	bool	{'true','false'}	Aggregated each carriers into the top 10-quantile (high), the bottom 90-quantile (low), and everything in between (medium).
aggregation_strategies			
– generators			
– – {key}	str	{key} can be any of the component of the generator (str). It's value can be any that can be converted to pandas.Series using getattr(). For example one of {min, max, sum}.	Aggregates the component according to the given strategy. For example, if sum, then all values within each cluster are summed to represent the new generator.
– buses			
– – {key}	str	{key} can be any of the component of the bus (str). It's value can be any that can be converted to pandas.Series using getattr(). For example one of {min, max, sum}.	Aggregates the component according to the given strategy. For example, if sum, then all values within each cluster are summed to represent the new bus.

Note: `feature:` in `simplify_network:` are only relevant if `hac` were chosen in `algorithm`.

Tip: use `min` in `p_nom_max:` for more ` conservative assumptions.

4.2.25 `adjustments`

	Unit	Values	Description
adjustments			
– electricity	bool or dict		Parameter adjustments for capital cost, marginal cost, and maximum capacities of carriers. Applied in `prepare_network.`
– – {attr}			Attribute can be `e_nom_opt`, `p_nom_opt`, `marginal_cost` or `capital_cost`
– – – {carrier}	float	per-unit	Any carrier of the network to which parameter adjustment factor should be applied.
– sector	bool or dict		Parameter adjustments for capital cost, marginal cost, and maximum capacities of carriers. Applied in `prepare_sector_network.`
– – {attr}			Attribute can be `e_nom_opt`, `p_nom_opt`, `marginal_cost` or `capital_cost`
– – – {carrier}	float	per-unit	Any carrier of the network to which parameter adjustment factor should be applied.

4.2.26 `solving`

	Unit	Values	Description
options			
– clip_p_max_pu	p.u.	float	To avoid too small values in the renewables` per-unit availability time series values below this threshold are set to zero.
– load_shedding	bool/flo	{'true','false', float}	Add generators with very high marginal cost to simulate load shedding and avoid problem infeasibilities. If load shedding is a float, it denotes the marginal cost in EUR/kWh.
– noisy_costs	bool	{'true','false'}	Add random noise to marginal cost of generators by $\mathcal{U}(0.009, 0, 011)$ and capital cost of lines and links by $\mathcal{U}(0.09, 0, 11)$.
– skip_iterations	bool	{'true','false'}	Skip iterating, do not update impedances of branches. Defaults to true.
– rolling_horizon	bool	{'true','false'}	Whether to optimize the network in a rolling horizon manner, where the snapshot range is split into slices of size *horizon* which are solved consecutively.
– seed	–	int	Random seed for increased deterministic behaviour.
– custom_extra_functionality	–	str	Path to a Python file with custom extra functionality code to be injected into the solving rules of the workflow relative to `rules` directory.
– io_api	string	{'lp','mps','direct'}	Passed to linopy and determines the API used to communicate with the solver. With the `'lp'` and `'mps'` options linopy passes a file to the solver; with the `'direct'` option (only supported for HIGHS and Gurobi) linopy uses an in-memory python API resulting in better performance.
– track_iterations	bool	{'true','false'}	Flag whether to store the intermediate branch capacities and objective function values are recorded for each iteration in `network.lines['s_nom_opt_X']` (where `X` labels the iteration)
– min_iterations	–	int	Minimum number of solving iterations in between which resistance and reactence (`x/r`) are updated for branches according to `s_nom_opt` of the previous run.
– max_iterations	–	int	Maximum number of solving iterations in between which resistance and reactence (`x/r`) are updated for branches according to `s_nom_opt` of the previous run.
– transmission_losses	int	[0-9]	Add piecewise linear approximation of transmission losses based on n tangents. Defaults to 0, which means losses are ignored.
– linearized_unit_commitment	bool	{'true','false'}	Whether to optimise using the linearized unit commitment formulation.
– horizon	–	int	Number of snapshots to consider in each iteration. Defaults to 100.
agg_p_nom_limits			Configure per carrier generator nominal capacity constraints for individual countries if `'CCL'` is in `{opts}` wildcard.
– agg_offwind	bool	{'true','false'}	Aggregate together all the types of offwind

4.2.27 plotting

Warning: More comprehensive documentation for this segment will be released soon.

4.3 Foresight Options

4.3.1 Overnight (greenfield) scenarios

The default is to calculate a rebuilding of the energy system to meet demand, a so-called overnight or greenfield approach.

In this case, the `planning_horizons` parameter specifies the reference year for exogenously given transition paths (e.g. the level of steel recycling). It does not affect the year for cost and technology assumptions, which is set separately in the config.

```
scenario:
  planning_horizons:
  - 2050

costs:
  year: 2030
```

For running overnight scenarios, use in the `config/config.yaml`:

```
foresight: overnight
```

4.3.2 Perfect foresight scenarios

Warning: Perfect foresight is currently implemented as an experimental test version.

For running perfect foresight scenarios, you can adjust the `config/config.perfect.yaml`:

```
foresight: perfect
```

4.3.3 Myopic foresight scenarios

The myopic code can be used to investigate progressive changes in a network, for instance, those taking place throughout a transition path. The capacities installed in a certain time step are maintained in the network until their operational lifetime expires.

The myopic approach was initially developed and used in the paper Early decarbonisation of the European Energy system pays off (2020) and later further extended in Speed of technological transformations required in Europe to achieve different climate goals (2022). The current implementation complies with the PyPSA-Eur-Sec standard working flow and is compatible with using the higher resolution electricity transmission model PyPSA-Eur rather than a one-node-per-country model.

The current code applies the myopic approach to generators, storage technologies and links in the power sector. It furthermore applies it to the space and water heating sector (e.g., the share of district heating and reduced space heat demand), industry processes (e.g., steel, direct reduced iron, and aluminum production via primary route), the share of fuel cell and battery electric vehicles in land transport, and the hydrogen share in shipping (see *Supply and demand* for further information).

The following subjects within the land transport and biomass currently do not evolve with the myopic approach:

- The percentage of electric vehicles that allow demand-side management and vehicle-to-grid services.
- The annual biomass potential (default year and scenario for which potential is taken is 2030, as defined in config)

Configuration

For running myopic foresight transition scenarios, set in `config/config.yaml`:

```
foresight: myopic
```

The following options included in the `config/config.yaml` file are relevant for the myopic code.

The `{planning_horizons}` wildcard indicates the year in which the network is optimized. For a myopic optimization, this is equivalent to the investment year. To set the investment years which are sequentially simulated for the myopic investment planning, select for example:

```
planning_horizons:
- 2030
- 2040
- 2050
```

existing capacities

Grouping years indicates the bins limits for grouping the existing capacities of different technologies. Note that separate bins are defined for the power and heating plants due to different data sources.

`grouping_years_power: [1980, 1985, 1990, 1995, 2000, 2005, 2010, 2015, 2020, 2025, 2030]`

`grouping_years_heat: [1980, 1985, 1990, 1995, 2000, 2005, 2010, 2015, 2019]`

threshold capacity

If for a technology, node, and grouping bin, the capacity is lower than threshold_capacity, it is ignored.

`threshold_capacity: 10`

conventional carriers

Conventional carriers indicate carriers used in the existing conventional technologies.

conventional_carriers:

- lignite

- coal

- oil

- uranium

Options

The total carbon budget for the entire transition path can be indicated in the sector_opts in `config/config.yaml`. The carbon budget can be split among the `planning_horizons` following an exponential or beta decay. E.g. `'cb40ex0'` splits a carbon budget equal to 40 Gt $_{CO_2}$ following an exponential decay whose initial linear growth rate r is zero. They can also follow some user-specified path, if defined here. The paper Speed of technological transformations required in Europe to achieve different climate goals (2022) defines CO_2 budgets corresponding to global temperature increases (1.5C – 2C) as response to the emissions. Here, global carbon budgets are converted to European budgets assuming equal-per capita distribution which translates into a 6.43% share for Europe. The carbon budgets are in this paper distributed throughout the transition paths assuming an exponential decay. Emissions e(t) in every year t are limited by

$$e(t) = e_0(1 + (r + m)t)e^{-mt}$$

where r is the initial linear growth rate, which here is assumed to be r=0, and the decay parameter m is determined by imposing the integral of the path to be equal to the budget for Europe. Following this approach, the CO_2 budget is defined. Following the same approach as in this paper, add the following to the `scenario.sector_opts` E.g. `-cb25.7ex0` (1.5C increase) Or `cb73.9ex0` (2C increase). See details in Supplemental Note S1 Speed of technological transformations required in Europe to achieve different climate goals (2022).

General myopic code structure

The myopic code solves the network for the time steps included in `planning_horizons` in a recursive loop, so that:

1. The existing capacities (those installed before the base year are added as fixed capacities with p_nom=value, p_nom_extendable=False). E.g. for baseyear=2020, capacities installed before 2020 are added. In addition, the network comprises additional generator, storage, and link capacities with p_nom_extendable=True. The non-solved network is saved in `results/run_name/networks/prenetworks-brownfield`.

The base year is the first element in `planning_horizons`. Step 1 is implemented with the rule add_baseyear for the base year and with the rule add_brownfield for the remaining planning_horizons.

2. The 2020 network is optimized. The solved network is saved in `results/run_name/networks/postnetworks`
3. For the next planning horizon, e.g. 2030, the capacities from a previous time step are added if they are still in operation (i.e., if they fulfil planning horizon <= commissioned year + lifetime). In addition, the network comprises additional generator, storage, and link capacities with p_nom_extendable=True. The non-solved network is saved in `results/run_name/networks/prenetworks-brownfield`.

Steps 2 and 3 are solved recursively for all the planning_horizons included in `config/config.yaml`.

Rule overview

- rule add_existing baseyear

  The rule add_existing_baseyear loads the network in 'results/run_name/networks/prenetworks' and performs the following operations:

  1. Add the conventional, wind and solar power generators that were installed before the base year.
  2. Add the heating capacities that were installed before the base year.

  The existing conventional generators are retrieved from the powerplants.csv file generated by pypsa-eur which, in turn, is based on the powerplantmatching database.

  Existing wind and solar capacities are retrieved from IRENA annual statistics and distributed among the nodes in a country proportional to capacity factor. (This will be updated to include capacity distributions closer to reality.)

Existing heating capacities are retrieved from the report Mapping and analyses of the current and future (2020 - 2030) heating/cooling fuel deployment (fossil/renewables).

The heating capacities are assumed to have a lifetime indicated by the parameter lifetime in the configuration file, e.g 25 years. They are assumed to be decommissioned linearly starting on the base year, e.g., from 2020 to 2045.

Then, the resulting network is saved in `results/run_name/networks/prenetworks-brownfield`.

- rule add_brownfield

  The rule add_brownfield loads the network in `results/run_name/networks/prenetworks` and performs the following operation:

  1. Read the capacities optimized in the previous time step and add them to the network if they are still in operation (i.e., if they fulfill planning horizon < commissioned year + lifetime)

  Then, the resulting network is saved in `results/run_name/networks/prenetworks_brownfield`.

4.4 Techno-Economic Assumptions

The database of cost assumptions is retrieved from the repository PyPSA/technology-data and then saved to a file `resources/costs_{year}.csv`. The `config/config.yaml` provides options to choose a reference year and use a specific version of the repository.

The file includes cost assumptions for all included technologies for specific years compiled from various sources, namely for

- discount rate,
- lifetime,
- investment (CAPEX),
- fixed operation and maintenance (FOM),
- variable operation and maintenance (VOM),
- fuel costs,
- efficiency, and
- carbon-dioxide intensity.

Many values are taken from a database published by the Danish Energy Agency (DEA).

The given overnight capital costs are annualised to net present costs with a discount rate of r over the economic lifetime n using the annuity factor

$$a = \frac{1 - (1 + r)^{-n}}{r}.$$

Based on the parameters above the `marginal_cost` and `capital_cost` of the system components are automatically calculated.

4.4.1 Modifying Assumptions

Some cost assumptions (e.g. marginal cost and capital cost) can be directly set in the `config/config.yaml` (cf. Section *costs* in *Configuration*). To change cost assumptions in more detail, make a copy of `resources/costs_{year}.csv` and reference the new cost file in the `Snakefile`:

CHAPTER

FIVE

RULES OVERVIEW

5.1 Retrieving Data

Not all data dependencies are shipped with the git repository, since git is not suited for handling large changing files. Instead we provide separate data bundles which can be obtained using the `retrieve*` rules.

5.1.1 Rule `retrieve_databundle`

The data bundle (1.4 GB) contains common GIS datasets like NUTS3 shapes, EEZ shapes, CORINE Landcover, Natura 2000 and also electricity specific summary statistics like historic per country yearly totals of hydro generation, GDP and POP on NUTS3 levels and per-country load time-series.

This rule downloads the data bundle from zenodo and extracts it in the `data` sub-directory, such that all files of the bundle are stored in the `data/bundle` subdirectory.

The *Tutorial: Electricity-Only* uses a smaller data bundle than required for the full model (188 MB)

Relevant Settings

```
tutorial:
```

See also:

Documentation of the configuration file `config/config.yaml` at *Top-level configuration*

Outputs

- `data/bundle`: input data collected from various sources

5.1.2 Rule `retrieve_cutout`

DOI 10.5281/zenodo.6382570

Cutouts are spatio-temporal subsets of the European weather data from the ECMWF ERA5 reanalysis dataset and the CMSAF SARAH-2 solar surface radiation dataset for the year 2013. They have been prepared by and are for use with the atlite tool. You can either generate them yourself using the `build_cutouts` rule or retrieve them directly from zenodo through the rule `retrieve_cutout`. The *Tutorial: Electricity-Only* uses a smaller cutout than required for the full model (30 MB), which is also automatically downloaded.

Note: To download cutouts yourself from the ECMWF ERA5 you need to set up the CDS API.

Relevant Settings

```
tutorial:
enable:
    build_cutout:
```

See also:

Documentation of the configuration file `config/config.yaml` at *Top-level configuration*

Outputs

- `cutouts/{cutout}`: weather data from either the ERA5 reanalysis weather dataset or SARAH-2 satellite-based historic weather data.

See also:

For details see *`build_cutout`* and read the atlite documentation.

5.1.3 Rule `retrieve_natura_raster`

DOI 10.5281/zenodo.4706686

This rule, as a substitute for *`build_natura_raster`*, downloads an already rasterized version (natura.tiff) of Natura 2000 natural protection areas to reduce computation times. The file is placed into the `resources` sub-directory.

Relevant Settings

```
enable:
    build_natura_raster:
```

See also:

Documentation of the configuration file `config/config.yaml` at *Top-level configuration*

Outputs

- `resources/natura.tiff`: Rasterized version of Natura 2000 natural protection areas to reduce computation times.

See also:

For details see *`build_natura_raster`*.

5.1.4 Rule `retrieve_electricity_demand`

This rule downloads hourly electric load data for each country from the OPSD platform.

Relevant Settings

None.

Outputs

- `data/electricity_demand_raw.csv`

5.1.5 Rule `retrieve_cost_data`

This rule downloads techno-economic assumptions from the technology-data repository.

Relevant Settings

```
enable:
    retrieve_cost_data:

costs:
    year:
    version:
```

See also:

Documentation of the configuration file `config/config.yaml` at *costs*

Outputs

- `resources/costs.csv`

5.1.6 Rule `retrieve_irena`

This rule downloads the existing capacities from IRENASTAT and extracts it in the `data/existing_capacities` sub-directory.

Relevant Settings

```
enable:
    retrieve_irena:
```

See also:

Documentation of the configuration file `config.yaml` at *enable*

Outputs

- `data/existing_capacities`: existing capacities for offwind, onwind and solar

5.1.7 Rule `retrieve_ship_raster`

This rule downloads data on global shipping traffic density from the World Bank Data Catalogue.

Relevant Settings

None.

Outputs

- `data/shipdensity_global.zip`

5.1.8 Rule `retrieve_sector_databundle`

In addition to the databundle required for electricity-only studies, another databundle is required for modelling sector-coupled systems. The size of this data bundle is around 640 MB.

5.2 Building Electricity Networks

The preparation process of the PyPSA-Eur energy system model consists of a group of `snakemake` rules which are briefly outlined and explained in detail in the sections below.

Not all data dependencies are shipped with the git repository. Instead we provide separate data bundles which can be obtained using the `retrieve*` rules (*Retrieving Data*). Having downloaded the necessary data,

- *`build_shapes`* generates GeoJSON files with shapes of the countries, exclusive economic zones and NUTS3 areas.
- *`build_cutout`* prepares smaller weather data portions from ERA5 for cutout `europe-2013-era5` and SARAH for cutout `europe-2013-sarah`.

With these and the externally extracted ENTSO-E online map topology (`data/entsoegridkit`), it can build a base PyPSA network with the following rules:

- *`base_network`* builds and stores the base network with all buses, HVAC lines and HVDC links, while
- *`build_bus_regions`* determines Voronoi cells for all substations.

Then the process continues by calculating conventional power plant capacities, potentials, and per-unit availability time series for variable renewable energy carriers and hydro power plants with the following rules:

- *`build_powerplants`* for today's thermal power plant capacities using powerplantmatching allocating these to the closest substation for each powerplant,
- *`build_natura_raster`* for rasterising NATURA2000 natural protection areas,
- *`build_ship_raster`* for building shipping traffic density,
- *`build_renewable_profiles`* for the hourly capacity factors and installation potentials constrained by land-use in each substation's Voronoi cell for PV, onshore and offshore wind, and
- *`build_hydro_profile`* for the hourly per-unit hydro power availability time series.

The central rule *`add_electricity`* then ties all the different data inputs together into a detailed PyPSA network stored in `networks/elec.nc`.

5.2.1 Rule `build_bus_regions`

Creates Voronoi shapes for each bus representing both onshore and offshore regions.

Relevant Settings

```
countries:
```

See also:

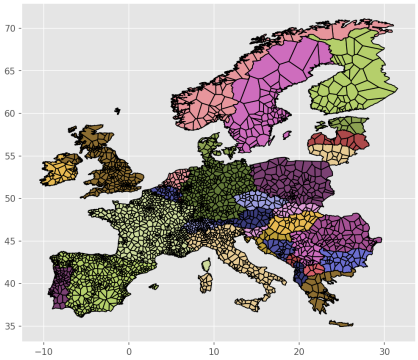
Documentation of the configuration file `config/config.yaml` at *Top-level configuration*

Inputs

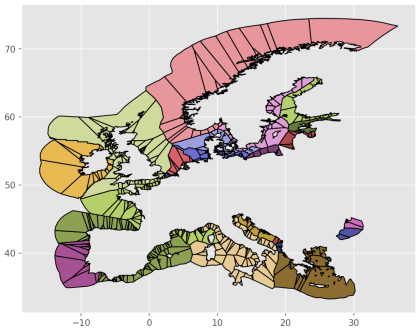
- `resources/country_shapes.geojson`: confer *Rule build_shapes*
- `resources/offshore_shapes.geojson`: confer *Rule build_shapes*
- `networks/base.nc`: confer *Rule base_network*

Outputs

- `resources/regions_onshore.geojson`:



- `resources/regions_offshore.geojson`:



Description

5.2.2 Rule `build_cutout`

Create cutouts with atlite.

For this rule to work you must have

- installed the Copernicus Climate Data Store `cdsapi` package (*install with `pip`*) and
- registered and setup your CDS API key as described on their website.

See also:

For details on the weather data read the atlite documentation. If you need help specifically for creating cutouts the corresponding section in the atlite documentation should be helpful.

Relevant Settings

```
atlite:
    nprocesses:
    cutouts:
        {cutout}:
```

See also:

Documentation of the configuration file `config/config.yaml` at *atlite*

Inputs

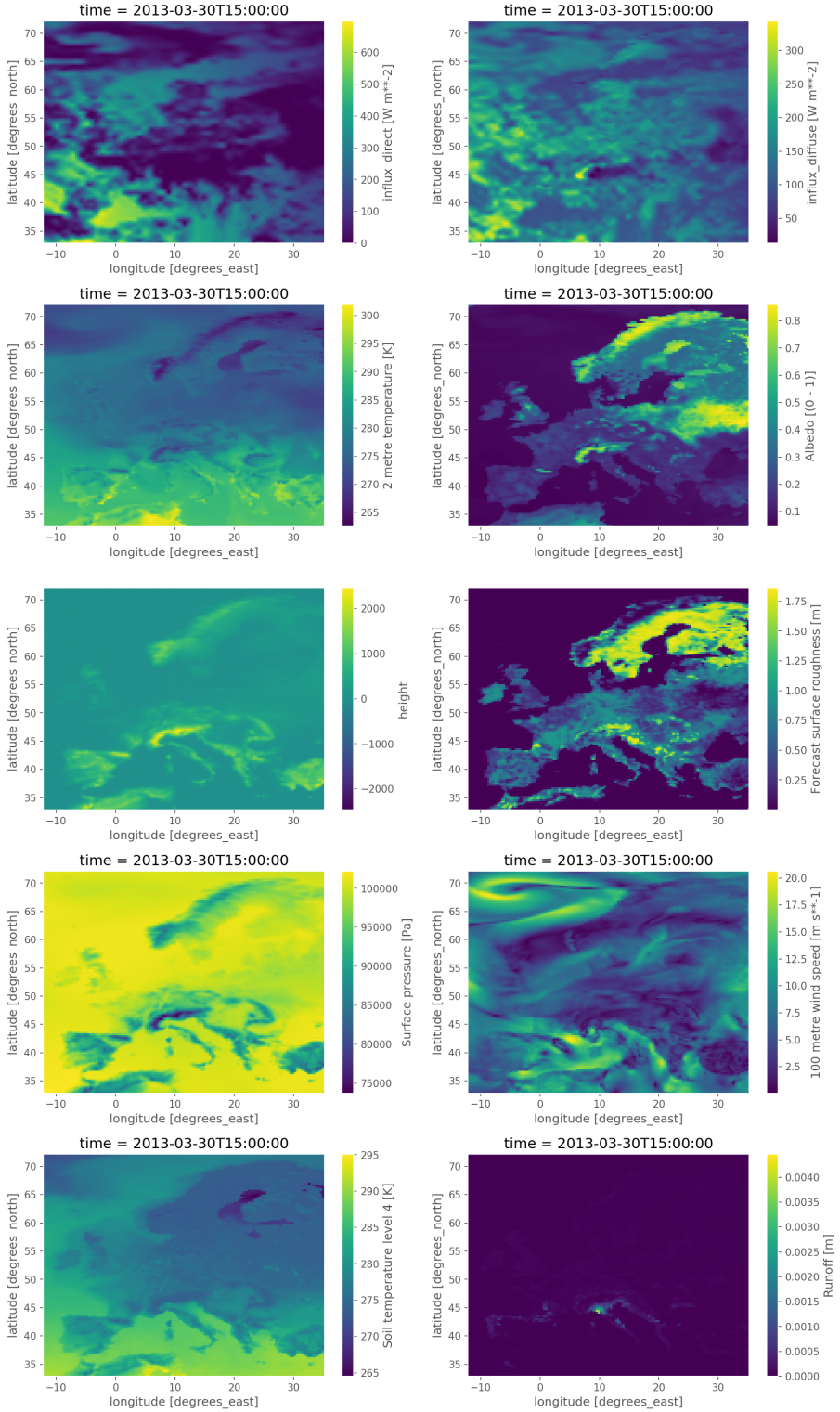
None

Outputs

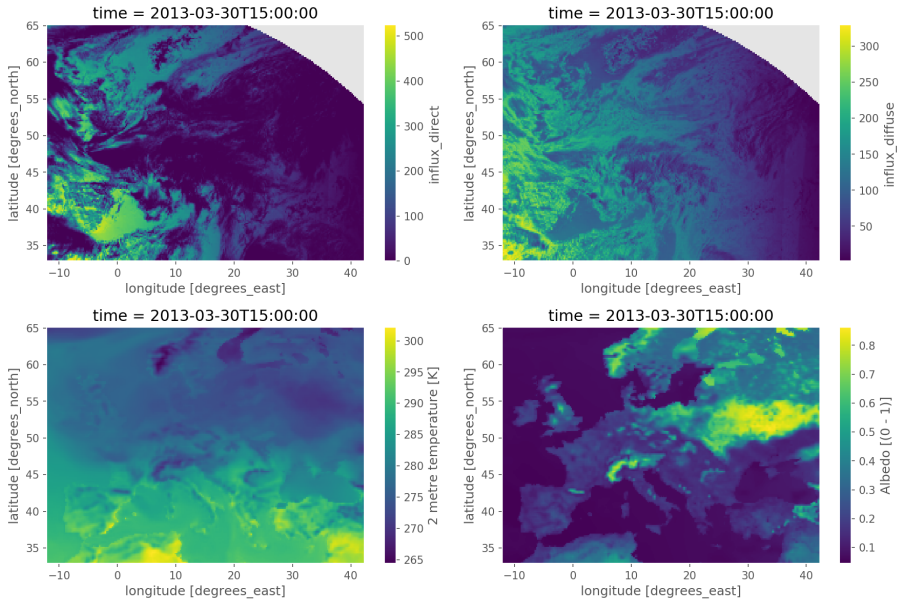
- `cutouts/{cutout}`: weather data from either the ERA5 reanalysis weather dataset or SARAH-2 satellite-based historic weather data with the following structure:

ERA5 cutout:

Field	Dimensions	Unit	Description
pressure	time, y, x	Pa	Surface pressure
temperature	time, y, x	K	Air temperature 2 meters above the surface.
soil temperature	time, y, x	K	Soil temperature between 1 meters and 3 meters depth (layer 4).
influx_toa	time, y, x	Wm*2	Top of Earth's atmosphere TOA incident solar radiation
influx_dir	time, y, x	Wm*2	Total sky direct solar radiation at surface
runoff	time, y, x	m	Runoff (volume per area)
roughness	y, x	m	Forecast surface roughness (roughness length)
height	y, x	m	Surface elevation above sea level
albedo	time, y, x	–	Albedo measure of diffuse reflection of solar radiation. Calculated from relation between surface solar radiation downwards (Jm**-2) and surface net solar radiation (Jm**-2). Takes values between 0 and 1.
influx_dif	time, y, x	Wm*2	Diffuse solar radiation at surface. Surface solar radiation downwards minus direct solar radiation.
wnd100	time, y, x	ms**1	Wind speeds at 100 meters (regardless of direction)



A **SARAH-2 cutout** can be used to amend the fields `temperature`, `influx_toa`, `influx_direct`, `albedo`, `influx_diffuse` of ERA5 using satellite-based radiation observations.



Description

5.2.3 Rule `prepare_links_p_nom`

Extracts capacities of HVDC links from `Wikipedia.

<https://en.wikipedia.org/wiki/List_of_HVDC_projects>`_.

Relevant Settings

```
enable:
    prepare_links_p_nom:
```

See also:

Documentation of the configuration file `config/config.yaml` at *Top-level configuration*

Inputs

None

Outputs

- `data/links_p_nom.csv`: A plain download of https://en.wikipedia.org/wiki/List_of_HVDC_projects#Europe plus extracted coordinates.

Description

None

5.2.4 Rule `build_natura_raster`

Rasters the vector data of the `Natura 2000.

<https://en.wikipedia.org/wiki/Natura_2000>`_ natural protection areas onto all cutout regions.

Relevant Settings

```
renewable:
    {technology}:
        cutout:
```

See also:

Documentation of the configuration file `config/config.yaml` at *renewable*

Inputs

- `data/bundle/natura/Natura2000_end2015.shp`: Natura 2000 natural protection areas.

Outputs

- `resources/natura.tiff`: Rasterized version of Natura 2000 natural protection areas to reduce computation times.

Description

5.2.5 Rule `base_network`

Creates the network topology from a `ENTSO-E map extract.

<https://github.com/PyPSA/GridKit/tree/master/entsoe>`_ (March 2022) as a PyPSA network.

Relevant Settings

```
countries:

electricity:
    voltages:

lines:
    types:
    s_max_pu:
    under_construction:

links:
    p_max_pu:
    under_construction:
    include_tyndp:

transformers:
    x:
    s_nom:
    type:
```

See also:

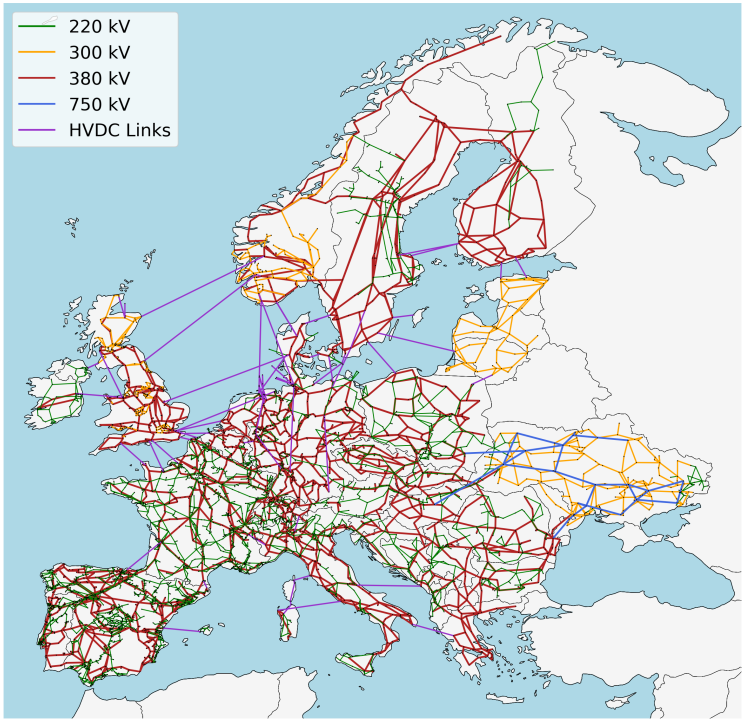
Documentation of the configuration file `config/config.yaml` at *snapshots*, *Top-level configuration*, *electricity*, *load*, *conventional*, *links*, *transformers*

Inputs

- `data/entsoegridkit`: Extract from the geographical vector data of the online ENTSO-E Interactive Map by the GridKit toolkit dating back to March 2022.
- `data/parameter_corrections.yaml`: Corrections for `data/entsoegridkit`
- `data/links_p_nom.csv`: confer links
- `data/links_tyndp.csv`: List of projects in the TYNDP 2018 that are at least *in permitting* with fields for start- and endpoint (names and coordinates), length, capacity, construction status, and project reference ID.
- `resources/country_shapes.geojson`: confer *Rule build_shapes*
- `resources/offshore_shapes.geojson`: confer *Rule build_shapes*
- `resources/europe_shape.geojson`: confer *Rule build_shapes*

Outputs

- `networks/base.nc`



Description

5.2.6 Rule `build_shapes`

Creates GIS shape files of the countries, exclusive economic zones and `NUTS3 < https://en.wikipedia.org/wiki/Nomenclature_of_Territorial_Units_for_Statistics> `_ areas.

Relevant Settings

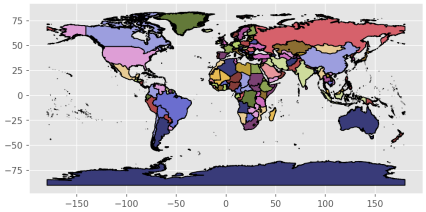
```
countries:
```

See also:

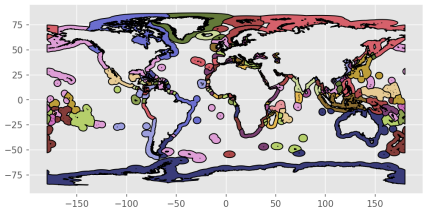
Documentation of the configuration file `config/config.yaml` at *Top-level configuration*

Inputs

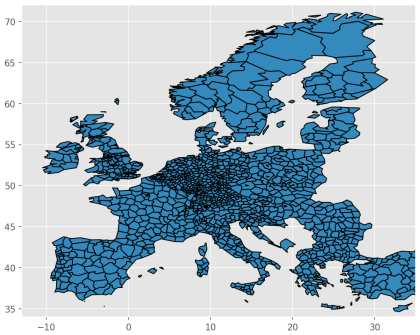
- `data/bundle/naturalearth/ne_10m_admin_0_countries.shp`: World country shapes



- `data/bundle/eez/World_EEZ_v8_2014.shp`: World exclusive economic zones (EEZ)



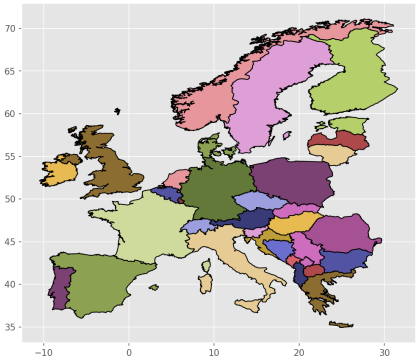
- `data/bundle/NUTS_2013_60M_SH/data/NUTS_RG_60M_2013.shp`: Europe NUTS3 regions



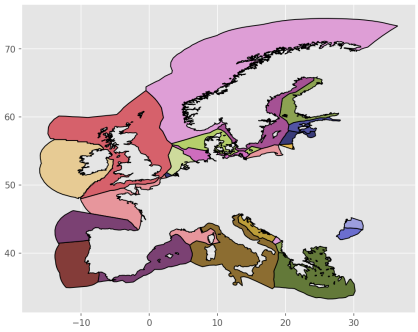
- `data/bundle/nama_10r_3popgdp.tsv.gz`: Average annual population by NUTS3 region (eurostat)
- `data/bundle/nama_10r_3gdp.tsv.gz`: Gross domestic product (GDP) by NUTS 3 regions (eurostat)
- `data/bundle/ch_cantons.csv`: Mapping between Swiss Cantons and NUTS3 regions
- `data/bundle/je-e-21.03.02.xls`: Population and GDP data per Canton (BFS - Swiss Federal Statistical Office)

Outputs

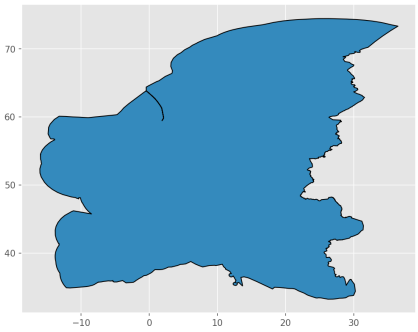
- `resources/country_shapes.geojson`: country shapes out of country selection



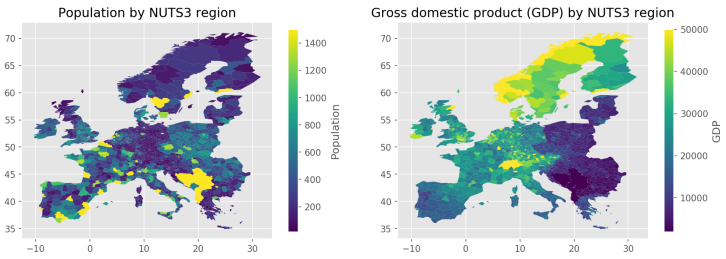
- `resources/offshore_shapes.geojson`: EEZ shapes out of country selection



- `resources/europe_shape.geojson`: Shape of Europe including countries and EEZ



- `resources/nuts3_shapes.geojson`: NUTS3 shapes out of country selection including population and GDP data.



Description

5.2.7 Rule `build_powerplants`

Retrieves conventional powerplant capacities and locations from powerplantmatching, assigns these to buses and creates a `.csv` file. It is possible to amend the powerplant database with custom entries provided in `data/custom_powerplants.csv`. Lastly, for every substation, powerplants with zero-initial capacity can be added for certain fuel types automatically.

Relevant Settings

```
electricity:
  powerplants_filter:
  custom_powerplants:
  everywhere_powerplants:
```

See also:

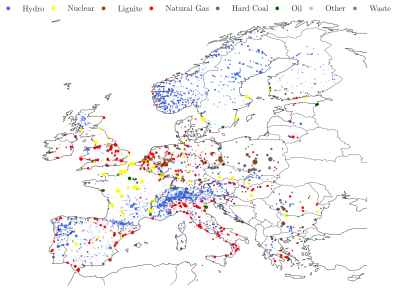
Documentation of the configuration file `config/config.yaml` at *Rule add_electricity*

Inputs

- `networks/base.nc`: confer *Rule base_network*.
- `data/custom_powerplants.csv`: custom powerplants in the same format as powerplantmatching provides

Outputs

- `resource/powerplants.csv`: A list of conventional power plants (i.e. neither wind nor solar) with fields for name, fuel type, technology, country, capacity in MW, duration, commissioning year, retrofit year, latitude, longitude, and dam information as documented in the powerplantmatching README; additionally it includes information on the closest substation/bus in `networks/base.nc`.



Source: powerplantmatching on GitHub

Description

The configuration options `electricity: powerplants_filter` and `electricity: custom_powerplants` can be used to control whether data should be retrieved from the original powerplants database or from custom amendmends. These specify pandas.query commands. In addition the configuration option `electricity: everywhere_powerplants` can be used to place powerplants with zero-initial capacity of certain fuel types at all substations.

1. Adding all powerplants from custom:

```
powerplants_filter: false
custom_powerplants: true
```

2. Replacing powerplants in e.g. Germany by custom data:

```
powerplants_filter: Country not in ['Germany']
custom_powerplants: true
```

or

```
powerplants_filter: Country not in ['Germany']
custom_powerplants: Country in ['Germany']
```

3. Adding additional built year constraints:

```
powerplants_filter: Country not in ['Germany'] and YearCommissioned <= 2015
custom_powerplants: YearCommissioned <= 2015
```

4. Adding powerplants at all substations for 4 conventional carrier types:

```
everywhere_powerplants: ['Natural Gas', 'Coal', 'nuclear', 'OCGT']
```

5.2.8 Rule `build_electricity_demand`

This rule downloads the load data from Open Power System Data Time series. For all countries in the network, the per country load timeseries are extracted from the dataset. After filling small gaps linearly and large gaps by copying time-slice of a given period, the load data is exported to a `.csv` file.

Relevant Settings

```
snapshots:

load:
    interpolate_limit: time_shift_for_large_gaps: manual_adjustments:
```

See also:

Documentation of the configuration file `config/config.yaml` at *load*

Inputs

- `data/electricity_demand_raw.csv`:

Outputs

- `resources/electricity_demand.csv`:

5.2.9 Rule `build_monthly_prices`

This script extracts monthly fuel prices of oil, gas, coal and lignite, as well as CO2 prices.

Inputs

- `data/energy-price-trends-xlsx-5619002.xlsx`: energy price index of fossil fuels
- `emission-spot-primary-market-auction-report-2019-data.xls`: CO2 Prices spot primary auction

Outputs

- `data/validation/monthly_fuel_price.csv`
- `data/validation/CO2_price_2019.csv`

Description

The rule *`build_monthly_prices`* collects monthly fuel prices and CO2 prices and translates them from different input sources to pypsa syntax

Data sources:

[1] Fuel price index. Destatis https://www.destatis.de/EN/Home/_node.html [2] average annual fuel price lignite, ENTSO-E https://2020.entsos-tyndp-scenarios.eu/fuel-commodities-and-carbon-prices/ [3] CO2 Prices, Emission spot primary auction, EEX https://www.eex.com/en/market-data/environmental-markets/eua-primary-auction-spot-download

Data was accessed at 16.5.2023

5.2.10 Rule `build_ship_raster`

Transforms the global ship density data from the `World Bank Data Catalogue.

<https://datacatalog.worldbank.org/search/dataset/0037580/Global-Shipping-Traffic-Density>`_ to the size of the considered cutout. The global ship density raster is later used for the exclusion when calculating the offshore potentials.

Relevant Settings

```
renewable:
    {technology}:
        cutout:
```

See also:

Documentation of the configuration file `config/config.yaml` at *renewable*

Inputs

- `data/bundle/shipdensity/shipdensity_global.zip`: Global shipping traffic density from World Bank Data Catalogue.

Outputs

- `resources/europe_shipdensity_raster.nc`: Reduced version of global shipping traffic density from World Bank Data Catalogue to reduce computation time.

Description

5.2.11 Rule `determine_availability_matrix_MD_UA`

Create land elibility analysis for Ukraine and Moldova with different datasets.

5.2.12 Rule `build_renewable_profiles`

Calculates for each network node the (i) installable capacity (based on land- use), (ii) the available generation time series (based on weather data), and (iii) the average distance from the node for onshore wind, AC-connected offshore wind, DC-connected offshore wind and solar PV generators. In addition for offshore wind it calculates the fraction of the grid connection which is under water.

Note: Hydroelectric profiles are built in script `build_hydro_profiles`.

Relevant settings

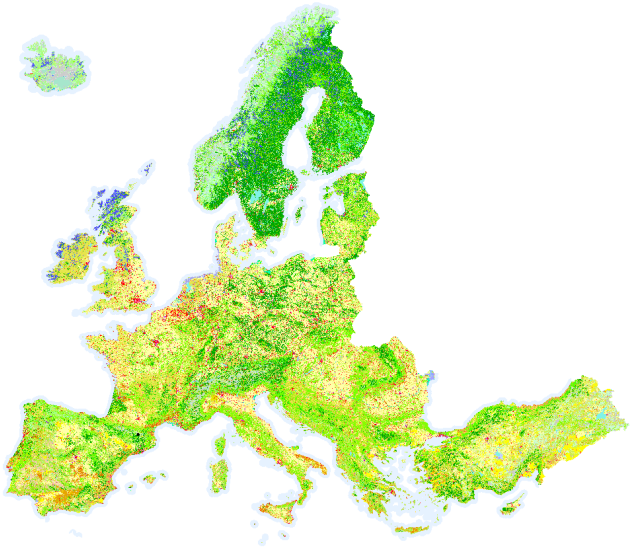
```
snapshots:

atlite:
    nprocesses:

renewable:
    {technology}:
        cutout: corine: luisa: grid_codes: distance: natura: max_depth:
        max_shore_distance: min_shore_distance: capacity_per_sqkm:
        correction_factor: min_p_max_pu: clip_p_max_pu: resource:
```

See also:

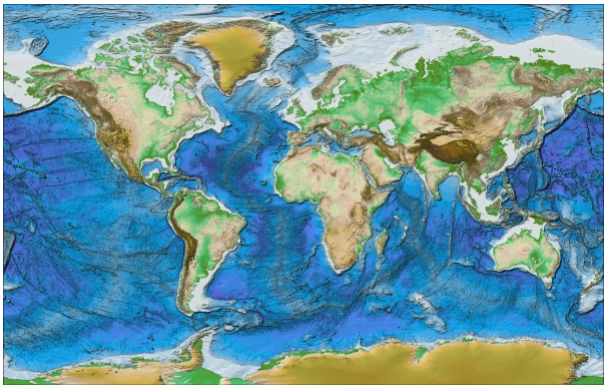
Documentation of the configuration file `config/config.yaml` at *snapshots*, *atlite*, *renewable*

Inputs

- `data/bundle/corine/g250_clc06_V18_5.tif`: CORINE Land Cover (CLC) inventory on 44 classes of land use (e.g. forests, arable land, industrial, urban areas) at 100m resolution.



- `data/LUISA_basemap_020321_50m.tif`: LUISA Base Map land coverage dataset at 50m resolution similar to CORINE. For codes in relation to CORINE land cover, see Annex 1 of the technical documentation.
- `data/bundle/GEBCO_2014_2D.nc`: A bathymetric data set with a global terrain model for ocean and land at 15 arc-second intervals by the General Bathymetric Chart of the Oceans (GEBCO).



Source: GEBCO

- `resources/natura.tiff`: confer *Rule build_natura_raster*
- `resources/offshore_shapes.geojson`: confer *Rule build_shapes*
- `resources/regions_onshore.geojson`: (if not offshore wind), confer *Rule build_bus_regions*
- `resources/regions_offshore.geojson`: (if offshore wind), *Rule build_bus_regions*
- `"cutouts/" + params["renewable"][{technology}]['cutout']`: *Rule build_cutout*

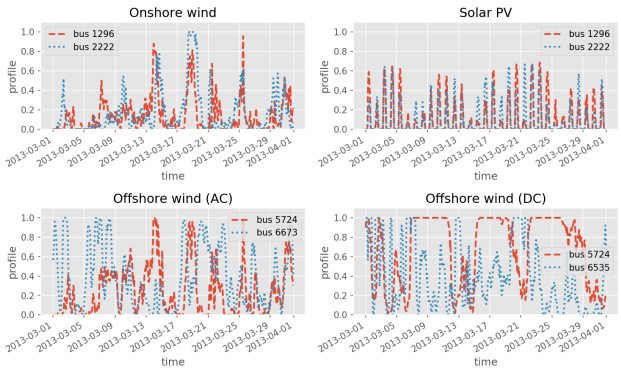
- `networks/base.nc`: *Rule base_network*

Outputs

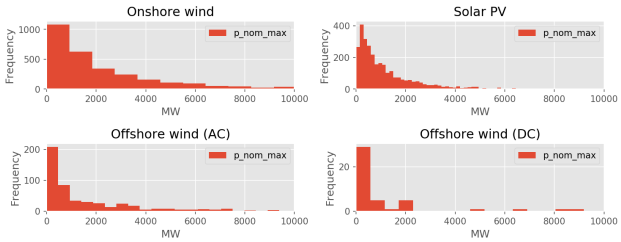
- `resources/profile_{technology}.nc` with the following structure

Field	Dimensions	Description
profile	bus, time	the per unit hourly availability factors for each node
weight	bus	sum of the layout weighting for each node
p_nom_max	bus	maximal installable capacity at the node (in MW)
potential	y, x	layout of generator units at cutout grid cells inside the Voronoi cell (maximal installable capacity at each grid cell multiplied by capacity factor)
average_distance	bus	average distance of units in the Voronoi cell to the grid node (in km)
underwater_fraction	bus	fraction of the average connection distance which is under water (only for offshore)

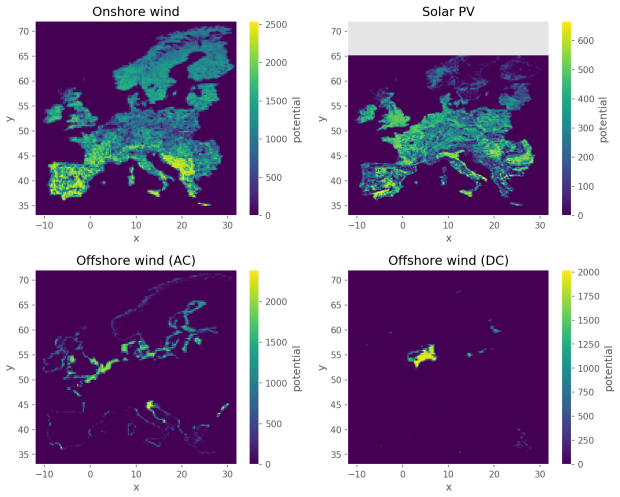
– **profile**



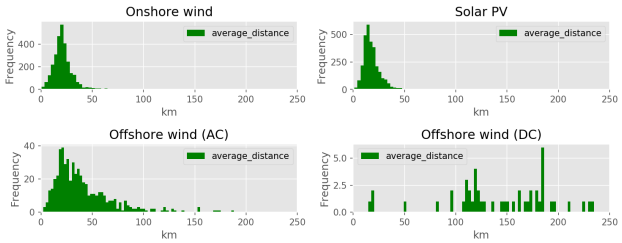
– **p_nom_max**



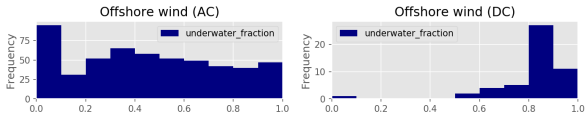
– **potential**



– **average_distance**



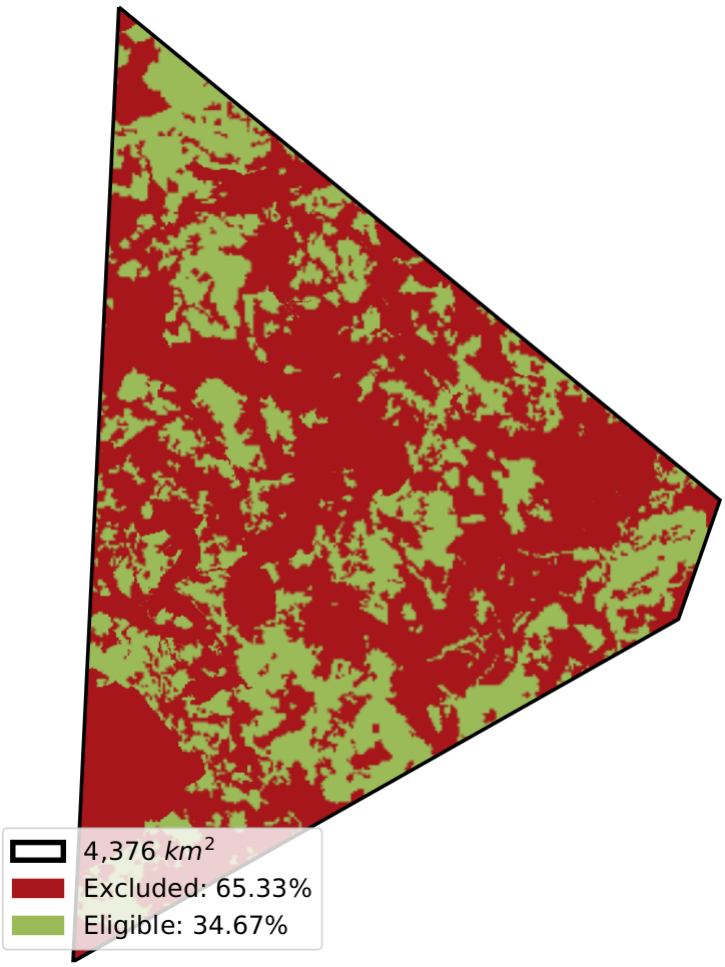
– **underwater_fraction**



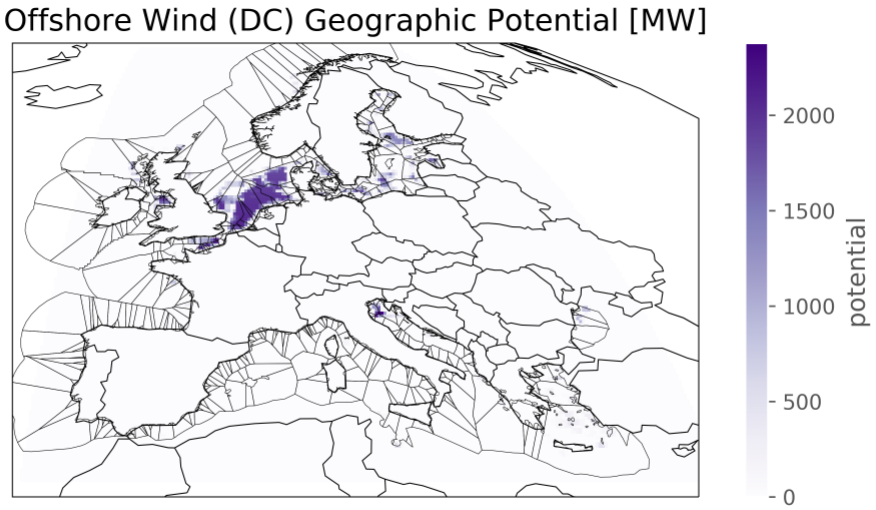
Description

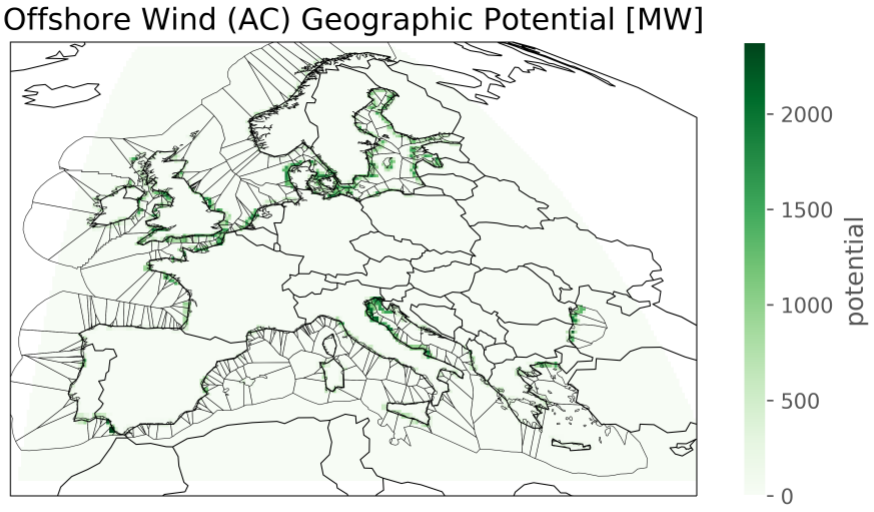
This script functions at two main spatial resolutions: the resolution of the network nodes and their Voronoi cells, and the resolution of the cutout grid cells for the weather data. Typically the weather data grid is finer than the network nodes, so we have to work out the distribution of generators across the grid cells within each Voronoi cell. This is done by taking account of a combination of the available land at each grid cell and the capacity factor there.

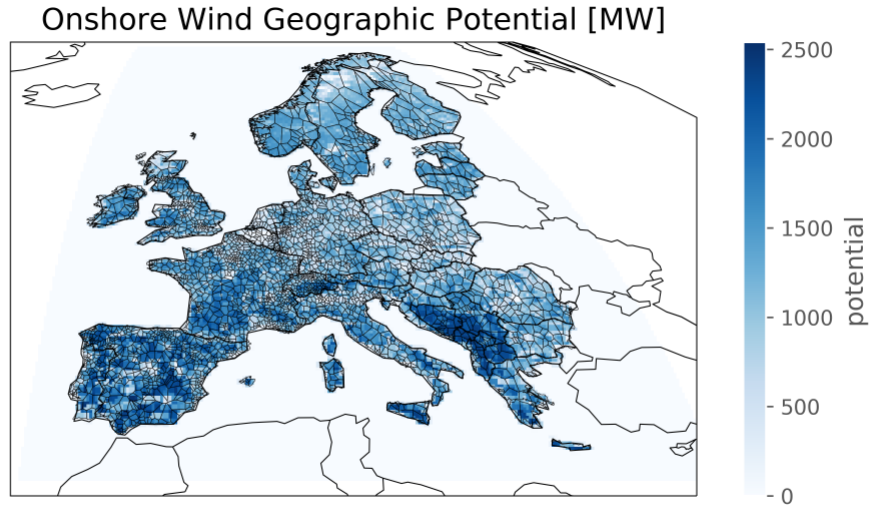
First the script computes how much of the technology can be installed at each cutout grid cell and each node using the atlite library. This uses the CORINE land use data, LUISA land use data, Natura2000 nature reserves, GEBCO bathymetry data, and shipping lanes.

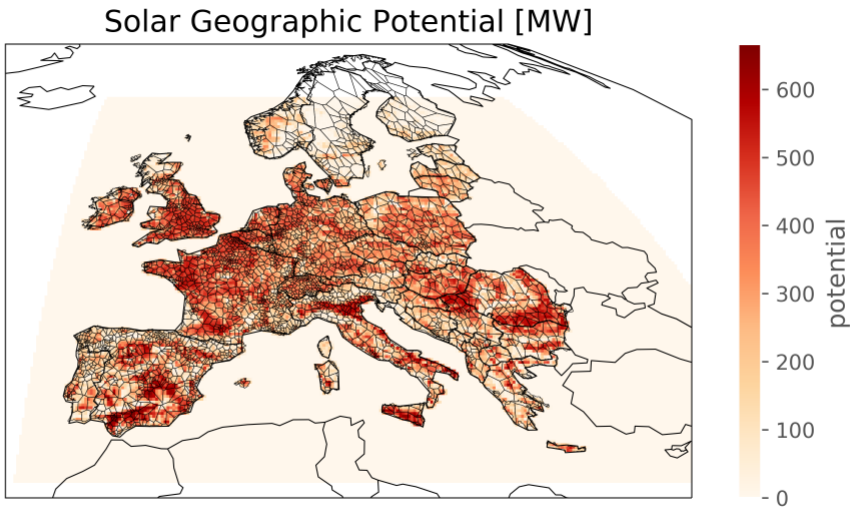


To compute the layout of generators in each node's Voronoi cell, the installable potential in each grid cell is multiplied with the capacity factor at each grid cell. This is done since we assume more generators are installed at cells with a higher capacity factor.




Offshore Wind (AC) Geographic Potential [MW]
2000
1500
1000
500
0
potential


Onshore Wind Geographic Potential [MW]
2500
2000
1500
1000
500
0
potential



This layout is then used to compute the generation availability time series from the weather data cutout from `atlite`.

The maximal installable potential for the node (*p_nom_max*) is computed by adding up the installable potentials of the individual grid cells. If the model comes close to this limit, then the time series may slightly overestimate production since it is assumed the geographical distribution is proportional to capacity factor.

5.2.13 Rule `build_hydro_profile`

Build hydroelectric inflow time-series for each country.

Relevant Settings

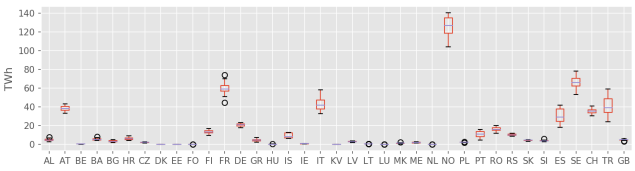
```
countries:

renewable:
    hydro:
        cutout:
        clip_min_inflow:
```

See also:

Documentation of the configuration file `config/config.yaml` at *Top-level configuration*, *renewable*

Inputs

- `data/bundle/eia_hydro_annual_generation.csv`: Hydroelectricity net generation per country and year (EIA)

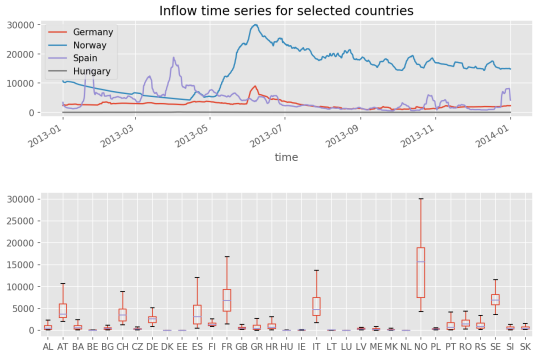


- `resources/country_shapes.geojson`: confer *Rule build_shapes*
- `"cutouts/" + config["renewable"]['hydro']['cutout']`: confer *Rule build_cutout*

Outputs

- `resources/profile_hydro.nc`:

Field	Dimensions	Description
inflow	countries, time	Inflow to the state of charge (in MW), e.g. due to river inflow in hydro reservoir.



Description

See also:

build_renewable_profiles

5.2.14 Rule `add_electricity`

Adds electrical generators and existing hydro storage units to a base network.

Relevant Settings

```
costs:
    year:
    version:
    dicountrate:
    emission_prices:

electricity:
    max_hours:
    marginal_cost:
    capital_cost:
    conventional_carriers:
    co2limit:
```

(continues on next page)

(continued from previous page)

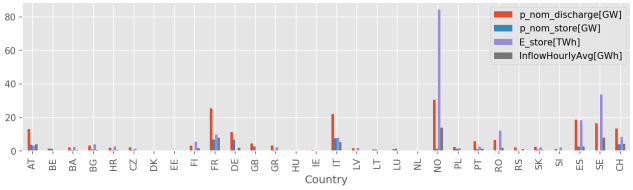
```
    extendable_carriers:
    estimate_renewable_capacities:


load:
    scaling_factor:

renewable:
    hydro:
        carriers:
        hydro_max_hours:
        hydro_capital_cost:

lines:
    length_factor:
```

See also:

Documentation of the configuration file `config/config.yaml` at *costs*, *electricity*, *load*, *renewable*, *conventional*

Inputs

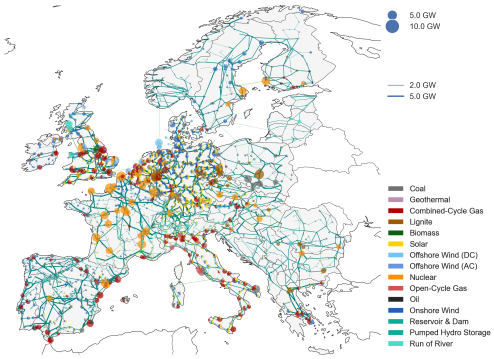
- `resources/costs.csv`: The database of cost assumptions for all included technologies for specific years from various sources; e.g. discount rate, lifetime, investment (CAPEX), fixed operation and maintenance (FOM), variable operation and maintenance (VOM), fuel costs, efficiency, carbon-dioxide intensity.
- `data/bundle/hydro_capacities.csv`: Hydropower plant store/discharge power capacities, energy storage capacity, and average hourly inflow by country.



- `data/geth2015_hydro_capacities.csv`: alternative to capacities above; not currently used!
- `resources/electricity_demand.csv` Hourly per-country electricity demand profiles.
- `resources/regions_onshore.geojson`: confer *Rule build_bus_regions*
- `resources/nuts3_shapes.geojson`: confer *Rule build_shapes*
- `resources/powerplants.csv`: confer *Rule build_powerplants*
- `resources/profile_{}.nc`: all technologies in `config["renewables"].keys()`, confer *Rule build_renewable_profiles*.
- `networks/base.nc`: confer *Rule base_network*

Outputs

- `networks/elec.nc`:



Description

The rule *add_electricity* ties all the different data inputs from the preceding rules together into a detailed PyPSA network that is stored in `networks/elec.nc`. It includes:

- today's transmission topology and transfer capacities (optionally including lines which are under construction according to the config settings `lines: under_construction` and `links: under_construction`),
- today's thermal and hydro power generation capacities (for the technologies listed in the config setting `electricity: conventional_carriers`), and
- today's load time-series (upsampled in a top-down approach according to population and gross domestic product)

It further adds extendable `generators` with **zero** capacity for

- photovoltaic, onshore and AC- as well as DC-connected offshore wind installations with today's locational, hourly wind and solar capacity factors (but **no** current capacities),
- additional open- and combined-cycle gas turbines (if `OCGT` and/or `CCGT` is listed in the config setting `electricity: extendable_carriers`)

5.3 Simplifying Electricity Networks

The simplification `snakemake` rules prepare **approximations** of the full model, for which it is computationally viable to co-optimize generation, storage and transmission capacities.

- *simplify_network* transforms the transmission grid to a 380 kV only equivalent network, while
- *cluster_network* uses a k-means based clustering technique to partition the network into a given number of zones and then reduce the network to a representation with one bus per zone.

The simplification and clustering steps are described in detail in the paper

- Jonas Hörsch and Tom Brown. The role of spatial scale in joint optimisations of generation and transmission for European highly renewable scenarios), *14th International Conference on the European Energy Market*, 2017. arXiv:1705.07617, doi:10.1109/EEM.2017.7982024.

After simplification and clustering of the network, additional components may be appended in the rule *add_extra_components* and the network is prepared for solving in *prepare_network*.

5.3.1 Rule `simplify_network`

Lifts electrical transmission network to a single 380 kV voltage layer, removes dead-ends of the network, and reduces multi-hop HVDC connections to a single link.

Relevant Settings

```
clustering:
  simplify_network:
  cluster_network:
  aggregation_strategies:

costs:
    year:
    version:
    fill_values:
    marginal_cost:
    capital_cost:

electricity:
    max_hours:

lines:
    length_factor:

links:
    p_max_pu:

solving:
    solver:
        name:
```

See also:

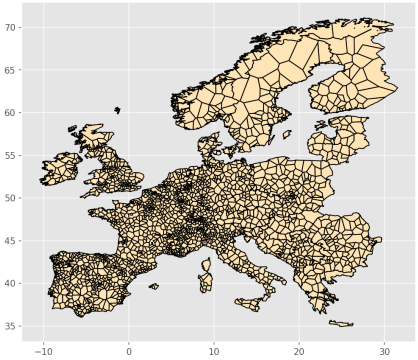
Documentation of the configuration file `config/config.yaml` at *costs*, *electricity*, *renewable*, *conventional*, *links*, *solving*

Inputs

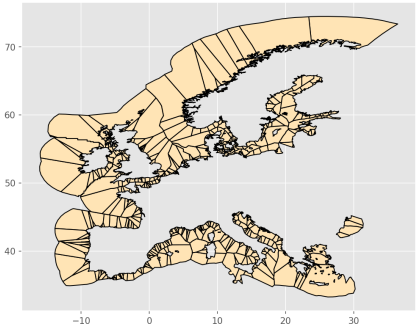
- `resources/costs.csv`: The database of cost assumptions for all included technologies for specific years from various sources; e.g. discount rate, lifetime, investment (CAPEX), fixed operation and maintenance (FOM), variable operation and maintenance (VOM), fuel costs, efficiency, carbon-dioxide intensity.
- `resources/regions_onshore.geojson`: confer *Rule build_bus_regions*
- `resources/regions_offshore.geojson`: confer *Rule build_bus_regions*
- `networks/elec.nc`: confer *Rule add_electricity*

Outputs

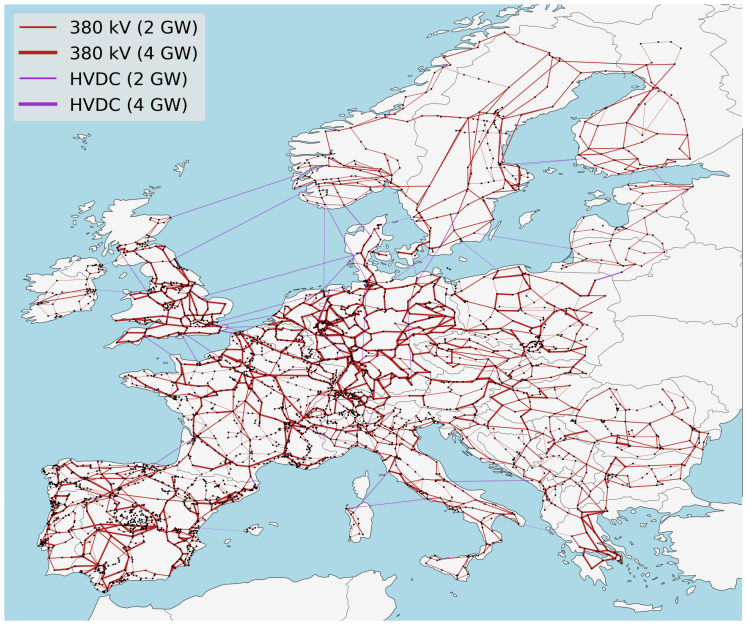
- `resources/regions_onshore_elec_s{simpl}.geojson`:



- `resources/regions_offshore_elec_s{simpl}.geojson`:



- `resources/busmap_elec_s{simpl}.csv`: Mapping of buses from `networks/elec.nc` to `networks/elec_s{simpl}.nc`;
- `networks/elec_s{simpl}.nc`:



Description

The rule *simplify_network* does up to four things:

1. Create an equivalent transmission network in which all voltage levels are mapped to the 380 kV level by the function `simplify_network(...)`.
2. DC only sub-networks that are connected at only two buses to the AC network are reduced to a single representative link in the function `simplify_links(...)`. The components attached to buses in between are moved to the nearest endpoint. The grid connection cost of offshore wind generators are added to the capital costs of the generator.
3. Stub lines and links, i.e. dead-ends of the network, are sequentially removed from the network in the function `remove_stubs(...)`. Components are moved along.
4. Optionally, if an integer were provided for the wildcard `{simpl}` (e.g. `networks/elec_s500.nc`), the network is clustered to this number of clusters with the routines from the `cluster_network` rule with the function `cluster_network.cluster(...)`. This step is usually skipped!

5.3.2 Rule `cluster_network`

Creates networks clustered to `{cluster}` number of zones with aggregated buses, generators and transmission corridors.

Relevant Settings

```
clustering:
  cluster_network:
  aggregation_strategies:
  focus_weights:

solving:
    solver:
        name:

lines:
    length_factor:
```

See also:

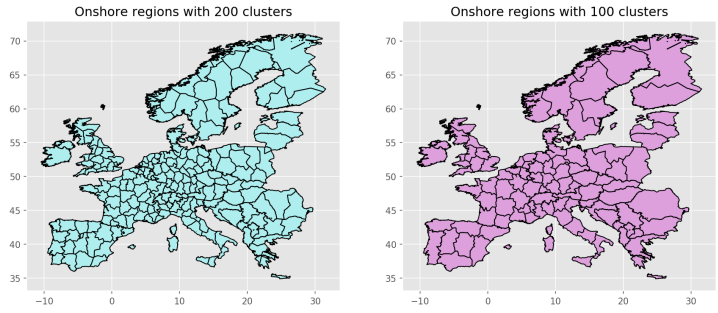
Documentation of the configuration file `config/config.yaml` at *Top-level configuration*, *renewable*, *solving*, *conventional*

Inputs

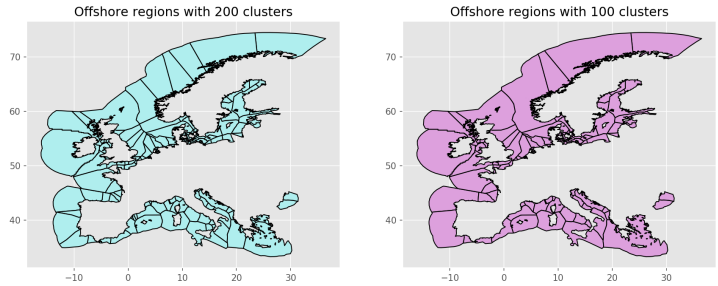
- `resources/regions_onshore_elec_s{simpl}.geojson`: confer *Rule simplify_network*
- `resources/regions_offshore_elec_s{simpl}.geojson`: confer *Rule simplify_network*
- `resources/busmap_elec_s{simpl}.csv`: confer *Rule simplify_network*
- `networks/elec_s{simpl}.nc`: confer *Rule simplify_network*
- `data/custom_busmap_elec_s{simpl}_{clusters}.csv`: optional input

Outputs

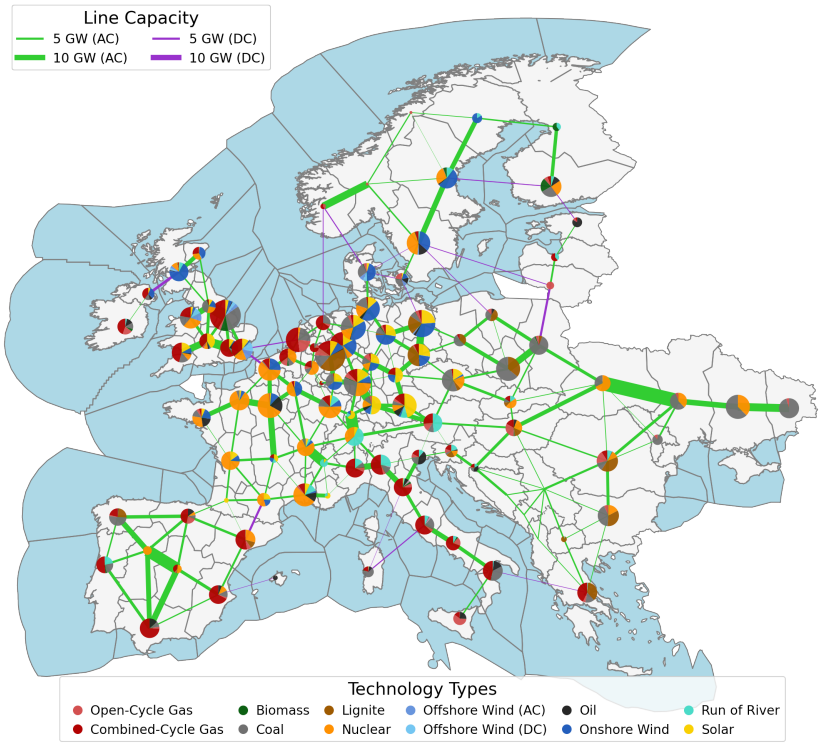
- `resources/regions_onshore_elec_s{simpl}_{clusters}.geojson`:



- `resources/regions_offshore_elec_s{simpl}_{clusters}.geojson`:



- `resources/busmap_elec_s{simpl}_{clusters}.csv`: Mapping of buses from `networks/elec_s{simpl}.nc` to `networks/elec_s{simpl}_{clusters}.nc`;
- `resources/linemap_elec_s{simpl}_{clusters}.csv`: Mapping of lines from `networks/elec_s{simpl}.nc` to `networks/elec_s{simpl}_{clusters}.nc`;
- `networks/elec_s{simpl}_{clusters}.nc`:



Description

Note: Why is clustering used both in `simplify_network` and `cluster_network` ?

Consider for example a network `networks/elec_s100_50.nc` in which `simplify_network` clusters the network to 100 buses and in a second step `cluster_network`` reduces it down to 50 buses.

In preliminary tests, it turns out, that the principal effect of changing spatial resolution is actually only partially due to the transmission network. It is more important to differentiate between wind generators with higher capacity factors from those with lower capacity factors, i.e. to have a higher spatial resolution in the renewable generation than in the number of buses.

The two-step clustering allows to study this effect by looking at networks like `networks/elec_s100_50m.nc`. Note the additional `m` in the `{cluster}` wildcard. So in the example network there are still up to 100 different wind generators.

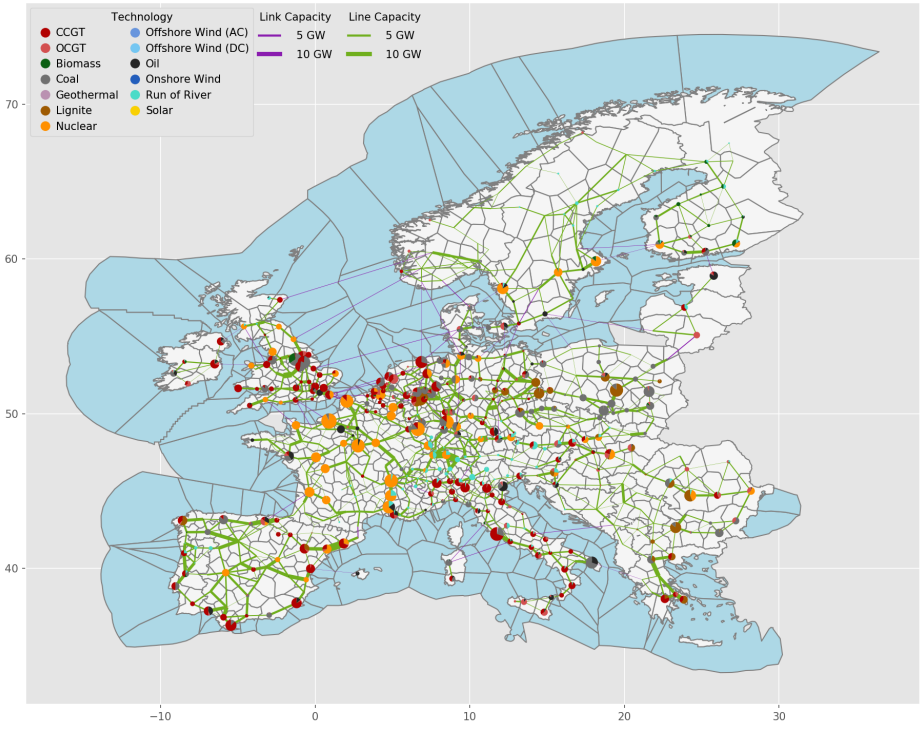
In combination these two features allow you to study the spatial resolution of the transmission network separately from the spatial resolution of renewable generators.

Is it possible to run the model without the `simplify_network` rule?

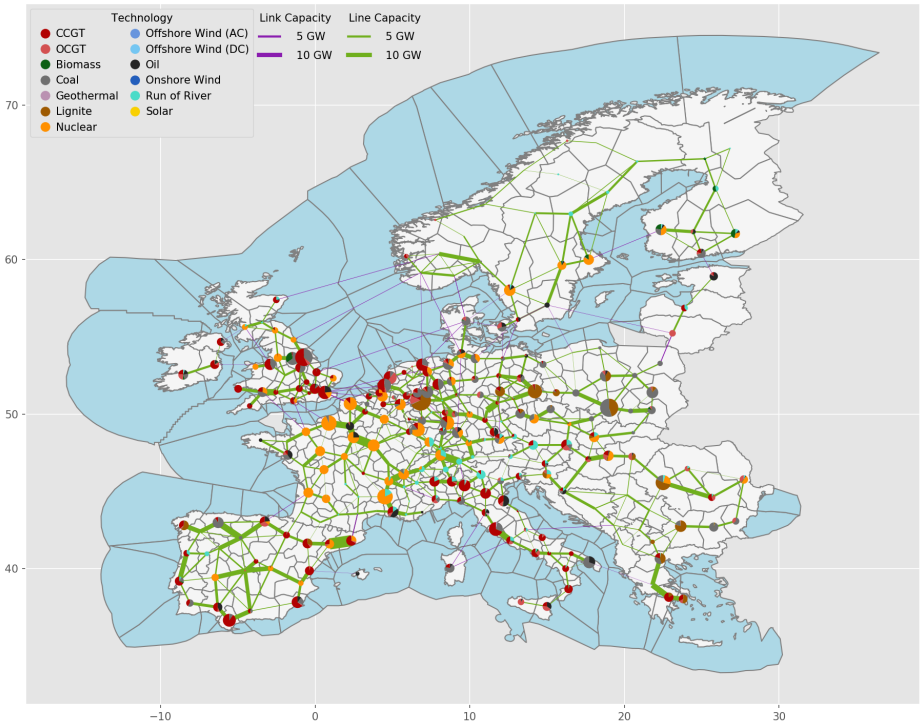
No, the network clustering methods in the PyPSA module pypsa.clustering.spatial do not work reliably with multiple voltage levels and transformers.

Tip: The rule `cluster_networks` runs for all `scenario` s in the configuration file the rule *`cluster_network`*.

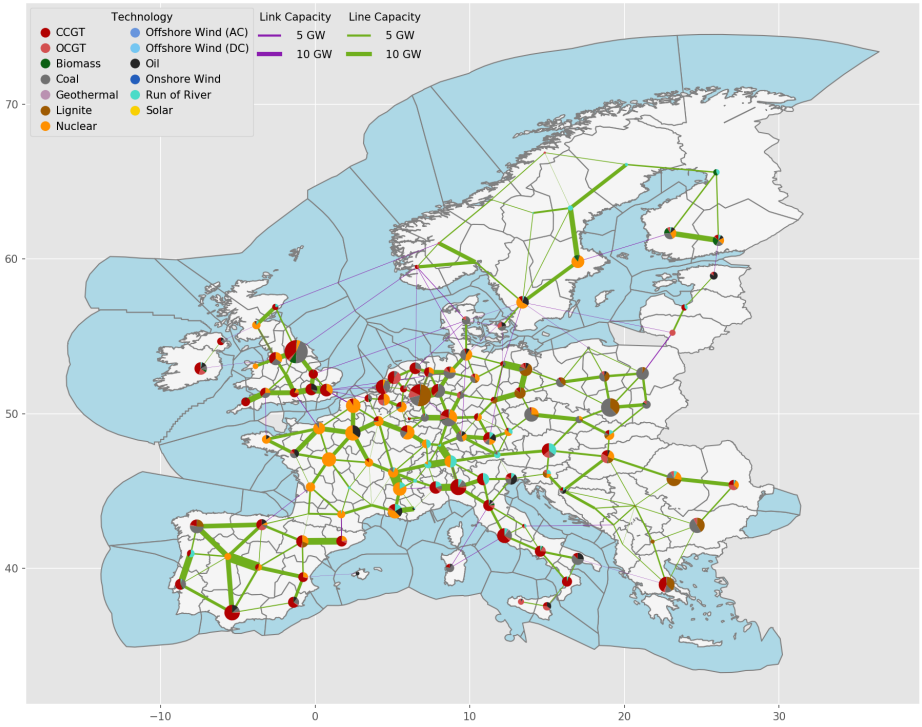
Exemplary unsolved network clustered to 512 nodes:



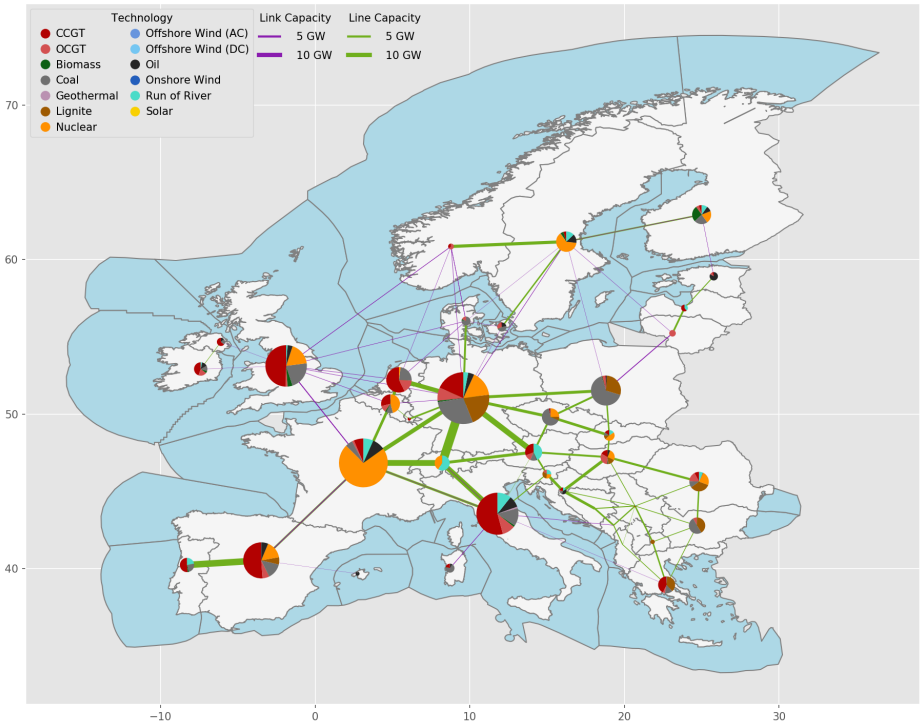
Exemplary unsolved network clustered to 256 nodes:



Exemplary unsolved network clustered to 128 nodes:



Exemplary unsolved network clustered to 37 nodes:



5.3.3 Rule `add_extra_components`

Adds extra extendable components to the clustered and simplified network.

Relevant Settings

```
costs:
    year:
    version:
    dicountrate:
    emission_prices:

electricity:
    max_hours:
    marginal_cost:
    capital_cost:
    extendable_carriers:
        StorageUnit:
        Store:
```

See also:

Documentation of the configuration file `config/config.yaml` at *costs*, *electricity*

Inputs

- `resources/costs.csv`: The database of cost assumptions for all included technologies for specific years from various sources; e.g. discount rate, lifetime, investment (CAPEX), fixed operation and maintenance (FOM), variable operation and maintenance (VOM), fuel costs, efficiency, carbon-dioxide intensity.

Outputs

- `networks/elec_s{simpl}_{clusters}_ec.nc`:

Description

The rule *add_extra_components* attaches additional extendable components to the clustered and simplified network. These can be configured in the `config/config.yaml` at `electricity: extendable_carriers:`. It processes `networks/elec_s{simpl}_{clusters}.nc` to build `networks/elec_s{simpl}_{clusters}_ec.nc`, which in contrast to the former (depending on the configuration) contain with **zero** initial capacity

- `StorageUnits` of carrier 'H2' and/or 'battery'. If this option is chosen, every bus is given an extendable `StorageUnit` of the corresponding carrier. The energy and power capacities are linked through a parameter that specifies the energy capacity as maximum hours at full dispatch power and is configured in `electricity: max_hours:`. This linkage leads to one investment variable per storage unit. The default `max_hours` lead to long-term hydrogen and short-term battery storage units.
- `Stores` of carrier 'H2' and/or 'battery' in combination with `Links`. If this option is chosen, the script adds extra buses with corresponding carrier where energy `Stores` are attached and which are connected to the corresponding power buses via two links, one each for charging and discharging. This leads to three investment variables for the energy capacity, charging and discharging capacity of the storage unit.

5.3.4 Rule `prepare_network`

Prepare PyPSA network for solving according to *The {opts} wildcard* and *The {ll} wildcard*, such as.

- adding an annual **limit** of carbon-dioxide emissions,
- adding an exogenous **price** per tonne emissions of carbon-dioxide (or other kinds),
- setting an **N-1 security margin** factor for transmission line capacities,
- specifying an expansion limit on the **cost** of transmission expansion,
- specifying an expansion limit on the **volume** of transmission expansion, and
- reducing the **temporal** resolution by averaging over multiple hours or segmenting time series into chunks of varying lengths using `tsam`.

Relevant Settings

```
costs:
    year:
    version:
    fill_values:
    emission_prices:
    marginal_cost:
    capital_cost:

electricity:
    co2limit:
    max_hours:
```

See also:

Documentation of the configuration file `config/config.yaml` at *costs*, *electricity*

Inputs

- `resources/costs.csv`: The database of cost assumptions for all included technologies for specific years from various sources; e.g. discount rate, lifetime, investment (CAPEX), fixed operation and maintenance (FOM), variable operation and maintenance (VOM), fuel costs, efficiency, carbon-dioxide intensity.
- `networks/elec_s{simpl}_{clusters}.nc`: confer *Rule cluster_network*

Outputs

- `networks/elec_s{simpl}_{clusters}_ec_l{ll}_{opts}.nc`: Complete PyPSA network that will be handed to the `solve_network` rule.

Description

Tip: The rule `prepare_elec_networks` runs for all `scenario` s in the configuration file the rule *prepare_network*.

5.4 Building Sector-Coupled Networks

Warning: This part of the documentation is under development.

5.4.1 Rule `add_brownfield`

Prepares brownfield data from previous planning horizon.

5.4.2 Rule `add_existing_baseyear`

Adds existing power and heat generation capacities for initial planning horizon.

5.4.3 Rule `build_existing_heating_distribution`

Builds table of existing heat generation capacities for initial planning horizon.

5.4.4 Rule `build_ammonia_production`

Build historical annual ammonia production per country in ktonNH3/a.

Inputs

- `data/bundle-sector/myb1-2017-nitro.xls`

Outputs

- `resources/ammonia_production.csv`

Description

This functions takes data from the Minerals Yearbook (June 2024) published by the US Geological Survey (USGS) and the National Minerals Information Center and extracts the annual ammonia production per country in ktonN/a. The data is converted to ktonNH3/a.

5.4.5 Rule build_biomass_potentials

Compute biogas and solid biomass potentials for each clustered model region using data from JRC ENSPRESO.

5.4.6 Rule build_biomass_transport_costs

Reads biomass transport costs for different countries of the JRC report.

"The JRC-EU-TIMES model. Bioenergy potentials for EU and neighbouring countries." (2015)

converts them from units 'EUR per km/ton' -> 'EUR/ (km MWh)'

assuming as an approximation energy content of wood pellets

@author: bw0928

5.4.7 Rule build_clustered_population_layouts

Build population layouts for all clustered model regions as total as well as split by urban and rural population.

5.4.8 Rule build_cop_profiles

Build coefficient of performance (COP) time series for air- or ground-sourced heat pumps.

The COP is a function of the temperature difference between source and sink.

The quadratic regression used is based on Staffell et al. (2012) https://doi.org/10.1039/C2EE22653G.

5.4.9 Rule build_energy_totals

Build total energy demands per country using JRC IDEES, eurostat, and EEA data.

5.4.10 Rule build_heat_totals

Approximate heat demand for all weather years.

5.4.11 Rule build_gas_input_locations

Build import locations for fossil gas from entry-points, LNG terminals and production sites with data from Sci-GRID_gas and Global Energy Monitor.

5.4.12 Rule build_gas_network

Preprocess gas network based on data from bthe SciGRID_gas project (https://www.gas.scigrid.de/).

5.4.13 Rule build_daily_heat_demand

Build heat demand time series using heating degree day (HDD) approximation.

5.4.14 Rule build_hourly_heat_demand

Build hourly heat demand time series from daily ones.

5.4.15 Rule build_district_heat_share

Build district heat shares at each node, depending on investment year.

5.4.16 Rule build_industrial_distribution_key

Build spatial distribution of industries from Hotmaps database.

Inputs

- `resources/regions_onshore_elec_s{simpl}_{clusters}.geojson`
- `resources/pop_layout_elec_s{simpl}_{clusters}.csv`

Outputs

- `resources/industrial_distribution_key_elec_s{simpl}_{clusters}.csv`

Description

This rule uses the *Hotmaps database <https://gitlab.com/hotmaps/industrial_sites/industrial_sites_Industrial_Database>*. After removing entries without valid locations, it assigns each industrial site to a bus region based on its location. Then, it calculates the nodal distribution key for each sector based on the emissions of the industrial sites in each region. This leads to a distribution key of 1 if there is only one bus per country and <1 if there are multiple buses per country. The sum over buses of one country is 1.

The following subcategories of industry are considered: - Iron and steel - Cement - Refineries - Paper and printing - Chemical industry - Glass - Non-ferrous metals - Non-metallic mineral products - Other non-classified Furthermore, the population distribution is added - Population

5.4.17 Rule build_industrial_energy_demand_per_country_today

Build industrial energy demand per country.

Inputs

- `data/bundle/jrc-idees-2015`
- `industrial_production_per_country.csv`

Outputs

- `resources/industrial_energy_demand_per_country_today.csv`

Description

This rule uses the industrial_production_per_country.csv file and the JRC-IDEES data to derive an energy demand per country and sector. If the country is not in the EU28, an average energy demand depending on the production volume is derived. For each country and each subcategory of

- Alumina production
- Aluminium - primary production
- Aluminium - secondary production
- Ammonia
- Cement
- Ceramics & other NMM
- Chlorine
- Electric arc
- Food, beverages and tobacco
- Glass production
- HVC (NSC)
- Integrated steelworks
- Machinery Equipment
- Methanol
- Other Industrial Sectors
- Other chemicals
- Other non-ferrous metals
- Paper production
- Pharmaceutical products etc.
- Printing and media reproduction
- Pulp production
- Textiles and leather
- Transport Equipment
- Wood and wood products

the output file contains the energy demand in TWh/a for the following carriers

- biomass
- electricity
- gas
- heat
- hydrogen
- liquid
- other
- solid
- waste

5.4.18 Rule `build_industrial_energy_demand_per_node_today`

Build industrial energy demand per model region.

Inputs

- `resources/industrial_distribution_key_elec_s{simpl}_{clusters}.csv`
- `resources/industrial_energy_demand_per_country_today.csv`

Outputs

- `resources/industrial_energy_demand_per_node_today_elec_s{simpl}_{clusters}.csv`

Description

This rule maps the industrial energy demand per country *industrial_energy_demand_per_country_today.csv* to each bus region. The energy demand per country is multiplied by the mapping value from the file `industrial_distribution_key_elec_s{simpl}_{clusters}.csv` between 0 and 1 to get the industrial energy demand per bus.

The unit of the energy demand is TWh/a.

5.4.19 Rule `build_industrial_energy_demand_per_node`

Build industrial energy demand per model region.

Inputs

- `resources/industrial_energy_demand_today_elec_s{simpl}_{clusters}.csv`
- `resources/industry_sector_ratios_{planning_horizons}.csv`
- `resources/industrial_production_elec_s{simpl}_{clusters}_{planning_horizons}.csv`

Outputs

- `resources/industrial_energy_demand_elec_s{simpl}_{clusters}_{planning_horizons}.csv`

Description

This rule aggregates the energy demand of the industrial sectors per model region. For each bus, the following carriers are considered: - electricity - coal - coke - solid biomass - methane - hydrogen - low-temperature heat - naphtha - ammonia - methanol - process emission - process emission from feedstock

which can later be used as values for the industry load.

5.4.20 Rule `build_industrial_production_per_country_tomorrow`

Build future industrial production per country.

Relevant Settings

```
industry:
    St_primary_fraction:
    DRI_fraction:
    Al_primary_fraction:
    HVC_primary_fraction:
    HVC_mechanical_recycling_fraction:
    HVC_chemical_recycling_fraction:
```

See also:

Documentation of the configuration file `config/config.yaml` at industry

Inputs

- `resources/industrial_production_per_country.csv`

Outputs

- `resources/industrial_production_per_country_tomorrow_{planning_horizons}.csv`

Description

This rule uses the `industrial_production_per_country.csv` file and the expected recycling rates to calculate the future production of the industrial sectors.

St_primary_fraction The fraction of steel that is coming from primary production. This is more energy intensive than recycling steel (secondary production).

DRI_CH4_fraction The fraction of primary steel that is produced in DRI H2 plants.

DRI_H2_fraction The fraction of primary steel that is produced in DRI CH4 plants.

Al_primary_fraction The fraction of aluminium that is coming from primary production. This is more energy intensive than recycling aluminium (secondary production).

HVC_NSC_fraction The fraction of high value chemicals that are coming from primary production (crude oil or Fischer Tropsch).

HVC_NSC_CC_fraction The fraction of high value chemicals that are coming from primary production (crude oil or Fischer Tropsch) using CC.

HVC_MTO_fraction The fraction of high value chemicals that are coming from primary production through methonol to olefin route (MTO).

HVC_mechanical_recycling_fraction The fraction of high value chemicals that are coming from mechanical recycling.

HVC_chemical_recycling_fraction The fraction of high value chemicals that are coming from chemical recycling.

If not already present, the information is added as new column in the output file.

The unit of the production is kt/a.

5.4.21 Rule `build_industrial_production_per_country`

This rule builds the historical industrial production per country.

Relevant Settings

```
countries:
```

Inputs

- `resources/ammonia_production.csv`
- `data/bundle-sector/jrc-idees-2015`
- `data/eurostat`

Outputs

- `resources/industrial_production_per_country.csv`

Description

The industrial production is taken from the *JRC-IDEES <https://joint-research-centre.ec.europa.eu/potencia-policy-oriented-tool-energy-and-climate-change-impact-assessment/jrc-idees_en)>*. This dataset provides detailed information about the consumption of energy for various processes. If the country is not part of the EU28, the energy consumption in the industrial sectors is taken from the *Eurostat <https://ec.europa.eu/eurostat/de/data/database>* dataset. The industrial production is calculated for the year specified in the config[“industry”][“reference_year”].

The ammonia production is provided by the rule *build_ammonia_production <https://pypsa-eur.readthedocs.io/en/latest/sector.html#module-build_ammonia_production>*. Since Switzerland is not part of the EU28 nor reported by eurostat, the energy consumption in the industrial sectors is taken from the `BFE <https://www.bfe.admin.ch/bfe/de/home/versorgung/statistik-und-geodaten/energiestatistiken/energieverbrauch-nach-verwendungszweck.html> dataset. After the industrial production is calculated, the basic chemicals are separated into ammonia, chlorine, methanol and HVC. The production of these chemicals is assumed to be proportional to the production of basic chemicals without ammonia.

The following subcategories [kton/a] are considered: - Electric arc - Integrated steelworks - Other chemicals - Pharmaceutical products etc. - Cement - Ceramics & other NMM - Glass production - Pulp production - Paper production - Printing and media reproduction - Food, beverages and tobacco - Alumina production - Aluminium - primary production - Aluminium - secondary production - Other non-ferrous metals - Transport Equipment - Machinery Equipment - Textiles and leather - Wood and wood products - Other Industrial Sectors - Ammonia - HVC (NSC) - Chlorine - Methanol

5.4.22 Rule `build_industrial_production_per_node`

Build industrial production per model region.

Inputs

- `resources/industrial_distribution_key_elec_s{simpl}_{clusters}.csv`
- `resources/industrial_production_per_country_tomorrow_{planning_horizons}.csv`

Outputs

- `resources/industrial_production_per_node_elec_s{simpl}_{clusters}_{planning_horizons}.csv`

Description

This rule maps the industrial production per country from a certain time horizon to each bus region. The mapping file provides a value between 0 and 1 for each bus and industry subcategory, indicating the share of the country’s production of that sector in that bus. The industrial production per country is multiplied by the mapping value to get the industrial production per bus. The unit of the production is kt/a.

5.4.23 Rule build_industry_sector_ratios

Build best case specific energy consumption by carrier and category.

Relevant Settings

```
industry:
    ammonia:
```

Inputs

- resources/ammonia_production.csv
- data/bundle-sector/jrc-idees-2015

Outputs

- resources/industry_sector_ratios.csv

Description

This script uses the *JRC-IDEES <https://joint-research-centre.ec.europa.eu/potencia-policy-oriented-tool-energy-and-climate-change-impact-assessment/jrc-idees_en>* data to calculate an EU28 average specific energy consumption by carrier and industries. The industries are according to the rule *industrial_production_per_country <https://pypsa-eur.readthedocs.io/en/latest/sector.html#module-build_industrial_production_per_country>*.

The following carriers are considered: - elec - coal - coke - biomass - methane - hydrogen - heat - naphtha - ammonia - methanol - process emission - process emission from feedstock

If the *config["industry"]["ammonia"] <https://pypsa-eur.readthedocs.io/en/latest/configuration.html#industry>* is set to true the ammonia demand is not converted to hydrogen and electricity but is considered as a separate carrier.

The unit of the specific energy consumption is MWh/t material and tCO2/t material for process emissions.

5.4.24 Rule build_population_layouts

Build mapping between cutout grid cells and population (total, urban, rural).

5.4.25 Rule build_population_weighted_energy_totals

Distribute country-level energy demands by population.

5.4.26 Rule `build_retro_cost`

This script calculates the space heating savings through better insulation of the thermal envelope of a building and corresponding costs for different building types in different countries.

Methodology

The energy savings calculations are based on the

EN ISO 13790 / seasonal method https://www.iso.org/obp/ui/#iso:std:iso:13790:ed-2:v1:en:

- calculations heavily oriented on the TABULAWebTool

http://webtool.building-typology.eu/ http://www.episcope.eu/fileadmin/tabula/public/docs/report/TABULA_CommonCalculationMethod.pdf which is following the EN ISO 13790 / seasonal method

- **building stock data:**
  mainly: hotmaps project https://gitlab.com/hotmaps/building-stock missing: EU building observatory https://ec.europa.eu/energy/en/eu-buildings-database
- **building types with typical surfaces/ standard values:**
  - tabula https://episcope.eu/fileadmin/tabula/public/calc/tabula-calculator.xlsx

Basic Equations

The basic equations:

The Energy needed for space heating E_space [W/m^2] are calculated as the sum of heat losses and heat gains:

E_space = H_losses - H_gains

Heat losses constitute from the losses through heat transmission (H_tr [W/m^{2}K]) (this includes heat transfer through building elements and thermal bridges) and losses by ventilation (H_ve [W/m^{2}K]):

H_losses = (H_tr + H_ve) * F_red * (T_threshold - T_averaged_d_heat) * d_heat * 1/365

F_red : reduction factor, considering non-uniform heating [°C], p.16 chapter 2.6 [-] T_threshold : heating temperature threshold, assumed 15 C d_heat : Length of heating season, number of days with daily averaged temperature below T_threshold T_averaged_d_heat : mean daily averaged temperature of the days within heating season d_heat

Heat gains constitute from the gains by solar radiation (H_solar) and internal heat gains (H_int) weighted by a gain utilisation factor nu:

H_gains = nu * (H_solar + H_int)

Structure

The script has the following structure:

(0) fixed parameters are set

(1) prepare data, bring to same format

(2) calculate space heat demand depending on additional insulation material

(3) calculate costs for corresponding additional insulation material

(4) get cost savings per retrofitting measures for each sector by weighting with heated floor area

5.4.27 Rule build_salt_cavern_potentials

Build salt cavern potentials for hydrogen storage.

Technical Potential of Salt Caverns for Hydrogen Storage in Europe CC-BY 4.0 https://doi.org/10.20944/preprints201910.0187.v1 https://doi.org/10.1016/j.ijhydene.2019.12.161

Figure 6. Distribution of potential salt cavern sites across Europe with their corresponding energy densities (cavern storage potential divided by the volume).

Figure 7. Total cavern storage potential in European countries classified as onshore, offshore and within 50 km of shore.

The regional distribution is taken from the map (Figure 6) and scaled to the capacities from the bar chart split by nearshore (<50km from sea), onshore (>50km from sea), offshore (Figure 7).

5.4.28 Rule build_sequestration_potentials

Build regionalised geological sequestration potential for carbon dioxide using data from CO2Stop.

5.4.29 Rule build_shipping_demand

Build regional demand for international navigation based on outflow volume of ports.

5.4.30 Rule build_solar_thermal_profiles

Build solar thermal collector time series.

5.4.31 Rule build_temperature_profiles

Build time series for air and soil temperatures per clustered model region.

5.4.32 Rule build_transport_demand

Build land transport demand per clustered model region including efficiency improvements due to drivetrain changes, time series for electric vehicle availability and demand-side management constraints.

5.4.33 Rule cluster_gas_network

Cluster gas transmission network to clustered model regions.

5.4.34 Rule `prepare_sector_network`

Adds all sector-coupling components to the network, including demand and supply technologies for the buildings, transport and industry sectors.

5.5 Solving Networks

After generating and simplifying the networks they can be solved through the rule *solve_network* by using the collection rules `solve_elec_networks` or `solve_sector_networks`. Moreover, networks can be solved for dispatch-only analyses on an already solved network with *solve_operations_network*.

5.5.1 Rule `solve_network`

Solves optimal operation and capacity for a network with the option to iteratively optimize while updating line reactances.

This script is used for optimizing the electrical network as well as the sector coupled network.

Description

Total annual system costs are minimised with PyPSA. The full formulation of the linear optimal power flow (plus investment planning is provided in the documentation of PyPSA.

The optimization is based on the `network.optimize()` function. Additionally, some extra constraints specified in *solve_network* are added.

Note: The rules `solve_elec_networks` and `solve_sector_networks` run the workflow for all scenarios in the configuration file (`scenario:`) based on the rule *solve_network*.

5.5.2 Rule `solve_operations_network`

Solves linear optimal dispatch in hourly resolution using the capacities of previous capacity expansion in rule *solve_network*.

5.5.3 Rule `solve_sector_network`

Warning: More comprehensive documentation for this rule will be released soon.

5.6 Plotting and Summaries

5.6.1 Rule `make_summary`

Create summary CSV files for all scenario runs including costs, capacities, capacity factors, curtailment, energy balances, prices and other metrics.

5.6.2 Rule `plot_summary`

Creates plots from summary CSV files.

5.6.3 Rule `plot_power_network`

Creates plots for optimised power network topologies and regional generation, storage and conversion capacities built.

5.6.4 Rule `plot_power_network_perfect`

Creates plots for optimised power network topologies and regional generation, storage and conversion capacities built for the perfect foresight scenario.

5.6.5 Rule `plot_hydrogen_network`

Creates map of optimised hydrogen network, storage and selected other infrastructure.

5.6.6 Rule `plot_gas_network`

Creates map of optimised gas network, storage and selected other infrastructure.

CHAPTER

SIX

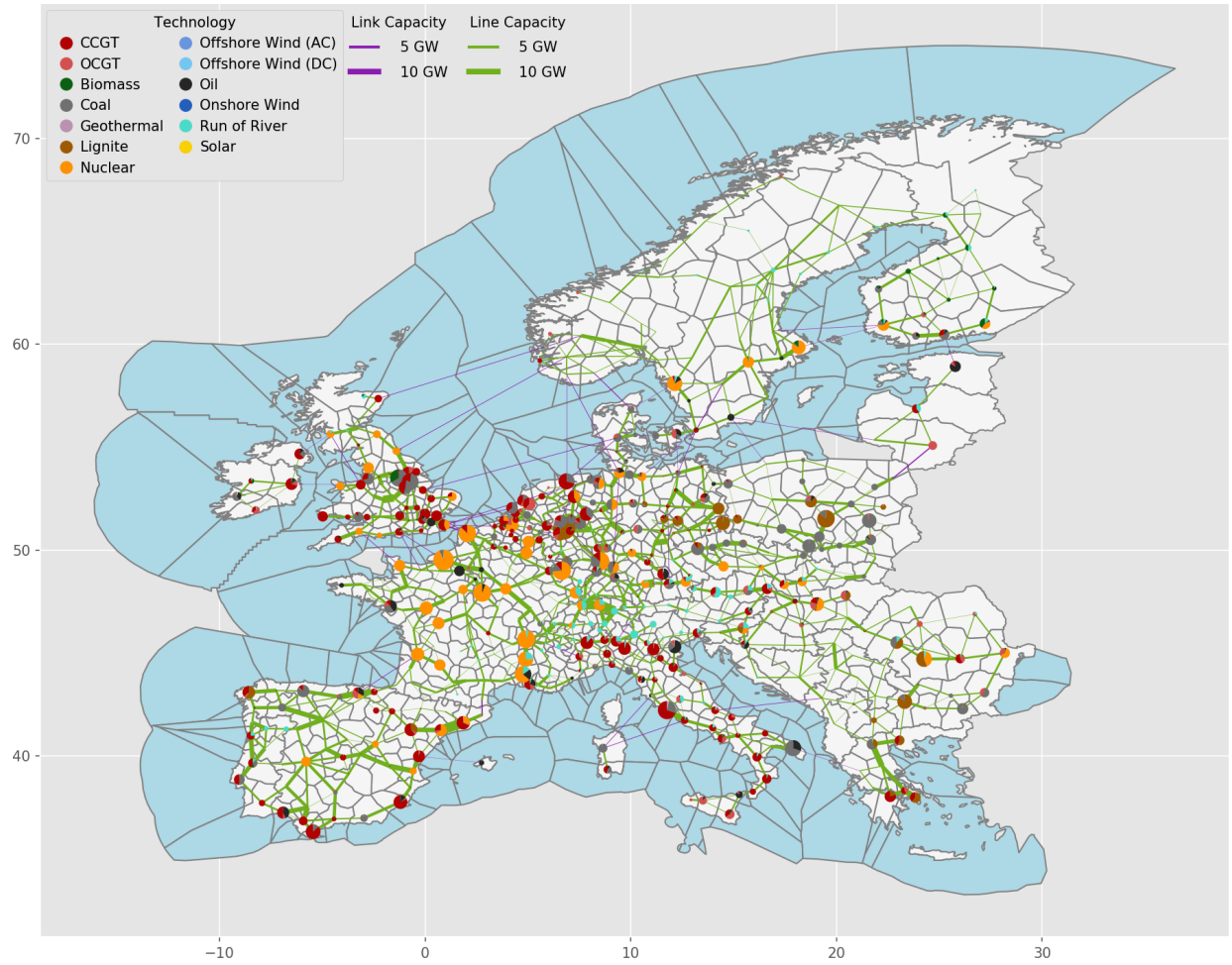
IMPLEMENTATION DETAILS FOR SECTOR-COUPLED SYSTEMS

6.1 Spatial resolution

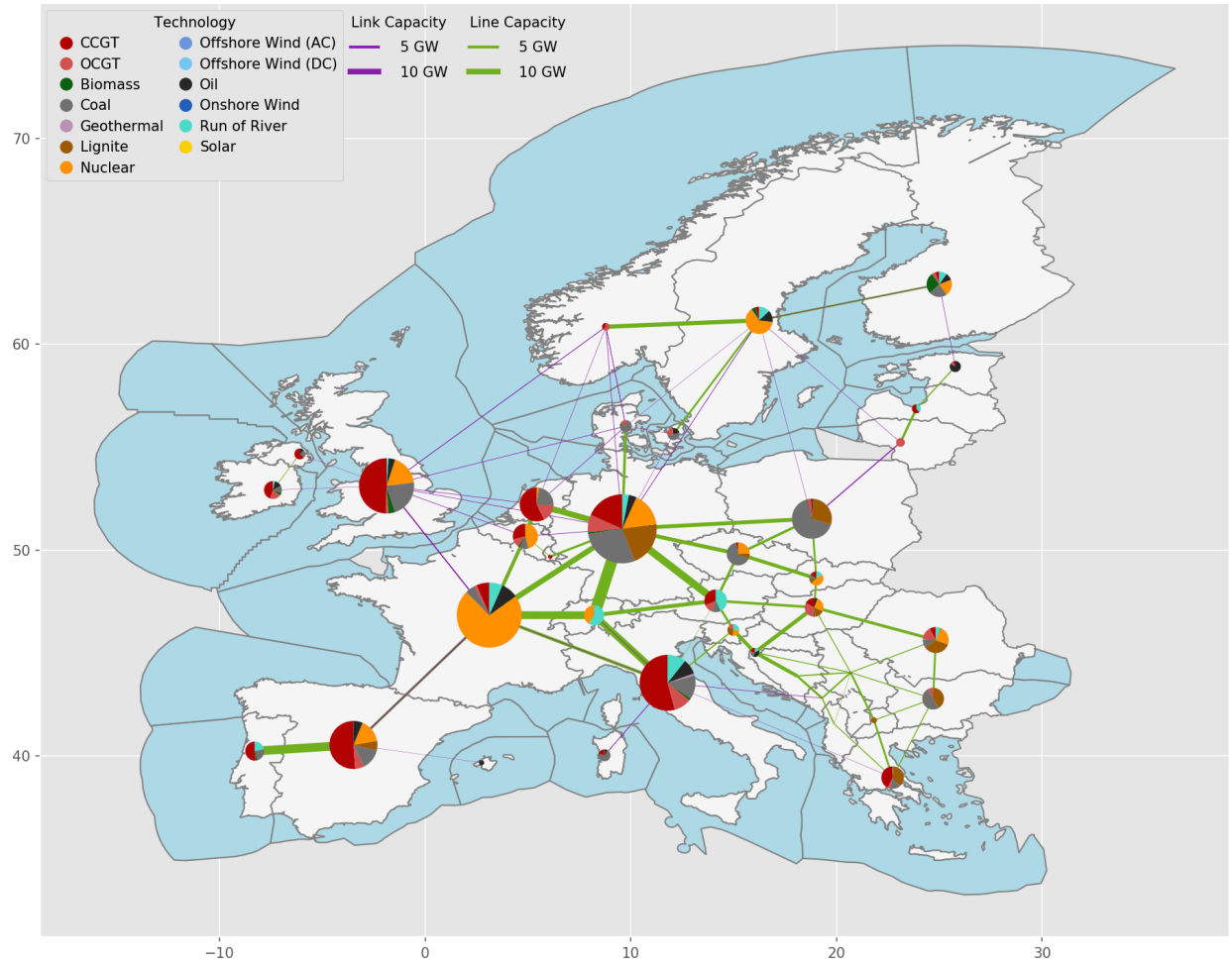
The default nodal resolution of the model follows the electricity generation and transmission model PyPSA-Eur, which clusters down the electricity transmission substations in each European country based on the k-means algorithm (See cluster_network for a complete explanation). This gives nodes which correspond to major load and generation centres (typically cities).

The total number of nodes for Europe is set in the `config/config.yaml` file under `clusters`. The number of nodes can vary between 37, the number of independent countries / synchronous areas, and several hundred. With 200-300 nodes the model needs 100-150 GB RAM to solve with a commercial solver like Gurobi.

Exemplary unsolved network clustered to 512 nodes:



Exemplary unsolved network clustered to 37 nodes:



The total number of nodes for Europe is set in the `config/config.yaml` file under clusters. The number of nodes can vary between 37, the number of independent countries/synchronous areas, and several hundred. With 200-300 nodes, the model needs 100-150 GB RAM to solve with a commercial solver like Gurobi. Not all of the sectors are at the full nodal resolution, and some demand for some sectors is distributed to nodes using heuristics that need to be corrected. Some networks are copper-plated to reduce computational times.

Here are some examples of how spatial resolution is set for different sectors in PyPSA-Eur-Sec:

- Electricity network: Modeled as nodal.
- Electricity residential and commercial demand: Modeled as nodal, distributed in each country based on population and GDP.
- Electricity distribution network: Not included in the model, but a link per node can be used to represent energy transferred between distribution and transmission levels (explained more in detail below).
- Residential and commercial building heating demand: Modeled as nodal, distributed in each country based on population.
- Electricity demand in industry: Modeled as nodal, based on the location of industrial facilities from HotMaps database.
- Industry demand (heat, chemicals, etc.) : Modeled as nodal, distributed in each country based on locations of industry from HotMaps database.
- Hydrogen network: Modeled as nodal (if activated in the config file).

- Methane network: It can be modeled as a single node for Europe or it can be nodally resolved if activated in the config. One node can be considered reasonable since future demand is expected to be low and no bottlenecks are expected. Also, the nodally resolved methane grid is based on SciGRID_gas data.
- Solid biomass: It can be modeled as a single node for Europe or it can be nodally resolved if activated in the config. Nodal modeling includes modeling biomass potential per country (given per country, then distributed by population density within) and the transport of solid biomass between countries.
- CO2: It can be modeled as a single node for Europe or it can be nodally resolved with CO2 transport pipelines if activated in the config. It should mentioned that in single node mode a transport and storage cost is added for sequestered CO2, the cost of which can be adjusted in the config.
- Carbonaceous fuels: Modeled as a single node for Europe by default, since transport costs for liquids are low and no bottlenecks are expected. Can be regionally resolved in configuration.

Electricity distribution network

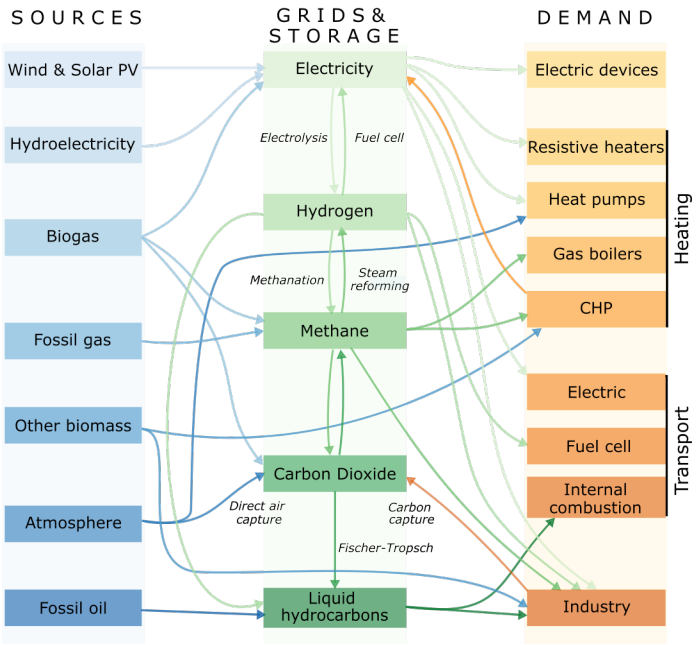
Contrary to the transmission grid, the grid topology at the distribution level (at and below 110 kV) is not included due to the very high computational burden. However, a link per node can be used (if activated in the Config file) to represent energy transferred between distribution and transmission levels at every node. In essence, the total energy capacity connecting the transmission grid and the low-voltage level is optimized. The cost assumptions for this link can be adjusted in Config file options , and is currently assumed to be 500 Eur/kW.

Rooftop PV, heat pumps, resistive heater, home batteries chargers for passenger EVs, as well as individual heating technologies (heat pumps and resistive heaters) are connected to low-voltage level. All the remaining generation and storage technologies are connected to the transmission grid. In practice, this means that the distribution grid capacity is only extended if it is necessary to balance the mismatch between local generation and demand.

6.2 Supply and demand

An initial orientation to the supply and demand options in the model PyPSA-Eur-Sec can be found in the description of the model PyPSA-Eur-Sec-30 in the paper Synergies of sector coupling and transmission reinforcement in a cost-optimised, highly renewable European energy system (2018). The latest version of PyPSA-Eur-Sec differs by including biomass, industry, industrial feedstocks, aviation, shipping, better carbon management, carbon capture and usage/sequestration, and gas networks.

The basic supply (left column) and demand (right column) options in the model are described in this figure:



6.2.1 Electricity supply and demand

Electricity supply and demand follows the electricity generation and transmission model PyPSA-Eur, except that hydrogen storage is integrated into the hydrogen supply, demand and network, and PyPSA-Eur-Sec includes CHPs.

Unlike PyPSA-Eur, PyPSA-Eur-Sec does not distribution electricity demand for industry according to population and GDP, but uses the geographical data from the Hotmaps Industrial Database.

Also unlike PyPSA-Eur, PyPSA-Eur-Sec subtracts existing electrified heating from the existing electricity demand, so that power-to-heat can be optimised separately.

The remaining electricity demand for households and services is distributed inside each country proportional to GDP and population.

6.2.2 Heat demand

Building heating in residential and services sectors is resolved regionally, both for individual buildings and district heating systems, which include different supply options (see *Heat supply*.) Annual heat demands per country are retrieved from JRC-IDEES and split into space and water heating. For space heating, the annual demands are converted to daily values based on the population-weighted Heating Degree Day (HDD) using the atlite tool, where space heat demand is proportional to the difference between the daily average ambient temperature (read from ERA5) and a threshold temperature above which space heat demand is zero. A threshold temperature of 15 °C is assumed by default. The daily space heat demand is distributed to the hours of the day following heat demand profiles from BDEW. These differ for weekdays and weekends/holidays and between residential and services demand.

Space heating

The space heating demand can be exogenously reduced by retrofitting measures that improve the buildings' thermal envelopes.

Co-optimsing of building renovation is also possible, if it is activated in the config file. Renovation of the thermal envelope reduces the space heating demand and is optimised at each node for every heat bus. Renovation measures through additional insulation material and replacement of energy inefficient windows are considered. In a first step, costs per energy savings are estimated in build_retro_cost.py. They depend on the insulation condition of the building

stock and costs for renovation of the building elements. In a second step, for those cost per energy savings two possible renovation strengths are determined: a moderate renovation with lower costs, a lower maximum possible space heat savings, and an ambitious renovation with associated higher costs and higher efficiency gains. They are added by stepwise linearisation in form of two additional generations in prepare_sector_network.py. Further information are given in the publication : Mitigating heat demand peaks in buildings in a highly renewable European energy system, (2021).

Water heating

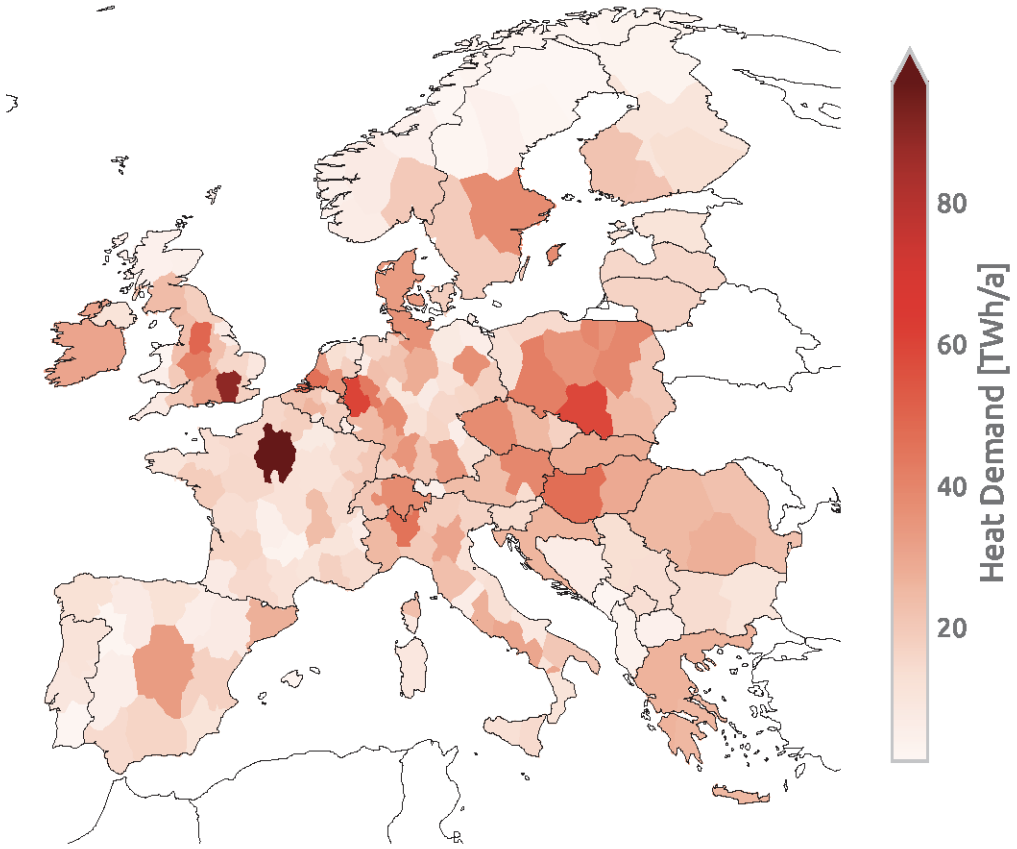
Hot water demand is assumed to be constant throughout the year.

Urban and rural heating

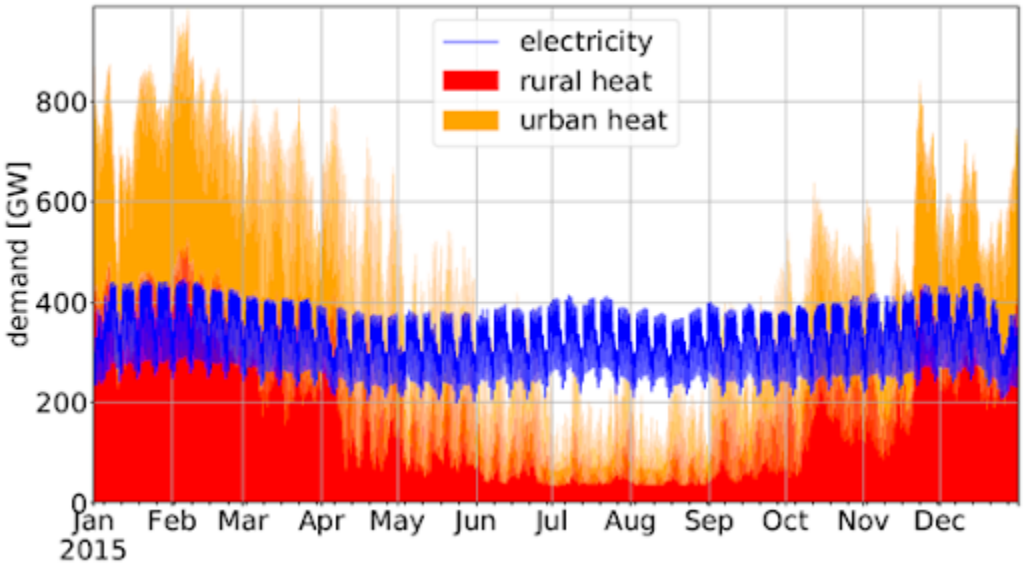
For every country, heat demand is split between low and high population density areas. These country-level totals are then distributed to each region in proportion to their rural and urban populations respectively. Urban areas with dense heat demand can be supplied with large-scale district heating systems. The percentage of urban heat demand that can be supplied by district heating networks as well as lump-sum losses in district heating systems is exogenously determined in the config file.

Cooling demand

Cooling is electrified and is included in the electricity demand. Cooling demand is assumed to remain at current levels. An example of regional distribution of the total heat demand for network 181 regions is depicted below.



As below figure shows, the current total heat demand in Europe is similar to the total electricity demand but features much more pronounced seasonal variations. The current total building heating demand in Europe adds up to 3084 TWh/a of which 78% occurs in urban areas.



In practice, in PyPSA-Eur-Sec, there are heat demand buses to which the corresponding heat demands are added.

1) Urban central heat: large-scale district heating networks in urban areas with dense heat population. Residential and services demand in these areas are added as demands to this bus
2) Residential urban decentral heat: heating for residential buildings in urban areas not using district heating
3) Services urban decentral heat: heating for services buildings in urban areas not using district heating
4) Residential rural heat: heating for residential buildings in rural areas with low population density.
5) Services rural heat: heating for residential services buildings in rural areas with low population density. Heat demand from agriculture sector is also included here.

6.2.3 Heat supply

Different supply options are available depending on whether demand is met centrally through district heating systems, or decentrally through appliances in individual buildings.

Urban central heat

For large-scale district heating systems the following options are available: combined heat and power (CHP) plants consuming gas or biomass from waste and residues with and without carbon capture (CC), large-scale air-sourced heat pumps, gas and oil boilers, resistive heaters, and fuel cell CHPs. Additionally, waste heat from the Fischer-Tropsch and Sabatier processes for the production of synthetic hydrocarbons can supply district heating systems. For more detailed explanation of these processes, see *Oil-based products supply* and *Methane supply*.

Residential and Urban decentral heat

Supply options in individual buildings include gas and oil boilers, air- and ground-sourced heat pumps, resistive heaters, and solar thermal collectors. Ground-source heat pumps are only allowed in rural areas because of space constraints. Thus, only air- source heat pumps are allowed in urban areas. This is a conservative assumption, since there are many possible sources of low-temperature heat that could be tapped in cities (e.g. waste water, ground water, or natural bodies of water). Costs, lifetimes and efficiencies for these technologies are retrieved from the technology-data repository.

Below are more detailed explanations for each heating supply component, all of which are modelled as links in PyPSA-Eur-Sec.

Large-scale CHP

Large Combined Heat and Power plants are included in the model if it is specified in the config file.

CHPs are based on back pressure plants operating with a fixed ratio of electricity to heat output. The efficiencies of each are given on the back pressure line, where the back pressure coefficient cb is the electricity output divided by the heat output. (For a more complete explanation of the operation of CHPs refer to the study by Dahl et al. : Cost sensitivity of optimal sector-coupled district heating production systems.

PyPSA-Eur-Sec includes CHP plants fueled by methane and solid biomass from waste and residues. Hydrogen fuel cells also produce both electricity and heat.

The methane CHP is modeled on the Danish Energy Agency (DEA) "Gas turbine simple cycle (large)" while the solid biomass CHP is based on the DEA's "09b Wood Pellets Medium". For biomass CHP, cb = 0.46 , whereas for gas CHP, cb = 1.

NB: The old PyPSA-Eur-Sec-30 model assumed an extraction plant (like the DEA coal CHP) for gas which has flexible production of heat and electricity within the feasibility diagram of Figure 4 in the study by Brown et al. We have switched to the DEA back pressure plants since these are more common for smaller plants for biomass, and because the extraction plants were on the back pressure line for 99.5% of the time anyway. The plants were all changed to back pressure in PyPSA-Eur-Sec v0.4.0.

Micro-CHP

PyPSA-Eur-Sec allows individual buildings to make use of micro gas CHPs that are assumed to be installed at the distribution grid level.

Heat pumps

The coefficient of performance (COP) of air- and ground-sourced heat pumps depends on the ambient or soil temperature respectively. Hence, the COP is a time-varying parameter (refer to Config file). Generally, the COP will be lower during winter when temperatures are low. Because the ambient temperature is more volatile than the soil temperature, the COP of ground-sourced heat pumps is less variable. Moreover, the COP depends on the difference between the source and sink temperatures:

$$\Delta T = T_{sink} - T_{source}$$

For the sink water temperature Tsink we assume 55 °C [Config file]. For the time- and location-dependent source temperatures Tsource, we rely on the ERA5 reanalysis weather data. The temperature differences are converted into COP time series using results from a regression analysis performed in the study by Stafell et al.. For air-sourced heat pumps (ASHP), we use the function:

$$COP(\Delta T) = 6.81 - 0.121\Delta T + 0.000630\Delta T^2$$

for ground-sourced heat pumps (GSHP), we use the function:

$$COP(\Delta T) = 8.77 - 0.150\Delta T + 0.000734\Delta T^2$$

Resistive heaters

Can be activated in Config from the boilers option. Resistive heaters produce heat with a fixed conversion efficiency (refer to Technology-data repository).

Gas, oil, and biomass boilers

Can be activated in Config from the boilers , oil boilers , and biomass boiler option. Similar to resistive heaters, boilers have a fixed efficiency and produce heat using gas, oil or biomass.

Solar thermal collectors

Can be activated in the config file from the solar_thermal option. Solar thermal profiles are built based on weather data and also have the options for setting the sky model and the orientation of the panel in the config file, which are then used by the atlite tool to calculate the solar resource time series.

Waste heat from Fuel Cells, Methanation and Fischer-Tropsch plants

Waste heat from fuel cells in addition to processes like Fischer-Tropsch, methanation, and Direct Air Capture (DAC) is dumped into district heating networks.

Existing heating capacities and decommissioning

For the myopic transition paths, capacities already existing for technologies supplying heat are retrieved from "Mapping and analyses of the current and future (2020 - 2030)" . For the sake of simplicity, coal, oil and gas boiler capacities are assimilated to gas boilers. Besides that, existing capacities for heat resistors, air-sourced and ground-sourced heat pumps are included in the model. For heating capacities, 25% of existing capacities in 2015 are assumed to be decommissioned in every 5-year time step after 2020.

Thermal Energy Storage

Activated in Config from the tes option.

Thermal energy can be stored in large water pits associated with district heating systems and individual thermal energy storage (TES), i.e., small water tanks. Water tanks are modelled as stores. A thermal energy density of 46.8 kWh $_{th}$/m3 is assumed, corresponding to a temperature difference of 40 K. The decay of thermal energy in the stores: 1- $e^{-1/24t}$ is assumed to have a time constant of τ=180 days for central TES and τ=3 days for individual TES, both modifiable through tes_tau in config file. Charging and discharging efficiencies are 90% due to pipe losses.

Retrofitting of the thermal envelope of buildings

Co-optimising building renovation is only enabled if in the config file. To reduce the computational burden, default setting is set as false.

Renovation of the thermal envelope reduces the space heating demand and is optimised at each node for every heat bus. Renovation measures through additional insulation material and replacement of energy inefficient windows are considered.

In a first step, costs per energy savings are estimated in the build_retro_cost.py script. They depend on the insulation condition of the building stock and costs for renovation of the building elements. In a second step, for those cost per energy savings two possible renovation strengths are determined: a moderate renovation with lower costs and lower maximum possible space heat savings, and an ambitious renovation with associated higher costs and higher efficiency gains. They are added by step-wise linearisation in form of two additional generations in the prepare_sector_network.py script.

Settings in the `config/config.yaml` concerning the endogenously optimisation of building renovation include cost factor, interest rate, annualised cost, tax weighting, and construction index.

Further information are given in the study by Zeyen et al. : Mitigating heat demand peaks in buildings in a highly renewable European energy system, (2021).

6.2.4 Hydrogen demand

Hydrogen is consumed in the industry sector (see *Industry demand*) to produce ammonia (see *Chemicals Industry*) and direct reduced iron (DRI) (see *Iron and Steel*). Hydrogen is also consumed to produce synthetic methane (see *Methane supply*) and liquid hydrocarbons (see *Oil-based products supply*) which have multiple uses in industry and other sectors. Hydrogen is also used for transport applications (see *Transportation*), where it is exogenously fixed. It is used in heavy-duty land transport and as liquified hydrogen in the shipping sector (see *Shipping*). Furthermore, stationary fuel cells may re-electrify hydrogen (with waste heat as a byproduct) to balance renewable fluctuations (see *Electricity supply and demand*). The waste heat from the stationary fuel cells can be used in district-heating systems.

6.2.5 Hydrogen supply

Today, most of the H_2 consumed globally is produced from natural gas by steam methane reforming (SMR)

$$CH_4 + H_2O \rightarrow CO + 3H_2$$

combined with a water-gas shift reaction

$$CO + H_2O \rightarrow CO_2 + H_2$$

SMR is included here. PyPSA-Eur-Sec allows this route of H_2 production with and without [carbon capture (CC)] (see *Carbon dioxide capture, usage and sequestration (CCU/S)*). These routes are often referred to as blue and grey hydrogen. Here, methane input can be both of fossil or synthetic origin.

Green hydrogen can be produced by electrolysis to split water into hydrogen and oxygen

$$2H_2O \rightarrow 2H_2 + O_2$$

For the electrolysis, alkaline electrolysers are chosen since they have lower cost and higher cumulative installed capacity than polymer electrolyte membrane (PEM) electrolysers. The techno-economic assumptions are taken from the technology-data repository. Waste heat from electrolysis is not leveraged in the model.

Transport

Hydrogen is transported by pipelines. H_2 pipelines are endogenously generated, either via a greenfield H_2 network, or by retrofitting natural gas pipelines). Retrofitting is implemented in such a way that for every unit of decommissioned gas pipeline, a share (60% is used in the study by Neumann et al.) of its nominal capacity (exogenously determined in the config file.) is available for hydrogen transport. When the gas network is not resolved, this input denotes the potential for gas pipelines repurposed into hydrogen pipelines. New pipelines can be built additionally on all routes where there currently is a gas or electricity network connection. These new pipelines will be built where no sufficient retrofitting options are available. The capacities of new and repurposed pipelines are a result of the optimisation.

Storage

Hydrogen can be stored in overground steel tanks or underground salt caverns. For the latter, energy storage capacities in every country are limited to the potential estimation for onshore salt caverns within 50 km of shore to avoid environmental issues associated with brine solution disposal. Underground storage potentials for hydrogen in European salt caverns is acquired from Caglayan et al.

6.2.6 Methane demand

Methane is used in individual and large-scale gas boilers, in CHP plants with and without carbon capture, in OCGT and CCGT power plants, and in some industry subsectors for the provision of high temperature heat (see *Industry demand*). Methane is not used in the transport sector because of engine slippage.

6.2.7 Methane supply

In addition to methane from fossil origins, the model also considers biogenic and synthetic sources. The gas network can either be modelled, or it can be assumed that gas transport is not limited. If gas infrastructure is regionally resolved, fossil gas can enter the system only at existing and planned LNG terminals, pipeline entry-points, and intra- European gas extraction sites, which are retrieved from the SciGRID Gas IGGIELGN dataset and the GEM Wiki. Biogas can be upgraded to methane. Synthetic methane can be produced by processing hydrogen and captures CO_2 in the Sabatier reaction

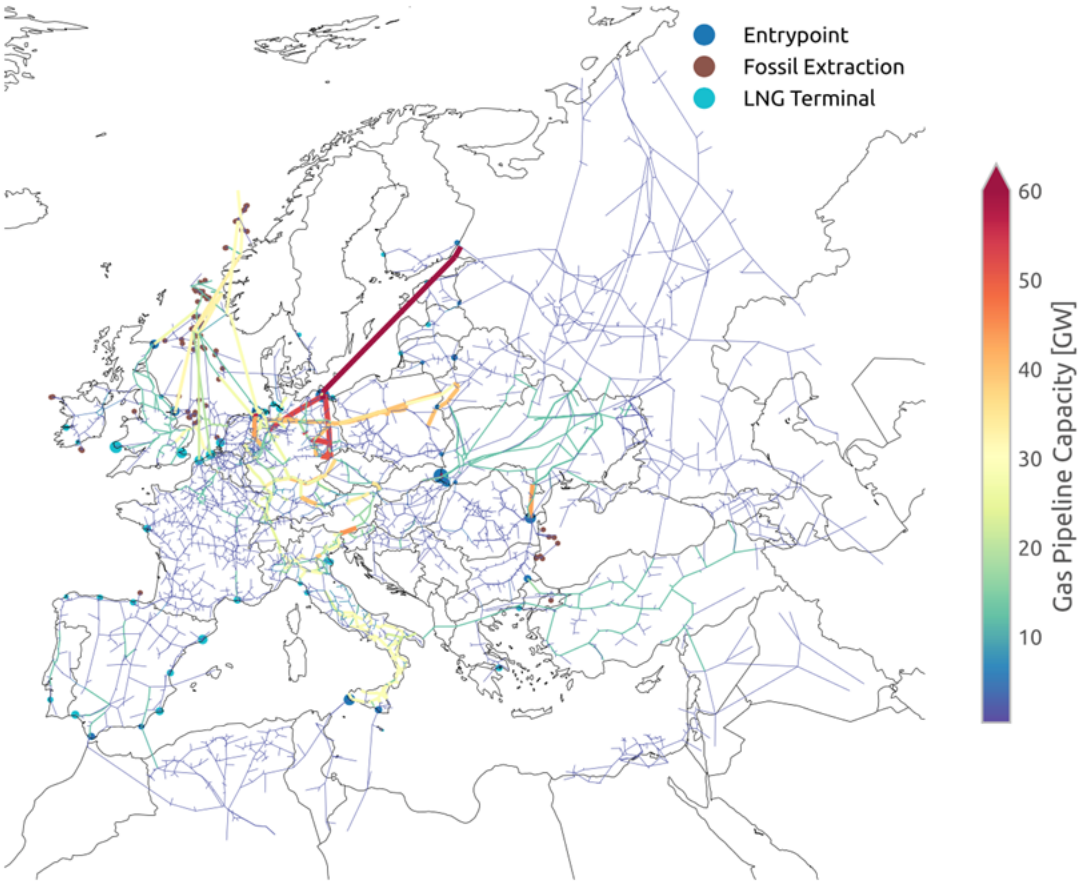
$$CO_2 + 4H_2 \rightarrow CH_4 + 2H_2O$$

Direct power-to-methane conversion with efficient heat integration developed in the HELMETH project is also an option. The share of synthetic, biogenic and fossil methane is an optimisation result depending on the techno-economic assumptions.

Methane transport

The existing European gas transmission network is represented based on the SciGRID Gas IGGIELGN dataset. This dataset is based on compiled and merged data from the ENTSOG maps and other publicly available data sources. It includes data on the capacity, diameter, pressure, length, and directionality of pipelines. Missing capacity data is conservatively inferred from the pipe diameter following conversion factors derived from an EHB report. The gas network is clustered to the selected number of model regions. Gas pipelines can be endogenously expanded or repurposed for hydrogen transport. Gas flows are represented by a lossless transport model. Methane is assumed to be transmitted without cost or capacity constraints because future demand is predicted to be low compared to available transport capacities.

The following figure shows the unclustered European gas transmission network based on the SciGRID Gas IGGIELGN dataset. Pipelines are color-coded by estimated capacities. Markers indicate entry-points, sites of fossil resource extraction, and LNG terminals.



6.2.8 Biomass Supply

Biomass supply potentials for each European country are taken from the JRC ENSPRESO database where data is available for various years (2010, 2020, 2030, 2040 and 2050) and scenarios (low, medium, high). No biomass import from outside Europe is assumed. More information on the data set can be found here.

6.2.9 Biomass demand

Biomass supply potentials for every NUTS2 region are taken from the JRC ENSPRESO database where data is available for various years (2010, 2020, 2030, 2040 and 2050) and different availability scenarios (low, medium, high). No biomass import from outside Europe is assumed. More information on the data set can be found here. The data for NUTS2 regions is mapped to PyPSA-Eur-Sec model regions in proportion to the area overlap.

The desired scenario can be selected in the PyPSA-Eur-Sec configuration. The script for building the biomass potentials from the JRC ENSPRESO data base is located here. Consult the script to see the keywords that specify the scenario options.

The configuration also allows the user to define how the various types of biomass are used in the model by using the following categories: biogas, solid biomass, and not included. Feedstocks categorized as biogas, typically manure and sludge waste, are available to the model as biogas, which can be upgraded to biomethane. Feedstocks categorized as solid biomass, e.g. secondary forest residues or municipal waste, are available for combustion in combined-heat-and power (CHP) plants and for medium temperature heat (below 500 °C) applications in industry. It can also converted to gas or liquid fuels.

Feedstocks labeled as not included are ignored by the model.

A typical use case for biomass would be the medium availability scenario for 2030 where only residues from agriculture and forestry as well as biodegradable municipal waste are considered as energy feedstocks. Fuel crops are avoided because they compete with scarce land for food production, while primary wood, as well as wood chips and pellets, are avoided because of concerns about sustainability. See the supporting materials of the paper for more details.

Solid biomass conversion and use

Solid biomass can be used directly to provide process heat up to 500˚C in the industry. It can also be burned in CHP plants and boilers associated with heating systems. These technologies are described elsewhere (see *Large-scale CHP* and *Industry demand*).

Solid biomass can be converted to syngas if the option is enabled in the config file. In this case the model will enable the technology BioSNG both with and without the option for carbon capture (see Technology-data repository).

Liquefaction of solid biomass can be enabled allowing the model to convert it into liquid hydrocarbons that can replace conventional oil products. This technology also comes with and without carbon capture (see Technology-data repository).

Transport of solid biomass

The transport of solid biomass can either be assumed unlimited between countries or it can be associated with a country specific cost per MWh/km. In the config file these options are toggled here. If the option is off, use of solid biomass is transport. If it is turned on, a biomass transport network will be created between all nodes. This network resembles road transport of biomass and the cost of transportation is a variable cost which is proportional to distance and a country specific cost per MWh/km. The latter is estimated from the country specific costs per ton/km used in the publication "The JRC-EU-TIMES model. Bioenergy potentials for EU and neighbouring countries".

Biogas transport and use

Biogas will be aggregated into a common European resources if a gas network is not modelled explicitly, i.e., the gas_network option is set to false. If, on the other hand, a gas network is included, the biogas potential will be associated with each node of origin. The model can only use biogas by first upgrading it to natural gas quality [see *Methane supply*] (bio methane) which is fed into the general gas network.

6.2.10 Oil-based products demand

Naphtha is used as a feedstock in the chemicals industry (see *Chemicals Industry*). Furthermore, kerosene is used as transport fuel in the aviation sector (see *Aviation*). Non-electrified agriculture machinery also consumes gasoline. Land transport [(see *Land transport*) that is not electrified or converted into using H_2-fuel cells also consumes oil-based products. While there is regional distribution of demand, the carrier is copperplated in the model, which means that transport costs and constraints are neglected.

6.2.11 Oil-based products supply

Oil-based products can be either of fossil origin or synthetically produced by combining H_2 (see *Hydrogen supply*) and captured CO_2 (see *Carbon dioxide capture, usage and sequestration (CCU/S)*) in Fischer-Tropsch plants

$$nCO + (2n+1)H_2ßC_nH_{2n+2} + nH_2O$$

with costs as included from the technology-data repository. The waste heat from the Fischer-Tropsch process is supplied to district heating networks. The share of fossil and synthetic oil is an optimisation result depending on the techno-economic assumptions.

Oil-based transport

Liquid hydrocarbons are assumed to be transported freely among the model region since future demand is predicted to be low, transport costs for liquids are low and no bottlenecks are expected.

6.2.12 Industry demand

Industry demand is split into a dozen different sectors with specific energy demands, process emissions of carbon dioxide, as well as existing and prospective mitigation strategies.

The Subsection overview below provides a general description of the modelling approach for the industry sector. The following subsections describe the current energy demands, available mitigation strategies, and whether mitigation is exogenously fixed or co-optimised with the other components of the model for each industry subsector in more detail. See details for Iron and Steel (see *Iron and Steel*), Chemicals Industry and Ammonia (see *Chemicals Industry*), Non-metallic Mineral products , Non-ferrous Metals , and other Industry Subsectors.

Overview

Greenhouse gas emissions associated with industry can be classified into energy-related and process-related emissions. Today, fossil fuels are used for process heat energy in the chemicals industry, but also as a non-energy feedstock for chemicals like ammonia (NH_3), ethylene (C_2H_4) and methanol (CH_3OH). Energy-related emissions can be curbed by using low-emission energy sources. The only option to reduce process-related emissions is by using an alternative manufacturing process or by assuming a certain rate of recycling so that a lower amount of virgin material is needed.

The overarching modelling procedure can be described as follows. First, the energy demands and process emissions for every unit of material output are estimated based on data from the JRC-IDEES database and the fuel and process switching described in the subsequent sections. Second, the 2050 energy demands and process emissions are calculated using the per-unit-of-material ratios based on the industry transformations and the country-level material production in 2015, assuming constant material demand.

Missing or too coarsely aggregated data in the JRC-IDEES database is supplemented with additional datasets: Eurostat energy balances, United States, Geological Survey for ammonia production, DECHEMA for methanol and chlorine, and national statistics from Switzerland.

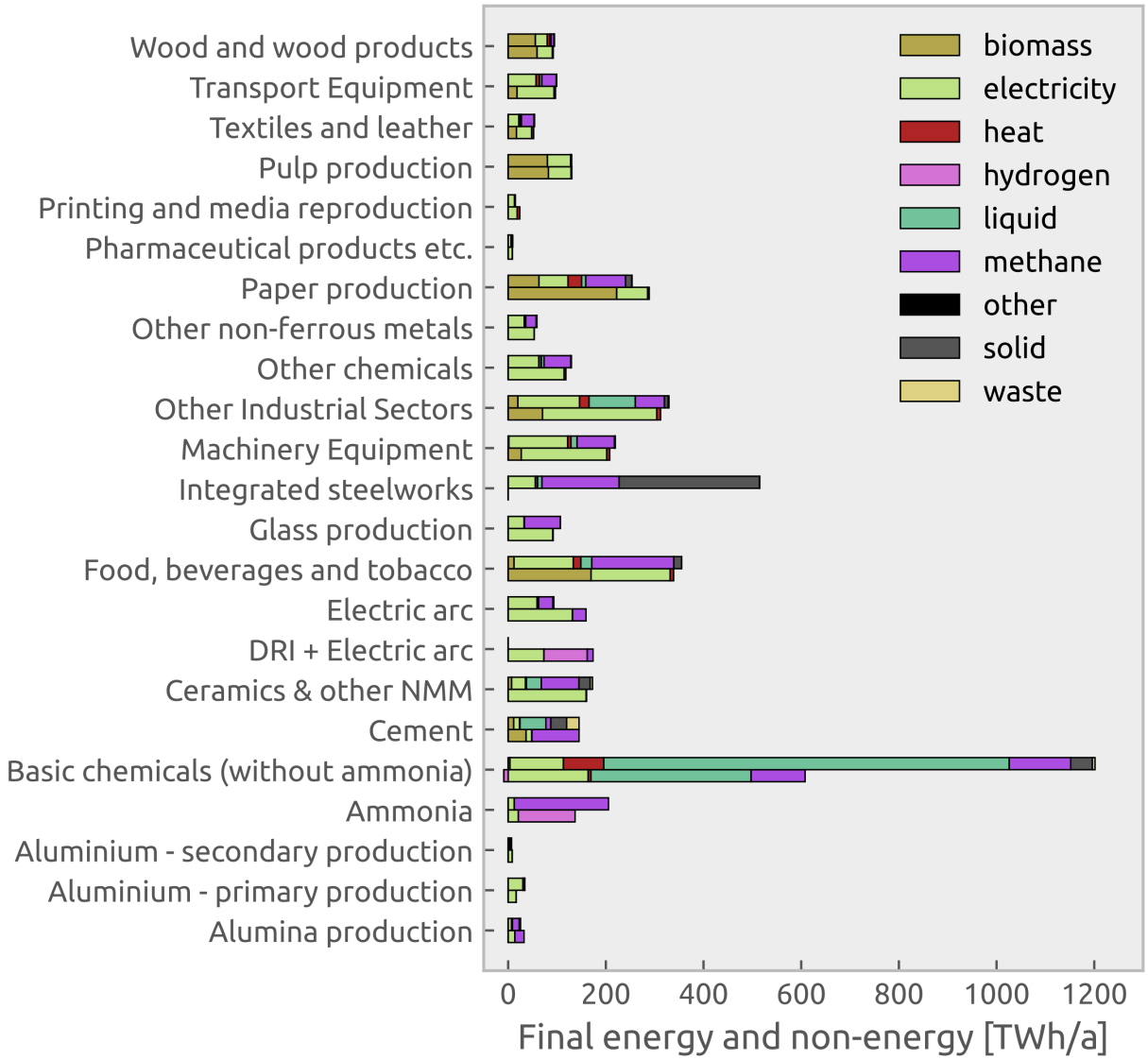
Where there are fossil and electrified alternatives for the same process (e.g. in glass manufacture or drying), we assume that the process is completely electrified. Current electricity demands (lighting, air compressors, motor drives, fans, pumps) will remain electric. Processes that require temperatures below 500 °C are supplied with solid biomass, since we

assume that residues and wastes are not suitable for high-temperature applications. We see solid biomass use primarily in the pulp and paper industry, where it is already widespread, and in food, beverages and tobacco, where it replaces natural gas. Industries which require high temperatures (above 500 °C), such as metals, chemicals and non-metallic minerals are either electrified where suitable processes already exist, or the heat is provided with synthetic methane.

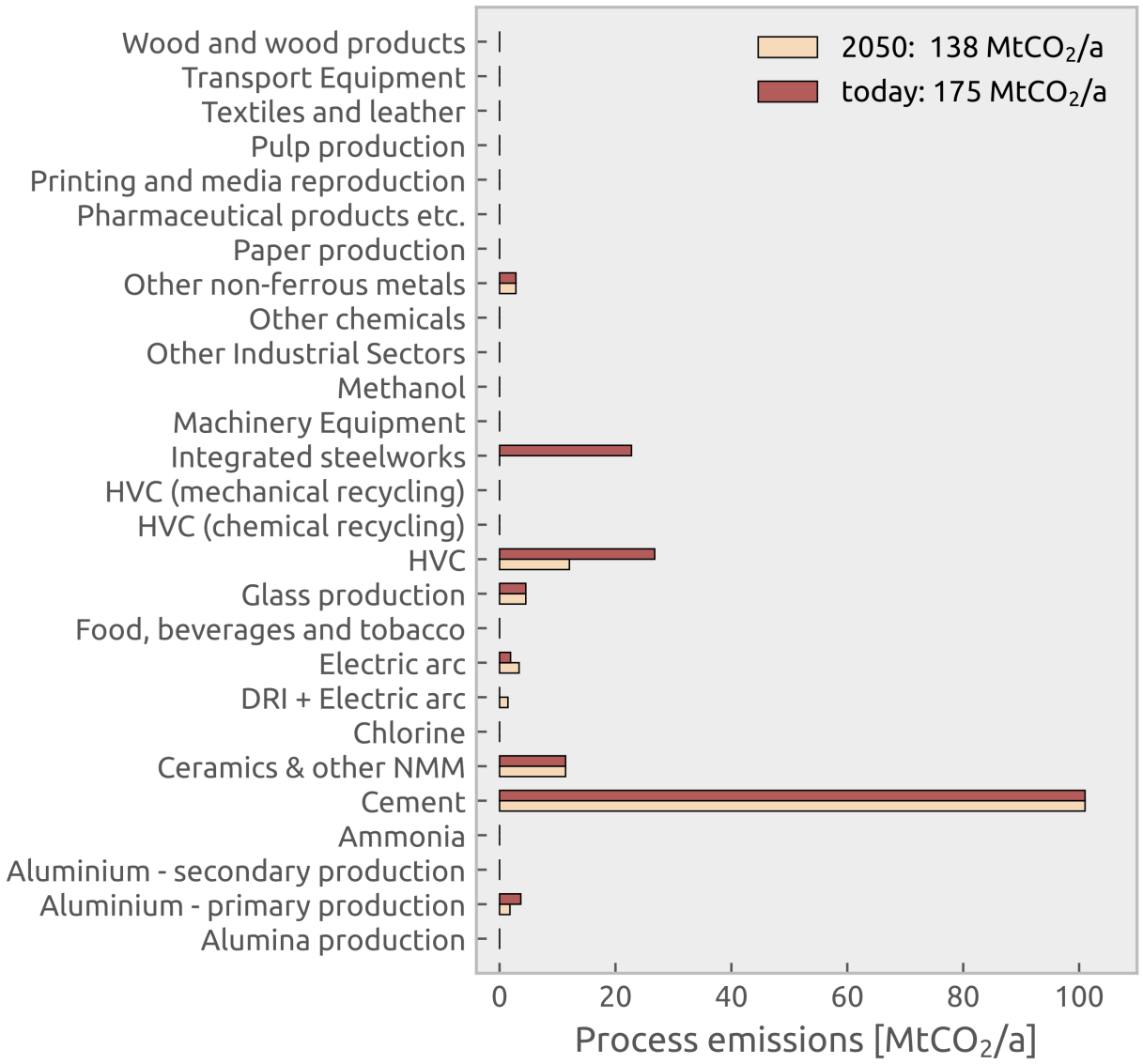
Hydrogen for high-temperature process heat is not part of the model currently.

Where process heat is required, our approach depends on the necessary temperature. For example, due to the high share of high-temperature process heat demand (see Naegler et al. and Rehfeldt el al.), we disregard geothermal and solar thermal energy as sources for process heat since they cannot attain high-temperature heat.

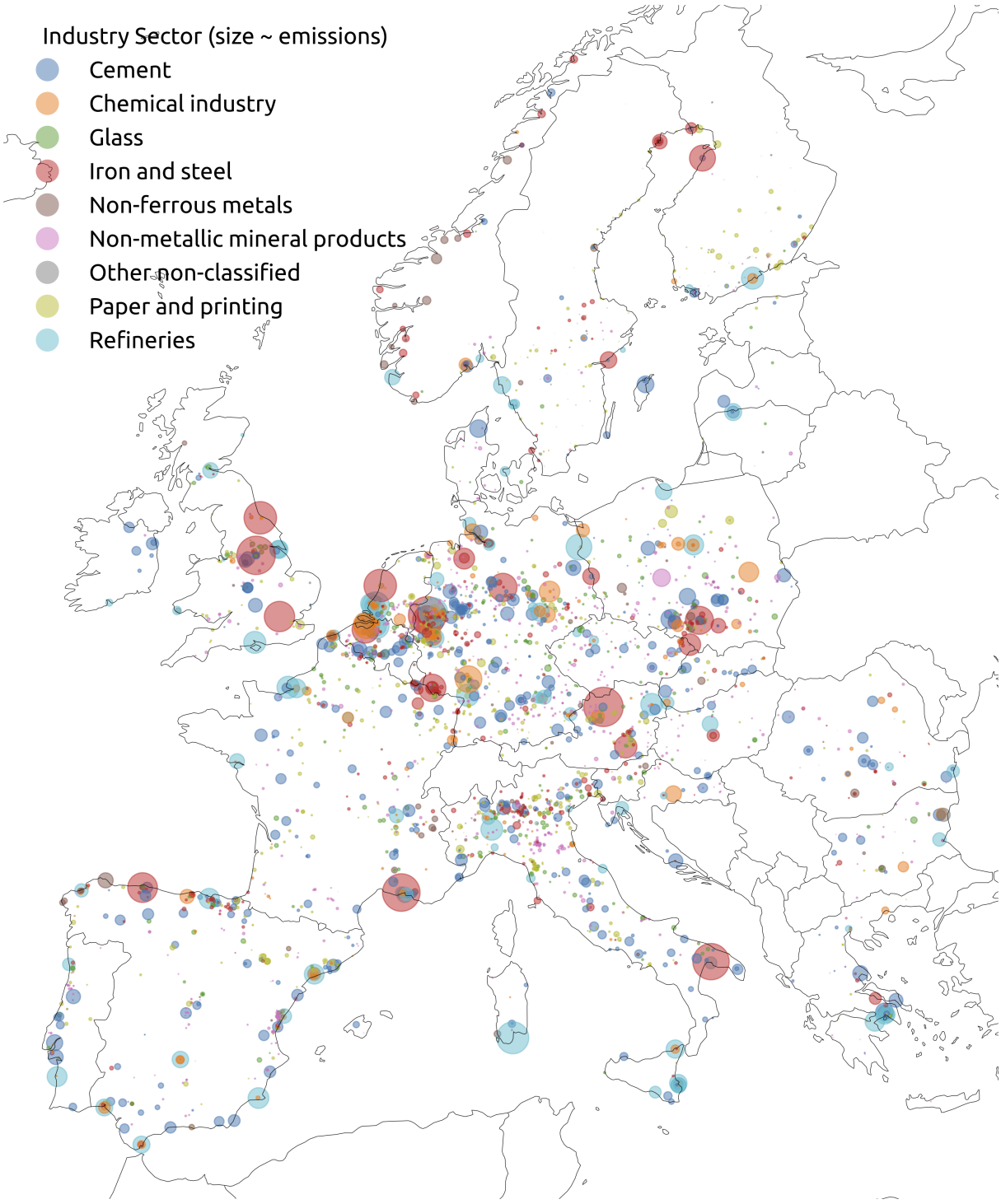
The following figure shows the final consumption of energy and non-energy feedstocks in industry today in comparison to the scenario in 2050 assumed in Neumann et al.



The following figure shows the process emissions in industry today (top bar) and in 2050 without carbon capture (bottom bar) assumed in Neumann et al.



Inside each country the industrial demand is then distributed using the Hotmaps Industrial Database, which is illustrated in the figure below. This open database includes georeferenced industrial sites of energy-intensive industry sectors in EU28, including cement, basic chemicals, glass, iron and steel, non-ferrous metals, non-metallic minerals, paper, and refineries subsectors. The use of this spatial dataset enables the calculation of regional and process-specific energy demands. This approach assumes that there will be no significant migration of energy-intensive industries.



Iron and Steel

Two alternative routes are used today to manufacture steel in Europe. The primary route (integrated steelworks) represents 60% of steel production, while the secondary route (electric arc furnaces, EAF), represents the other 40% (Lechtenböhmer et. al).

The primary route uses blast furnaces in which coke is used to reduce iron ore into molten iron, which is then converted

into steel:

$$CO_2 + C \rightarrow 2CO$$

$$3Fe_2O_3 + CO \rightarrow 2Fe_3O_4 + CO$$

$$Fe_3O_4 + CO \rightarrow 3FeO + CO_2$$

$$FeO + CO \rightarrow Fe + CO_2$$

The primary route of steelmaking implies large process emissions of 0.22 t $_{CO_2}$ /t of steel, amounting to 7% of global greenhouse gas emissions (Vogl et. al).

In the secondary route, electric arc furnaces are used to melt scrap metal. This limits the CO_2 emissions to the burning of graphite electrodes (Friedrichsen et. al), and reduces process emissions to 0.03 t $_{CO_2}$ /t of steel.

We assume that the primary route can be replaced by a third route in 2050, using direct reduced iron (DRI) and subsequent processing in an EAF.

$$3Fe_2O_3 + H_2 \rightarrow 2Fe_3O_4 + H_2O$$

$$Fe_3O_4 + H_2 \rightarrow 3FeO + H_2O$$

$$FeO + H_2 \rightarrow Fe + H_2O$$

This circumvents the process emissions associated with the use of coke. For hydrogen- based DRI, we assume energy requirements of 1.7 MWh $_{H_2}$ /t steel (Vogl et. al) and 0.322 MWh $_{el}$/t steel (HYBRIT 2016).

The share of steel produced via the primary route is exogenously set in the config file. The share of steel obtained via hydrogen-based DRI plus EAF is also set exogenously in the config file. The remaining share is manufactured through the secondary route using scrap metal in EAF. Bioenergy as alternative to coke in blast furnaces is not considered in the model (Mandova et.al, Suopajärvi et.al).

For the remaining subprocesses in this sector, the following transformations are assumed. Methane is used as energy source for the smelting process. Activities associated with furnaces, refining and rolling, and product finishing are electrified assuming the current efficiency values for these cases. These transformations result in changes in process emissions as outlined in the process emissions figure presented in the industry overview section (see *Overview*).

Chemicals Industry

The chemicals industry includes a wide range of diverse industries, including the production of basic organic compounds (olefins, alcohols, aromatics), basic inorganic compounds (ammonia, chlorine), polymers (plastics), and end-user products (cosmetics, pharmaceutics).

The chemicals industry consumes large amounts of fossil-fuel based feedstocks (see Levi et. al), which can also be produced from renewables as outlined for hydrogen (see *Hydrogen supply*), for methane (see *Methane supply*), and for oil-based products (see *Oil-based products supply*). The ratio between synthetic and fossil-based fuels used in the industry is an endogenous result of the optimisation.

The basic chemicals consumption data from the JRC IDEES database comprises high- value chemicals (ethylene, propylene and BTX), chlorine, methanol and ammonia. However, it is necessary to separate out these chemicals because their current and future production routes are different.

Statistics for the production of ammonia, which is commonly used as a fertilizer, are taken from the USGS for every country. Ammonia can be made from hydrogen and nitrogen using the Haber-Bosch process.

$$N_2 + 3H_2 \rightarrow 2NH_3$$

The Haber-Bosch process is not explicitly represented in the model, such that demand for ammonia enters the model as a demand for hydrogen (6.5 MWh $_{H_2}$ / t $_{NH_3}$) and electricity (1.17 MWh $_{el}$ /t $_{NH_3}$) (see Wang et. al). Today,

natural gas dominates in Europe as the source for the hydrogen used in the Haber-Bosch process, but the model can choose among the various hydrogen supply options described in the hydrogen section (see *Hydrogen supply*)

The total production and specific energy consumption of chlorine and methanol is taken from a DECHEMA report. According to this source, the production of chlorine amounts to 9.58 MtCl/a, which is assumed to require electricity at 3.6 MWh $_{el}$/t of chlorine and yield hydrogen at 0.937 MWh $_{H_2}$/t of chlorine in the chloralkali process. The production of methanol adds up to 1.5 MtMeOH/a. Low-carbon methanol production (or methanolisation) by hydrogenation of CO_2 requires hydrogen at 6.299 MWh $_{H_2}$/t of methanol, carbon dioxide at 1.373 t $_{CO_2}$/t of methanol and electricity at 1.5 MWh $_{el}$/t of methanol. The energy content of methanol is 5.528 MWh $_{MeOH}$/t of methanol. These values are set exogenously in the config file.

The production of ammonia, methanol, and chlorine production is deducted from the JRC IDEES basic chemicals, leaving the production totals of high-value chemicals. For this, we assume that the liquid hydrocarbon feedstock comes from synthetic or fossil- origin naphtha (14 MWh $_{naphtha}$/t of HVC, similar to Lechtenböhmer et al), ignoring the methanol-to-olefin route. Furthermore, we assume the following transformations of the energy-consuming processes in the production of plastics: the final energy consumption in steam processing is converted to methane since requires temperature above 500 °C (4.1 MWh $_{CH_4}$ /t of HVC, see Rehfeldt et al.); and the remaining processes are electrified using the current efficiency of microwave for high-enthalpy heat processing, electric furnaces, electric process cooling and electric generic processes (2.85 MWh $_{el}$/t of HVC).

The process emissions from feedstock in the chemical industry are as high as 0.369 t $_{CO_2}$/t of ethylene equivalent. We consider process emissions for all the material output, which is a conservative approach since it assumes that all plastic-embedded CO_2 will eventually be released into the atmosphere. However, plastic disposal in landfilling will avoid, or at least delay, associated CO_2 emissions.

Circular economy practices drastically reduce the amount of primary feedstock needed for the production of plastics in the model (see Kullmann et al., Meys et al. (2021), Meys et al. (2020), Gu et al.) and consequently, also the energy demands and level of process emission. The percentage of plastics that are assumed to be mechanically recycled can be selected in the config file, as well as the percentage that is chemically recycled, see config file The energy consumption for those recycling processes are respectively 0.547 MWh $_{el}$/t of HVC (as indicated in the config file) (Meys et al. (2020)), and 6.9 MWh $_{el}$/t of HVC (as indicated in the config file) based on pyrolysis and electric steam cracking (see Materials Economics report).

Non-metallic Mineral Products

This subsector includes the manufacturing of cement, ceramics, and glass.

Cement

Cement is used in construction to make concrete. The production of cement involves high energy consumption and large process emissions. The calcination of limestone to chemically reactive calcium oxide, also known as lime, involves process emissions of 0.54 t $_{CO_2}$ /t cement (see Akhtar et al..

$$CaCO_3 \rightarrow CaO + CO_2$$

Additionally, CO_2 is emitted from the combustion of fossil fuels to provide process heat. Thereby, cement constitutes the biggest source of industry process emissions in Europe.

Cement process emissions can be captured assuming a capture rate of 90%. Whether emissions are captured is decided by the model taking into account the capital costs of carbon capture modules. The electricity and heat demand of process emission carbon capture is currently ignored. For net-zero emission scenarios, the remaining process emissions need to be compensated by negative emissions.

With the exception of electricity demand and biomass demand for low-temperature heat (0.06 MWh/t and 0.2 MWh/t), the final energy consumption of this subsector is assumed to be supplied by methane (0.52 MWh/t), which is capable of delivering the required high-temperature heat. This implies a switch from burning solid fuels to burning gas which will require adjustments of the kilns. The share of fossil vs. synthetic methane consumed is a result of the optimisation

Ceramics

The ceramics sector is assumed to be fully electrified based on the current efficiency of already electrified processes which include microwave drying and sintering of raw materials, electric kilns for primary production processes, electric furnaces for the product finishing. In total, the final electricity consumption is 0.44 MWh/t of ceramic. The manufacturing of ceramics includes process emissions of 0.03 t $_{CO_2}$/t of ceramic. For a detailed overview of the ceramics industry sector see Furszyfer Del Rio et al.

Glass

The production of glass is assumed to be fully electrified based on the current efficiency of electric melting tanks and electric annealing which adds up to an electricity demand of 2.07 MWh $_{el}$/t of glass. The manufacturing of glass incurs process emissions of 0.1 t $_{CO_2}$/t of glass. Potential efficiency improvements, which according to Lechtenböhmer et al could reduce energy demands to 0.85 MW $_{el}$/t of glass, have not been considered. For a detailed overview of the glass industry sector see Furszyfer Del Rio et al.

Non-ferrous Metals

The non-ferrous metal subsector includes the manufacturing of base metals (aluminium, copper, lead, zinc), precious metals (gold, silver), and technology metals (molybdenum, cobalt, silicon).

The manufacturing of aluminium accounts for more than half of the final energy consumption of this subsector. Two alternative processing routes are used today to manufacture aluminium in Europe. The primary route represents 40% of the aluminium pro- duction, while the secondary route represents the remaining 60%.

The primary route involves two energy-intensive processes: the production of alumina from bauxite (aluminium ore) and the electrolysis to transform alumina into aluminium via the Hall-Héroult process

$$2Al_2O_3 + 3C \rightarrow 4Al + 3CO_2$$

The primary route requires high-enthalpy heat (2.3 MWh/t) to produce alumina which is supplied by methane and causes process emissions of 1.5 t $_{CO_2}$/t aluminium. According to Friedrichsen et al., inert anodes might become commercially available by 2030 that would eliminate the process emissions, but they are not included in the model. Assuming all subprocesses are electrified, the primary route requires 15.4 MWh $_{el}$/t of aluminium.

In the secondary route, scrap aluminium is remelted. The energy demand for this process is only 10% of the primary route and there are no associated process emissions. Assuming all subprocesses are electrified, the secondary route requires 1.7 MWh/t of aluminium. The share of aliminum manufactured by the primary and secondary route can be selected in the config file]

For the other non-ferrous metals, we assume the electrification of the entire manufacturing process with an average electricity demand of 3.2 MWh $_{el}$/t lead equivalent.

Other Industry Subsectors

The remaining industry subsectors include (a) pulp, paper, printing, (b) food, beverages, tobacco, (c) textiles and leather, (d) machinery equipment, (e) transport equipment, (f) wood and wood products, (g) others. Low- and mid-temperature process heat in these industries is assumed to be supplied by biomass while the remaining processes are electrified. None of the subsectors involve process emissions.

6.2.13 Agriculture demand

Energy demands for the agriculture, forestry and fishing sector per country are taken from the JRC-IDEES database. Missing countries are filled with Eurostat data. Agricultural energy demands are split into electricity (lighting, ventilation, specific electricity uses, electric pumping devices), heat (specific heat uses, low enthalpy heat), and machinery oil (motor drives, farming machine drives, diesel-fueled pumping devices). Heat demand is assigned at "services rural heat" buses. Time series for demands are assumed to be constant and distributed inside countries by population.

6.2.14 Transportation

Annual energy demands for land transport, aviation and shipping for every country are retrieved from JRC-IDEES data set. Below, the details of how each of these categories are treated is explained.

Land transport

Both road and rail transport is combined as land transport demand although electrified rail transport is excluded because that demand is included in the current electricity demand.

The most important settings for land transport are the exogenously fixed fuel mix (an option enabling the endogeous optimization of transport electrification is planned but not yet implemented). In the config file, the share of battery electric vehicles (BEV) and hydrogen fuel cell vehicles (FCEV) can be set. The remaining percentage will be treated as internal combustion engines (ICE) that consume oil products.

Battery Electric vehicles (BEV)

For the electrified land transport, country-specific factors are computed by comparing the current car final energy consumption per km in (average for Europe 0.7 kWh/km) to the 0.18 kWh/km value assumed for battery-to-wheels efficiency in EVs. The characteristic weekly profile provided by the German Federal Highway Research Institute (BASt) is used to obtain hourly time series for European countries taking into account the corresponding local times. Furthermore, a temperature dependence is included in the time series to account for heating/cooling demand in transport. For temperatures below/above certain threshold values, e.g. 15 °C/20 °C, temperature coefficients of typically 0.98%/°C and 0.63%/°C are assumed, based on the paper.

For BEVs the user can define the storage energy capacity, charging power capacity, and charging efficiency.

For BEV, smart charging is an option. A certain share of the BEV fleet can shift their charging time. The BEV state of charge is forced to be higher than a set percentage, e.g. 75%, every day at a specified hour, e.g., 7 am, to ensure that the batteries are sufficiently charged for peak usage in the morning and they not behave as seasonal storage. They also have the option to participate in vehicle-to-grid (V2G) services to facilitate system operation if that is enabled.

The battery cost of BEV is not included in the model since it is assumed that BEV owners buy them to primarily satisfy their mobility needs.

Hydrogen fuel cell vehicles (FCEV)

The share of all land transport that is specified to be be FCEV will be converted to a demand for hydrogen (see *Hydrogen supply*) using the FCEV efficiency.

FCEVs are typically used to simulate demand for transport that is hard to electrify directly, e.g. heavy construction machinery. But it may also be used to investigate a more widespread adoption of the technology.

Internal combustion engine vehicles (ICE)

All land transport that is not specified to be either BEV or FCEV will be treated as conventional ICEs. The transport demand is converted to a demand for oil products (see *Oil-based products supply*) using the ICE efficiency.

Aviation

The demand for aviation includes international and domestic use. It is modelled as an oil demand since aviation consumes kerosene. This can be produced synthetically or have fossil-origin (see *Oil-based products supply*).

Shipping

Shipping energy demand is covered by a combination of oil, hydrogen and methanol. Other fuel options, like ammonia, are currently not included in PyPSA-Eur-Sec. The share of shipping that is assumed to be supplied by hydrogen or methanol can be selected in the config file.

To estimate the hydrogen demand, the average fuel efficiency of the fleet is used in combination with the efficiency of the fuel cell defined in the technology-data repository. The average fuel efficiency is set in the config file.

The consumed hydrogen comes from the general hydrogen bus where it can be produced by SMR, SMR+CC or electrolysers (see *Hydrogen supply*). The fraction that is not converted into hydrogen use oil products, i.e. is connected to the general oil bus.

The energy demand for liquefaction of the hydrogen used for shipping can be included. If this option is selected, liquifaction will happen at the node where the shipping demand occurs.

The consumed methanol comes from the general methanol bus where it is produced through methanolisation (see *Chemicals Industry*).

6.2.15 Carbon dioxide capture, usage and sequestration (CCU/S)

PyPSA-Eur-Sec includes carbon capture from air (i.e., direct air capture (DAC)), electricity generators, and industrial facilities. It furthermore includes carbon dioxide storage and transport, the usage of carbon dioxide in synthetic methane and oil products, as well as the sequestration of carbon dioxide underground.

Carbon dioxide capture

For the following point source emissions, carbon capture is applicable:

- Industry process emissions, e.g., from limestone in cement production
- Methane or biomass used for process heat in the industry
- Hydrogen production by SMR
- CHP plants using biomass or methane
- Coal power plants.

Point source emissions are captured assuming a capture rate, e.g. 90%, which can be specified in the config file. The electricity and heat demand of process emission carbon capture is currently ignored.

DAC (if included) includes the adsorption phase where electricity and heat consumptionsare required to assist the adsorption process and regenerate the adsorbent. It also includes the drying and compression of CO_2 prior to storage which consumes electricity and rejects heat.

Carbon dioxide usage

Captured CO_2 can be used to produce synthetic methane and synthetic oil products (e.g. naphtha). If captured carbon is used, the CO_2 emissions of the synthetic fuels are net-neutral.

Carbon dioxide sequestration

Captured CO_2 can also be sequestered underground up to an annual sequestration limit of 200 Mt $_{CO_2}$/a. This limit can be chosen in the config file. As stored carbon dioxide is modelled as a single node for Europe, CO_2 transport constraints are neglected. Since CO_2 sequestration is an immature technology, the cost assumption is defined in the config file.

Carbon dioxide transport

Carbon dioxide can be modelled as a single node for Europe (in this case, CO_2 transport constraints are neglected). A network for modelling the transport of CO_2 among the different nodes can also be created if selected in the config file.

CHAPTER

SEVEN

REFERENCES

7.1 Release Notes

7.1.1 Upcoming Release

- Bugfix: The configuration setting `electricity: estimate_renewable_capacities: enable:` for rule *add_electricity* is not compatible with `foresight: myopic`, as the former adds existing renewable capacities in a different way in *add_existing_baseyear*. The logic was changed so that adding existing renewable capacities is now skipped in *add_electricity* if the foresight mode is `myopic`.
- Bugfix: Make sure that gas-fired power plants are correctly added as OCGT or CCGT in *add_electricity*. Previously they were always added as OCGT.
- Added default values for power distribution losses, assuming uniform losses of 3% on distribution grid links (cf. `sector: transmission_efficiency: electricity distribution grid: efficiency_static: 0.97`). Since distribution losses are included in national load reports (cf. this report), these are deducted from the national load time series to avoid double counting of losses. Further extensions to country-specific loss factors and developments by planning horizon are planned.
- Doubled solar rooftop potentials to roughly 1 TW for Europe based on recent European Commission reports.
- Remove exogenously set share of rooftop PV (`costs: rooftop_share:`). Rooftop and utility-scale PV are now largely handled as separate technologies with endogenous shares.
- New technology, solar PV with single-axis horizontal tracking (on a N-S axis), with a carrier called `solar-hsat` to the networks. The default option for adding this technology is set to `true` in the `config.yaml`.
- The technology-data version was updated to v0.9.0.
- Bugfix to avoid duplicated offshore regions.
- Added option `industry: HVC_environment_sequestration_fraction:` to specify the fraction of carbon contained plastics that is permanently sequestered in landfill. The default assumption is that all carbon contained in plastics is eventually released to the atmosphere.
- Added option for building waste-to-energy plants with and without carbon capture to consume non-recycled and non-sequestered plastics. The config settings are `industry: waste_to_energy:` and `industry: waste_to_energy_cc`. This does not include municipal solid waste.
- Bump minimum `powerplantmatching` version to v0.5.15.
- Add floating wind technology for water depths below 60m
- Add config `run: shared_resources: exclude:` to specify additional files that should be excluded from shared resources with the setting `run: shared_resources: base`. The function `_helpers/get_run_path()` now takes an additional keyword argument `exclude_from_shared` with a list of files that should not be shared. This keyword argument accepts a list of strings where the string only needs

to match the start of a filename (e.g. `"transport_data"` would exclude both `transport_data.csv` and `transport_data_{simpl}_{clusters}.csv` from being shared across scenarios.

- Move switch `run: shared_resources:` to `run: shared_resources: policy:`.
- Add config land_transport_demand_factor to model growth in land transport demand for different time horizons.
- Allow dictionary for the config aviation_demand_factor.
- Add option to post-discretize line and link capacities based on unit sizes and rounding thresholds specified in the configuration under `solving: options: post_discretization:` when iterative solving is enables (`solving: optiosn: skip_iterations: false`). This option is disabled by default.
- Group existing capacities to the earlier grouping_year for consistency with optimized capacities.
- Update data bundle:
  - Merge electricity-only and sector-coupled data bundles into one bundle. This means that the rule `retrieve_sector_databundle` was removed.
  - Include rasterised `natura.tiff` in data bundle and remove rule `retrieve_natura_raster`.
  - Remove rule `build_natura_raster` as this rule is rarely run and increases the data bundle size considerably.
  - Remove outdated files from data bundle (e.g., Eurostat energy balances)
  - Reduce spatial scope of GEBCO bathymetry data to Europe to save space.
  - Remove the use of a separate data bundle for tutorials.
  - Directly download Hotmaps Industrial Database from source and remove `Industrial_Database.csv` from data bundle.
- bugfix: installed heating capacities were 5% lower than existing heating capacities
- Include gas and oil fields and saline aquifers in estimation of CO2 sequestration potential.
- bugfix: convert Strings to pathlib.Path objects as input to ConfigSettings
- Allow the use of more solvers in clustering (Xpress, COPT, Gurobi, CPLEX, SCIP, MOSEK).
- Enhanced support for choosing different weather years (https://github.com/PyPSA/pypsa-eur/pull/204):
  - Processed energy statistics from eurostat (1990-2021) and IDEES (2000-2015) are now initially stored for all available years and filtered by the year given in `energy: energy_totals_year:`.
  - Added option to supplement electricity load data with synthetic time series for years not contained in OPSD (from https://zenodo.org/records/10820928, `load: supplement_synthetic:`).
  - The total annual heat demand for years not contained in the energy statistics by eurostat (1990-2021) or IDEES (2000-2015) are scaled based on a regression between the total number of heating degree days and the total annual heat demand between the years 2007-2021, assuming a similar building stock.
  - Added option to scale annual hydro-electricity generation data for years not contained in the in EIA (1980-2021) based on a regression between annual generation and total runoff per country for the years 1980-2021 (`renewable: hydro: eia_approximate_missing:`)
  - Added option to normalize annual hydro generation data by the associated installed capacity reported by EIA (1980-2021) in order to eliminate changes in generation due to newly built capacity (`renewable: hydro: eia_approximate_missing: eia_correct_by_capacity:`).
  - Added option to make hydro generation data independent of weather year (`renewable: hydro: eia_approximate_missing: eia_norm_year:`).
  - Added option to drop leap days (`enable: drop_leap_day:`).

  - Added option to make electric load data independent of weather year (`load: fixed_year:`).
  - Include time series of Swiss number of passenger vehicles from the Swiss Federal Statistical Office.
  - Updated hydro-electricity generation and capacity data from EIA.
  - The easiest way to sweep over multiple weather years is to use the new scenario management. An example for the necessary *create_scenarios.py* script can be found in this Github gist.
- Removed rule `copy_config`. Instead, a config file is created for each network output of the `solve_*` rules, with the same content as `n.meta`.
- Added new HVDC transmission projects from TYNDP 2024 draft projects.
- Upgrade to Snakemake v8.5+. This version is the new minimum version required. To upgrade an existing environment, run `conda install -c bioconda snakemake-minimal">=8.5"` and `pip install snakemake-storage-plugin-http` (https://github.com/PyPSA/pypsa-eur/pull/825).
- Corrected a bug leading to power plants operating after their DateOut (https://github.com/PyPSA/pypsa-eur/pull/958). Added additional grouping years before 1980.
- Add decommissioning of existing renewables assets in *add_existing_baseyear*.
- The Eurostat data was updated to the 2023 version in *build_energy_totals*.
- The latest Swiss energy totals have been updated to the 2023 version.
- The JRC-IDEES data is only available until 2015. For energy totals years (`energy: energy_totals_year`) after 2015, the data scaled using the ratio of Eurostat data reported for the energy totals year and 2015.
- The default energy totals year (`energy: energy_totals_year`) was updated to 2019.
- Upgrade default techno-economic assumptions to `technology-data` v0.8.1.
- Add possibility to download cost data from custom fork of `technology-data`.
- Linearly interpolate missing investment periods in year-dependent configuration options.
- Added new scenario management that supports the simultaneous execution of multiple scenarios with a single `snakemake` call. For this purpose, a `scenarios.yaml` file is introduced which contains customizable scenario names with configuration overrides. To enable it, set the `run: scenarios: true` and define the list of scenario names to run under `run: name:` in the configuration file. The latter must be a subset of toplevel keys in the scenario file.
  - To get started, a scenarios template file `config/scenarios.template.yaml` is included in the repository, which is copied to `config/scenarios.yaml` on first use.
  - The scenario file can be changed via `run: scenarios: file:`.
  - If scenario management is activated with `run: scenarios: enable: true`, a new wildcard `{run}` is introduced. This means that the configuration settings may depend on the new `{run}` wildcard. Therefore, a new `config_provider()` function is used in the `Snakefile` and `.smk` files, which takes wildcard values into account. The calls to the `config` object have been reduced in `.smk` files since there is no awareness of wildcard values outside rule definitions.
  - The scenario files can also be programmatically created using the template script `config/create_scenarios.py`. This script can be run with `snakemake -j1 create_scenarios` and creates the scenarios file referenced under `run: scenarios: file:`.
  - The setting `run: name: all` will run all scenarios in `config/scenarios.yaml`. Otherwise, it will run those passed as list in `run: name:` as long as `run: scenarios: enable: true`.
  - The setting `run: shared_resources:` indicates via a boolean whether the resources should be encapsulated by the `run: name:`. The special setting `run: shared_resources: base` shares resources until `add_electricity` that do not contain wildcards other than `{"technology", "year", "scope"}`.

  - Added new configuration options for all `{opts}` and `{sector_opts}` wildcard values to create a unique configuration file (`config.yaml`) per PyPSA network file. This is done with the help of a new function `update_config_from_wildcards()` which parses configuration settings from wildcards and updates the `snakemake.config` object. These updated configuration settings are used in the scripts rather than directly parsed values from `snakemake.wildcards`.
  - The cost data was moved from `data/costs_{year}.csv` to `resources/costs_{year}.csv` since it depends on configuration settings. The `retrieve_cost_data` rule was changed to calling a Python script.
  - Moved time clustering settings to `clustering: temporal:` from `snapshots:` so that the latter is only used to define the `pandas.DatetimeIndex` which simplifies the scenario management.
  - Collection rules get a new wildcard `run=config["run"]["name"]` so they can collect outputs across different scenarios.
  - It is further possible to encapsulate your scenarios in a directory using the setting `run: prefix:`.
  - **Warning:** One caveat remains for the scenario management with myopic or perfect foresight pathway optimisation. The first investment period must be shared across all scenarios. The reason is that the `wildcard_constraints` defined for the rule `add_existing_baseyear` do not accept wildcard-aware input functions (cf. **`https://github.com/snakemake/snakemake/issues/2703`**_).

- The outputs of the rule `retrieve_gas_infrastructure_data` no longer marked as `protected()` as the download size is small.
- Bugfix: allow modelling sector-coupled landlocked regions. (Fixed handling of offshore wind.)
- Adapt the disabling of transmission expansion in myopic foresight optimisations when limit is already reached to also handle cost limits.
- Fix duplicated years and grouping years reference in *add_land_use_constraint_m*.
- Fix type error with *m* option in *cluster_network*.
- Fix error with *symbol* of *buses* in *simplify_network*.
- Fix index of existing capacities in *add_power_capacities_installed_before_baseyear* with *m* option.
- Fix fill missing data in *build_industry_sector_ratios_intermediate*.
- Fix custom busmap read in *cluster_network*.
- Clarify suffix usage in *add_existing_baseyear*.
- Fix p_nom_min of renewables generators for myopic approach and add check of existing capacities in *add_land_use_constraint_m*.
- Improved the behaviour of *agg_p_nom_limits*:
  - Moved the associated configuration to *solving*. This allows *Snakemake* to correctly decide which rules to run when the configuration changes.
  - Added the ability to enable aggregation of all *offwind* types (*offwind-ac* and *offwind-dc*) when writing the constraint.
  - Added the possibility to take existing capacities into account when writing the constraint.
  - Added the possibility to have a different file for each planning horizon.
- Fix gas network retrofitting in *add_brownfield*.
- Add *nodal_supply_energy* to *make_summary*.
- Change the methanol energy demand of industry to the low-carbon route defined by DECHEMA report.

7.1.2 PyPSA-Eur 0.10.0 (19th February 2024)

New Features

- Improved representation of industry transition pathways. A new script was added to interpolate industry sector ratios from today's status quo to future systems (i.e. specific emissions and demands for energy and feedstocks). For each country we gradually switch industry processes from today's specific energy carrier usage per ton material output to the best-in-class energy consumption of tomorrow. This is done on a per-country basis. The ratio of today to tomorrow's energy consumption is set with the `industry: sector_ratios_fraction_future:` parameter (https://github.com/PyPSA/pypsa-eur/pull/929).
- Add new default to overdimension heating in individual buildings. This allows them to cover heat demand peaks e.g. 10% higher than those in the data. The disadvantage of manipulating the costs is that the capacity is then not quite right. This way at least the costs are right (https://github.com/PyPSA/pypsa-eur/pull/918).
- Allow industrial coal demand to be regional so its emissions can be included in regional emission limits (https://github.com/PyPSA/pypsa-eur/pull/923).
- Add option to specify to set a default heating lifetime for existing heating (`existing_capacities: default_heating_lifetime:`) (https://github.com/PyPSA/pypsa-eur/pull/918).
- Added option to specify turbine and solar panel models for specific years as a dictionary (e.g. `renewable: onwind: resource: turbine:`). The years will be interpreted as years from when the the corresponding turbine model substitutes the previous model for new installations. This will only have an effect on workflows with foresight `"myopic"` and still needs to be added foresight option `"perfect"` (https://github.com/PyPSA/pypsa-eur/pull/912).
- New configuration option `everywhere_powerplants` to build conventional powerplants everywhere, irrespective of existing powerplants locations, in the network (https://github.com/PyPSA/pypsa-eur/pull/850).
- Add the option to customise map projection in plotting config under `plotting: projection: name` (https://github.com/PyPSA/pypsa-eur/pull/898).
- Add support for the linopy `io_api` option under `solving: options: io_api:`. Set to `"direct"` to increase model reading and writing performance for the highs and gurobi solvers on slow file systems (https://github.com/PyPSA/pypsa-eur/pull/892).
- It is now possible to determine the directory for shared resources by setting *shared_resources* to a string (https://github.com/PyPSA/pypsa-eur/pull/906).
- Improve `mock_snakemake()` for usage in Snakemake modules (https://github.com/PyPSA/pypsa-eur/pull/869).

Breaking Changes

- Remove long-deprecated function `attach_extendable_generators` in *add_electricity*.
- Remove option for wave energy as technology data is not maintained.
- The order of buses (bus0, bus1, ...) for DAC components has changed to meet the convention of the other components. Therefore, *bus0* refers to the electricity bus (input), *bus1* to the heat bus (input), 'bus2' to the CO2 atmosphere bus (input), and *bus3* to the CO2 storage bus (output) (https://github.com/PyPSA/pypsa-eur/pull/901).

Changes

- Upgrade default techno-economic assumptions to `technology-data` v0.8.0.
- Update hydrogen pipeline losses to latest data from Danish Energy Agency (https://github.com/PyPSA/pypsa-eur/pull/933).
- Move building of daily heat profile to its own rule *build_hourly_heat_demand* from *prepare_sector_network* (https://github.com/PyPSA/pypsa-eur/pull/884).

- In *build_energy_totals*, district heating shares are now reported in a separate file (https://github.com/PyPSA/pypsa-eur/pull/884).
- Move calculation of district heating share to its own rule *build_district_heat_share* (https://github.com/PyPSA/pypsa-eur/pull/884).
- Move building of distribution of existing heating to own rule *build_existing_heating_distribution*. This makes the distribution of existing heating to urban/rural, residential/services and spatially more transparent (https://github.com/PyPSA/pypsa-eur/pull/884).
- Default settings for recycling rates and primary product shares of high-value chemicals have been set in accordance with the values used in Neumann et al. (2023) linearly interpolated between 2020 and 2050. The recycling rates are based on data from Agora Energiewende (2021).
- Air-sourced heat pumps can now also be built in rural areas. Previously, only ground-sourced heat pumps were considered for this category (https://github.com/PyPSA/pypsa-eur/pull/890).
- The default configuration `config/config.default.yaml` is now automatically used as a base configuration file. The file `config/config.yaml` can now be used to only define deviations from the default configuration. The `config/config.default.yaml` is still copied into `config/config.yaml` on first usage (https://github.com/PyPSA/pypsa-eur/pull/925).
- Regions are assigned to all buses with unique coordinates in the network with a preference given to substations. Previously, only substations had assigned regions, but this could lead to issues when a high spatial resolution was applied (https://github.com/PyPSA/pypsa-eur/pull/922).
- Define global constraint for CO2 emissions on the final state of charge of the CO2 atmosphere store. This gives a more sparse constraint that should improve the performance of the solving process (https://github.com/PyPSA/pypsa-eur/pull/862).
- Switched the energy totals year from 2011 to 2013 to comply with the assumed default weather year (https://github.com/PyPSA/pypsa-eur/pull/934).
- Cluster residential and services heat buses by default. Can be disabled with `cluster_heat_buses: false` (https://github.com/PyPSA/pypsa-eur/pull/877).
- The rule `plot_network` has been split into separate rules for plotting electricity, hydrogen and gas networks (https://github.com/PyPSA/pypsa-eur/pull/900).
- To determine the optimal topology to meet the number of clusters, the workflow used pyomo in combination with `ipopt` or `gurobi`. This dependency has been replaced by using `linopy` in combination with `scipopt` or `gurobi`. The environment file has been updated accordingly (https://github.com/PyPSA/pypsa-eur/pull/903).
- The `highs` solver was added to the default environment file.
- New default solver settings for COPT solver (https://github.com/PyPSA/pypsa-eur/pull/882).
- Data retrieval rules now use their own minimal conda environment. This can avoid unnecessary reruns of the workflow (https://github.com/PyPSA/pypsa-eur/pull/888).
- Merged two OPSD time series data versions into such that the option `load: power_statistics:` becomes superfluous and was hence removed (https://github.com/PyPSA/pypsa-eur/pull/924).
- The filtering of power plants in the `config.default.yaml` has been updated regarding phased-out power plants in 2023.
- Include all countries in ammonia production resource. This is so that the full EU28 ammonia demand can be correctly subtracted in the rule *build_industry_sector_ratios* (https://github.com/PyPSA/pypsa-eur/pull/931).
- Correctly source the existing heating technologies for buildings since the source URL has changed. It represents the year 2012 and is only for buildings, not district heating (https://github.com/PyPSA/pypsa-eur/pull/918).

- Add warning when BEV availability weekly profile has negative values in *build_transport_demand* (https://github.com/PyPSA/pypsa-eur/pull/858).
- Time series clipping for very small values was added for Links (https://github.com/PyPSA/pypsa-eur/pull/870).
- A `test.sh` script was added to the repository to run the tests locally.
- The CI now tests additionally against `master` versions of PyPSA, atlite and powerplantmatching (https://github.com/PyPSA/pypsa-eur/pull/904).
- A function `sanitize_locations()` was added to improve the coverage of the `location` attribute of network components.

Bugs and Compatibility

- Bugfix: Do not reduce district heat share when building population-weighted energy statistics. Previously the district heating share was being multiplied by the population weighting, reducing the DH share with multiple nodes (https://github.com/PyPSA/pypsa-eur/pull/884).
- Bugfix: The industry coal emissions for industry were not properly tracked (https://github.com/PyPSA/pypsa-eur/pull/923).
- Bugfix: Correct units of subtracted chlorine and methanol demand in *build_industry_sector_ratios* (https://github.com/PyPSA/pypsa-eur/pull/930).
- Various minor bugfixes to the perfect foresight workflow, though perfect foresight must still be considered experimental (https://github.com/PyPSA/pypsa-eur/pull/910).
- Fix plotting of retrofitted hydrogen pipelines with myopic pathway optimisation (https://github.com/PyPSA/pypsa-eur/pull/937).
- Bugfix: Correct technology keys for the electricity production plotting to work out the box.
- Bugfix: Assure entering of code block which corrects Norwegian heat demand (https://github.com/PyPSA/pypsa-eur/pull/870).
- Stacktrace of uncaught exceptions should now be correctly included inside log files (via *configure_logging(..)*) (https://github.com/PyPSA/pypsa-eur/pull/875).
- Bugfix: Correctly read out number of solver threads from configuration file (https://github.com/PyPSA/pypsa-eur/pull/889).
- Made copying default config file compatible with snakemake module (https://github.com/PyPSA/pypsa-eur/pull/894).
- Compatibility with `pandas=2.2` (https://github.com/PyPSA/pypsa-eur/pull/861).

Special thanks for this release to Koen van Greevenbroek (@koen-vg) for various new features, bugfixes and taking care of deprecations.

7.1.3 PyPSA-Eur 0.9.0 (5th January 2024)

New Features

- Add option to specify losses for bidirectional links, e.g. pipelines or HVDC links, in configuration file under `sector: transmission_efficiency:`. Users can specify static or length-dependent values as well as a length-dependent electricity demand for compression, which is implemented as a multi-link to the local electricity buses. The bidirectional links will then be split into two unidirectional links with linked capacities (https://github.com/PyPSA/pypsa-eur/pull/739).
- Merged option to extend geographical scope to Ukraine and Moldova. These countries are excluded by default and is currently constrained to power-sector only parts of the workflow. A special config file *config/config.entsoe-all.yaml* was added as an example to run the workflow with all ENTSO-E member countries (including observer

members like Ukraine and Moldova). Moldova can currently only be included in conjunction with Ukraine due to the absence of demand data. The Crimean power system is manually reconnected to the main Ukrainian grid with the configuration option *reconnect_crimea* (https://github.com/PyPSA/pypsa-eur/pull/321).

- New experimental support for multi-decade optimisation with perfect foresight (`foresight: perfect`). Maximum growth rates for carriers, global carbon budget constraints and emission constraints for particular investment periods.
- Add option to reference an additional source file where users can specify custom `extra_functionality` constraints in the configuration file. The default setting points to an empty hull at `data/custom_extra_functionality.py` (https://github.com/PyPSA/pypsa-eur/pull/824).
- Add locations, capacities and costs of existing gas storage using Global Energy Monitor's Europe Gas Tracker (https://github.com/PyPSA/pypsa-eur/pull/835).
- Add option to use LUISA Base Map 50m land coverage dataset for land eligibility analysis in *build_renewable_profiles*. Settings are analogous to the CORINE dataset but with the key `luisa:` in the configuration file. To leverage the dataset's full advantages, set the excluder resolution to 50m (`excluder_resolution: 50`). For land category codes, see Annex 1 of the technical documentation (https://github.com/PyPSA/pypsa-eur/pull/842).
- Add option to capture CO2 contained in biogas when upgrading (`sector: biogas_to_gas_cc`) (https://github.com/PyPSA/pypsa-eur/pull/615).
- If load shedding is activated, it is now applied to all carriers, not only electricity (https://github.com/PyPSA/pypsa-eur/pull/784).
- Add option for heat vents in district heating (`sector: central_heat_vent:`). The combination of must-run conditions for some power-to-X processes, waste heat usage enabled and decreasing heating demand, can lead to infeasibilities in pathway optimisation for some investment periods since larger Fischer-Tropsch capacities are needed in early years but the waste heat exceeds the heat demand in later investment periods. (https://github.com/PyPSA/pypsa-eur/pull/791).
- Allow possibility to go from copperplated to regionally resolved methanol and oil demand with switches `sector: regional_methanol_demand: true` and `sector: regional_oil_demand: true`. This allows nodal/regional CO2 constraints to be applied (https://github.com/PyPSA/pypsa-eur/pull/827).
- Allow retrofitting of existing gas boilers to hydrogen boilers in pathway optimisation.
- Add option to add time-varying CO2 emission prices (electricity-only, `costs: emission_prices: co2_monthly_prices: true`). This is linked to the new `{opts}` wildcard option `Ept`.
- Network clustering can now consider efficiency classes when aggregating carriers. The option `clustering: consider_efficiency_classes:` aggregates each carriers into the top 10-quantile (high), the bottom 90-quantile (low), and everything in between (medium).
- Added option `conventional: dynamic_fuel_price:` to consider the monthly fluctuating fuel prices for conventional generators. Refer to the CSV file `data/validation/monthly_fuel_price.csv`.
- For hydro-electricity, add switches `flatten_dispatch` to consider an upper limit for the hydro dispatch. The limit is given by the average capacity factor plus the buffer given in `flatten_dispatch_buffer`.
- Extend options for waste heat usage from Haber-Bosch, methanolisation and methanation (https://github.com/PyPSA/pypsa-eur/pull/834).
- Add new `sector_opts` wildcard option "nowasteheat" to disable all waste heat usage (https://github.com/PyPSA/pypsa-eur/pull/834).
- Add new rule `retrieve_irena` to automatically retrieve up-to-date values for existing renewables capacities (https://github.com/PyPSA/pypsa-eur/pull/756).

- Print Irreducible Infeasible Subset (IIS) if model is infeasible. Only for solvers with IIS support (https://github.com/PyPSA/pypsa-eur/pull/841).
- More wildcard options now have a corresponding config entry. If the wildcard is given, then its value is used. If the wildcard is not given but the options in config are enabled, then the value from config is used. If neither is given, the options are skipped (https://github.com/PyPSA/pypsa-eur/pull/827).
- Validate downloads from Zenodo using MD5 checksums. This identifies corrupted or incomplete downloads (https://github.com/PyPSA/pypsa-eur/pull/821).
- Add rule `sync` to synchronise with a remote machine using the `rsync` library. Configuration settings are found under `remote:`.

Breaking Changes

- Remove all negative loads on the `co2 atmosphere` bus representing emissions for e.g. fixed fossil demands for transport oil. Instead these are handled more transparently with a fixed transport oil demand and a link taking care of the emissions to the `co2 atmosphere` bus. This is also a preparation for endogenous transport optimisation, where demand will be subject to optimisation (e.g. fuel switching in the transport sector) (https://github.com/PyPSA/pypsa-eur/pull/827).
- Process emissions from steam crackers (i.e. naphtha processing for HVC) are now piped from the consumption link to the process emissions bus where the model can decide about carbon capture. Previously the process emissions for naphtha were a fixed load (https://github.com/PyPSA/pypsa-eur/pull/827).
- Distinguish between stored and sequestered CO2. Stored CO2 is stored overground in tanks and can be used for CCU (e.g. methanolisation). Sequestered CO2 is stored underground and can no longer be used for CCU. This distinction is made because storage in tanks is more expensive than underground storage. The link that connects stored and sequestered CO2 is unidirectional (https://github.com/PyPSA/pypsa-eur/pull/844).
- Files extracted from sector-coupled data bundle have been moved from `data/` to `data/sector-bundle`.
- Split configuration to enable SMR and SMR CC (`sector: smr:` and `sector: smr_cc:`) (https://github.com/PyPSA/pypsa-eur/pull/757).
- Add separate option to add resistive heaters to the technology choices (`sector: resistive_heaters:`). Previously they were always added when boilers were added (https://github.com/PyPSA/pypsa-eur/pull/808).
- Remove HELMETH option (`sector: helmeth:`).
- Remove "conservative" renewable potentials estimation option (https://github.com/PyPSA/pypsa-eur/pull/838).
- With this release we stop posting updates to the network pre-builts.

Changes

- Updated Global Energy Monitor LNG terminal data to March 2023 version (https://github.com/PyPSA/pypsa-eur/pull/707).
- For industry distribution, use EPRTR as fallback if ETS data is not available (https://github.com/PyPSA/pypsa-eur/pull/721).
- It is now possible to specify years for biomass potentials which do not exist in the JRC-ENSPRESO database, e.g. 2037. These are linearly interpolated (https://github.com/PyPSA/pypsa-eur/pull/744).
- In pathway mode, the biomass potential is linked to the investment year (https://github.com/PyPSA/pypsa-eur/pull/744).
- Increase allowed deployment density of solar to 5.1 MW/sqkm by default.
- Default to full electrification of land transport by 2050.
- Provide exogenous transition settings in 5-year steps.
- Default to approximating transmission losses in HVAC lines (`transmission_losses: 2`).

- Use electrolysis waste heat by default.
- Set minimum part loads for PtX processes to 30% for methanolisation and methanation, and to 70% for Fischer-Tropsch synthesis.
- Add VOM as marginal cost to PtX processes (https://github.com/PyPSA/pypsa-eur/pull/830).
- Add pelletizing costs for biomass boilers (https://github.com/PyPSA/pypsa-eur/pull/833).
- Update default offshore wind turbine model to "NREL Reference 2020 ATB 5.5 MW" (https://github.com/PyPSA/pypsa-eur/pull/832).
- Switch to using hydrogen and electricity inputs for Haber-Bosch from https://github.com/PyPSA/technology-data (https://github.com/PyPSA/pypsa-eur/pull/831).
- The configuration setting for country focus weights when clustering the network has been moved from `focus_weights:` to `clustering: focus_weights:`. Backwards compatibility to old config files is maintained (https://github.com/PyPSA/pypsa-eur/pull/794).
- The `mock_snakemake` function can now be used with a Snakefile from a different directory using the new `root_dir` argument (https://github.com/PyPSA/pypsa-eur/pull/771).
- Rule `purge` now initiates a dialog to confirm if purge is desired (https://github.com/PyPSA/pypsa-eur/pull/745).
- Files downloaded from zenodo are now write-protected to prevent accidental re-download (https://github.com/PyPSA/pypsa-eur/pull/730).
- Performance improvements for rule `build_ship_raster` (https://github.com/PyPSA/pypsa-eur/pull/845).
- Improve time logging in *`build_renewable_profiles`* (https://github.com/PyPSA/pypsa-eur/pull/837).
- In myopic pathway optimisation, disable power grid expansion if line volume already hit (https://github.com/PyPSA/pypsa-eur/pull/840).
- JRC-ENSPRESO data is now downloaded from a Zenodo mirror because the link was unreliable (https://github.com/PyPSA/pypsa-eur/pull/801).
- Add focus weights option for clustering to documentation (https://github.com/PyPSA/pypsa-eur/pull/781).
- Add proxy for biomass transport costs if no explicit biomass transport network is considered (https://github.com/PyPSA/pypsa-eur/pull/711).

Bugs and Compatibility

- The minimum PyPSA version is now 0.26.1.
- Update to `tsam>=0.2.3` for performance improvents in temporal clustering.
- Pin `snakemake` version to below 8.0.0, as the new version is not yet supported. The next release will switch to the requirement `snakemake>=8`.
- Bugfix: Add coke and coal demand for integrated steelworks (https://github.com/PyPSA/pypsa-eur/pull/718).
- Bugfix: Make *`build_renewable_profiles`* consider subsets of cutout time scope (https://github.com/PyPSA/pypsa-eur/pull/709).
- Bugfix: In `simplify network`, remove 'underground' column to avoid consense error (https://github.com/PyPSA/pypsa-eur/pull/714).
- Bugfix: Fix in *`add_existing_baseyear`* to account for the case when there is no rural heating demand for some nodes in network (https://github.com/PyPSA/pypsa-eur/pull/706).
- Bugfix: The unit of the capital cost of Haber-Bosch plants was corrected (https://github.com/PyPSA/pypsa-eur/pull/829).

- The minimum capacity for renewable generators when using the myopic option has been fixed (https://github.com/PyPSA/pypsa-eur/pull/728).
- Compatibility for running with single node and single country (https://github.com/PyPSA/pypsa-eur/pull/839).
- A bug preventing the addition of custom powerplants specified in `data/custom_powerplants.csv` was fixed. (https://github.com/PyPSA/pypsa-eur/pull/732)
- Fix nodal fraction in `add_existing_year` when using distributed generators (https://github.com/PyPSA/pypsa-eur/pull/798).
- Bugfix: District heating without progress caused division by zero (https://github.com/PyPSA/pypsa-eur/pull/796).
- Bugfix: Drop duplicates in `build_industrial_distribution_keys`, which can occur through the geopandas `.sjoin()` function if a point is located on a border (https://github.com/PyPSA/pypsa-eur/pull/726).
- For network clustering fall back to `ipopt` when `highs` is designated solver (https://github.com/PyPSA/pypsa-eur/pull/795).
- Fix typo in buses definition for oil boilers in `add_industry` in *`prepare_sector_network`* (https://github.com/PyPSA/pypsa-eur/pull/812).
- Resolve code issues for endogenous building retrofitting. Select correct sector names, address deprecations, distinguish between district heating, decentral heating in urban areas or rural areas for floor area calculations (https://github.com/PyPSA/pypsa-eur/pull/808).
- Addressed various deprecations.

7.1.4 PyPSA-Eur 0.8.1 (27th July 2023)

New Features

- Add option to consider dynamic line rating based on wind speeds and temperature according to Glaum and Hofmann (2022). See configuration section `lines: dynamic_line_rating:` for more details. (https://github.com/PyPSA/pypsa-eur/pull/675)
- Add option to include a piecewise linear approximation of transmission losses, e.g. by setting `solving: options: transmission_losses: 2` for an approximation with two tangents. (https://github.com/PyPSA/pypsa-eur/pull/664)
- Add plain hydrogen turbine as additional re-electrification option besides hydrogen fuel cell. Add switches for both re-electrification options under `sector: hydrogen_turbine:` and `sector: hydrogen_fuel_cell:`. (https://github.com/PyPSA/pypsa-eur/pull/647)
- Added configuration option `lines: max_extension:` and `links: max_extension:` to control the maximum capacity addition per line or link in MW. (https://github.com/PyPSA/pypsa-eur/pull/665)
- A `param:` section in the snakemake rule definitions was added to track changed settings in `config.yaml`. The goal is to automatically re-execute rules where parameters have changed. See Non-file parameters for rules in the snakemake documentation. (https://github.com/PyPSA/pypsa-eur/pull/663)
- A new function named `sanitize_carrier` ensures that all unique carrier names are present in the network's carriers attribute, and adds nice names and colors for each carrier according to the provided configuration dictionary. (https://github.com/PyPSA/pypsa-eur/pull/653, https://github.com/PyPSA/pypsa-eur/pull/690)
- The configuration settings have been documented in more detail. (https://github.com/PyPSA/pypsa-eur/pull/685)

Breaking Changes

- The configuration files are now located in the `config` directory. This includes the `config.default.yaml`, `config.yaml` as well as the test configuration files which are now located in the `config/test` directory. Config files that are still in the root directory will be ignored. (https://github.com/PyPSA/pypsa-eur/pull/640)
- Renamed script and rule name from `build_load_data` to `build_electricity_demand` and `retrieve_load_data` to `retrieve_electricity_demand`. (https://github.com/PyPSA/pypsa-eur/pull/642, https://github.com/PyPSA/pypsa-eur/pull/652)
- Updated to new spatial clustering module introduced in PyPSA v0.25. (https://github.com/PyPSA/pypsa-eur/pull/696)

Changes

- Handling networks with links with multiple inputs/outputs no longer requires to override component attributes. (https://github.com/PyPSA/pypsa-eur/pull/695)
- Added configuration option `enable: retrieve:` to control whether data retrieval rules from snakemake are enabled or not. Th default setting `auto` will automatically detect and enable/disable the rules based on internet connectivity. (https://github.com/PyPSA/pypsa-eur/pull/694)
- Update to `technology-data` v0.6.0. (https://github.com/PyPSA/pypsa-eur/pull/704)
- Handle data bundle extraction paths via `snakemake.output`.
- Additional technologies are added to `tech_color` in the configuration files to include previously unlisted carriers.
- Doc: Added note that Windows is only tested in CI with WSL. (https://github.com/PyPSA/pypsa-eur/issues/697)
- Doc: Add support section. (https://github.com/PyPSA/pypsa-eur/pull/656)
- Open `rasterio` files with `rioxarray`. (https://github.com/PyPSA/pypsa-eur/pull/474)
- Migrate CI to `micromamba`. (https://github.com/PyPSA/pypsa-eur/pull/700)

Bugs and Compatibility

- The new minimum PyPSA version is v0.25.1.
- Removed `vresutils` dependency. (https://github.com/PyPSA/pypsa-eur/pull/662)
- Adapt to new `powerplantmatching` version. (https://github.com/PyPSA/pypsa-eur/pull/687, https://github.com/PyPSA/pypsa-eur/pull/701)
- Bugfix: Correct typo in the CPLEX solver configuration in `config.default.yaml`. (https://github.com/PyPSA/pypsa-eur/pull/630)
- Bugfix: Error in `add_electricity` where carriers were added multiple times to the network, resulting in a non-unique carriers error.
- Bugfix of optional reserve constraint. (https://github.com/PyPSA/pypsa-eur/pull/645)
- Fix broken equity constraints logic. (https://github.com/PyPSA/pypsa-eur/pull/679)
- Fix addition of load shedding generators. (https://github.com/PyPSA/pypsa-eur/pull/649)
- Fix automatic building of documentation on readthedocs.org. (https://github.com/PyPSA/pypsa-eur/pull/658)
- Bugfix: Update network clustering to avoid adding deleted links in clustered network. (https://github.com/PyPSA/pypsa-eur/pull/678)
- Address `geopandas` deprecations. (https://github.com/PyPSA/pypsa-eur/pull/678)
- Fix bug with underground hydrogen storage creation, where for some small model regions no cavern storage is available. (https://github.com/PyPSA/pypsa-eur/pull/672)
- Addressed deprecation warnings for `pandas=2.0`. `pandas=2.0` is now minimum requirement.

7.1.5 PyPSA-Eur 0.8.0 (18th March 2023)

Note: This is the first release of PyPSA-Eur which incorporates its sector-coupled extension PyPSA-Eur-Sec (v0.7.0). PyPSA-Eur can now directly be used for high-resolution energy system modelling with sector-coupling including industry, transport, buildings, biomass, and detailed carbon management. The PyPSA-Eur-Sec repository is now deprecated.

- The *solve_network* script now uses the `linopy` backend of PyPSA and is applied for both electricity-only and sector-coupled models. This requires an adjustment of custom `extra_functionality`. See the migration guide in the PyPSA documentation.
- The configuration file `config.default.yaml` now also includes settings for sector-coupled models, which will be ignored when the user runs electricity-only studies. Common settings have been aligned.
- Unified handling of scenario runs. Users can name their scenarios in `run: name:`, which will encapsulate results in a correspondingly named folder under `results`. Additionally, users can select to encapsulate the `resources` folder in the same way, through the setting `run: shared_resources:`.
- The solver configurations in `config.default.yaml` are now modularized. To change the set of solver options, change to value in `solving: solver: options:` to one of the keys in `solving: solver_options:`.
- The `Snakefile` has been modularised. Rules are now organised in the `rules` directory.
- Unified wildcard for transmission line expansion from `{lv}` and `{ll}` to `{ll}`.
- Renamed collection rules to distinguish between sector-coupled and electricity-only runs: `cluster_networks`, `extra_components_networks`, `prepare_elec_networks`, `prepare_sector_networks`, `solve_elec_networks`, `solve_sector_networks`, `plot_networks`, `all`.
- Some rules with a small computational footprint have been declared as `localrules`.
- Added new utility rules `purge` for clearing workflow outputs from the directory, `doc` to build the documentation, and `dag` to create a workflow graph.
- The workflow can now be used with the `snakemake --use-conda` directive. In this way, Snakemake can automatically handle the installation of dependencies.
- Data retrieval rules now retry download twice in case of connection problems.
- The cutouts are now marked as `protected()` in the workflow to avoid accidental recomputation.
- The files contained in `data/bundle` are now marked as `ancient()` as they are not expected to be altered by workflow changes.
- Preparation scripts for sector-coupled models have been improved to only run for the subset of selected countries rather than all European countries.
- Added largely automated country code conversion using `country_converter`..
- Test coverage extended to an electricity-only run and sector-coupled runs for overnight and myopic foresight scenarios for Ubuntu, MacOS and Windows.
- Apply `black` and `snakefmt` code formatting.
- Implemented REUSE compatibility for merged code.
- Merged documentations of PyPSA-Eur and PyPSA-Eur-Sec.
- Added a tutorial for running sector-coupled models to the documentation (*Tutorial: Sector-Coupled*).
- Deleted `config.tutorial.yaml`, which is superseded by `test/config.electricity.yaml`.
- The `mock_snakemake` function now also takes configuration files as inputs.

- The helper scripts `helper.py` and `_helpers.py` have been merged into `_helpers.py`.
- The unused rule `plot_p_nom_max` has been removed.
- The rule `solve_network` from PyPSA-Eur-Sec was renamed to `solve_sector_network`.
- The plotting scripts from PyPSA-Eur (electricity-only) have been removed and are superseded by those from PyPSA-Eur-Sec (sector-coupled).

7.1.6 PyPSA-Eur Releases (pre-merge)

PyPSA-Eur 0.7.0 (16th February 2023)

New Features

- Carriers of generators can now be excluded from aggregation in clustering network and simplify network (see `exclude_carriers`).
- Added control for removing stubs in *simplify_network* with options `remove_stubs` and `remove_stubs_across_countries`.
- Add control for showing a progressbar in `atlite` processes (`show_progress`). Disabling the progressbar saves a lot of time.
- Added control for resolution of land eligibility analysis (see `excluder_resolution`).

Breaking Changes

- The config entry `snapshots: closed:` was renamed to `snapshots: inclusive:` to address the upstream deprecation with `pandas=1.4`. The previous setting `None` is no longer supported and replaced by `both`, see the pandas documentation. Minimum version is now `pandas>=1.4`.
- The configuration setting `summary_dir` was removed.

Changes

- Configuration defaults to new `technology-data` version 0.5.0.
- Fixed CRS warnings when projection of datasets was not specified.
- Cleaned shape unary unions.
- Increased resource requirements for some rules.
- Updated documentation.
- The documentation now uses the `sphinx_book_theme`.

Bugs and Compatibility

- Bugfix: Corrected extent of natural protection areas in *build_natura_raster*.
- Bugfix: Use correct load variables for formulating reserve constraints.
- Bugfix: Use all available energy-to-power ratios for hydropower plants.
- Bugfix: The most recent processing of the `entsoegridkit` extract required further manual corrections. Also, the connection points of TYNDP links were corrected.
- Bugfix: Handle absence of hydropower inflow in `EQ` constraint.
- Compatibility with `pyomo>=6.4.3` in *cluster_network*.
- Upgrade to `shapely>=2`.
- Updated version of CI cache action to version 3.

-

- Updated dependency constraints in `environment.yaml`.
- Address various deprecation warnings.

PyPSA-Eur 0.6.1 (20th September 2022)

- Individual commits are now tested against pre-commit hooks. This includes black style formatting, sorting of package imports, Snakefile formatting and others. Installation instructions can for the pre-commit can be found here.
- Pre-commit CI is now part of the repository's CI.
- The software now supports running the workflow with different settings within the same directory. A new config section `run` was created that specifies under which scenario `name` the created resources, networks and results should be stored. If `name` is not specified, the workflow uses the default paths. The entry `shared_cutouts` specifies whether the run should use cutouts from the default root directory or use run-specific cutouts.
- The heuristic distribution of today's renewable capacity installations is now enabled by default.
- The marginal costs of conventional generators are now taking the plant-specific efficiency into account where available.

PyPSA-Eur 0.6.0 (10th September 2022)

- Functionality to consider shipping routes when calculating the available area for offshore technologies were added. Data for the shipping density comes from the Global Shipping Traffic Density dataset.
- When transforming all transmission lines to a unified voltage level of 380kV, the workflow now preserves the transmission capacity rather than electrical impedance and reactance.
- Memory resources are now specified for all rules.
- Filtering of power plant data was adjusted to new versions of `powerplantmatching`.
- The resolution of land exclusion calculation is now a configurable option. See setting `excluder_resolution`.

PyPSA-Eur 0.5.0 (27th July 2022)

New Features

- New network topology extracted from the ENTSO-E interactive map.
- Added existing renewable capacities for all countries based on IRENA statistics (IRENASTAT) using new `powerplantmatching` version:
- The corresponding `config` entries changed from `estimate_renewable_capacities_from_capacity_stats` to `estimate_renewable_capacities`.
- The estimation is endabled by setting the subkey `enable` to `True`.
- Configuration of reference year for capacities can be configured (default: `2020`)
- The list of renewables provided by the OPSD database can be used as a basis, using the tag `from_opsd: True`. This adds the renewables from the database and fills up the missing capacities with the heuristic distribution.
- Uniform expansion limit of renewable build-up based on existing capacities can be configured using `expansion_limit` option (default: `false`; limited to determined renewable potentials)
- Distribution of country-level capacities proportional to maximum annual energy yield for each bus region

- The config key `renewable_capacities_from_OPSD` is deprecated and was moved under the section, `estimate_renewable_capacities`. To enable it, set `from_opsd` to `True`.
- Add operational reserve margin constraint analogous to GenX implementation. Can be activated with config setting `electricity: operational_reserve:`.
- Implement country-specific Energy Availability Factors (EAFs) for nuclear power plants based on IAEA 2018-2020 reported country averages. These are specified `data/nuclear_p_max_pu.csv` and translate to static `p_max_pu` values.
- Add function to add global constraint on use of gas in *`prepare_network`*. This can be activated by including the keyword `CH4L` in the `{opts}` wildcard which enforces the limit set in `electricity: gaslimit:` given in MWh thermal. Alternatively, it is possible to append a number in the `{opts}` wildcard, e.g. `CH4L200` which limits the gas use to 200 TWh thermal.
- Add option to alter marginal costs of a carrier through `{opts}` wildcard: `<carrier>+m<factor>`, e.g. `gas+m2.5`, will multiply the default marginal cost for gas by factor 2.5.
- Hierarchical clustering was introduced. Distance metric is calculated from renewable potentials on hourly (feature entry ends with `-time`) or annual (feature entry in config end with `-cap`) values.
- Greedy modularity clustering was introduced. Distance metric is based on electrical distance taking into account the impedance of all transmission lines of the network.
- Techno-economic parameters of technologies (e.g. costs and efficiencies) will now be retrieved from a separate repository PyPSA/technology-data that collects assumptions from a variety of sources. It is activated by default with `enable: retrieve_cost_data: true` and controlled with `costs: year:` and `costs: version:`. The location of this data changed from `data/costs.csv` to `resources/costs.csv` [#184].
- A new section `conventional` was added to the config file. This section contains configurations for conventional carriers.
- Add configuration option to implement arbitrary generator attributes for conventional generation technologies.
- Add option to set CO2 emission prices through `{opts}` wildcard: `Ep<number>`, e.g. `Ep180`, will set the EUR/tCO2 price.

Changes

- Add an efficiency factor of 88.55% to offshore wind capacity factors as a proxy for wake losses. More rigorous modelling is planned [#277].
- Following discussion in #285 we have disabled the correction factor for solar PV capacity factors by default while satellite data is used. A correction factor of 0.854337 is recommended if reanalysis data like ERA5 is used.
- The default deployment density of AC- and DC-connected offshore wind capacity is reduced from 3 MW/sqkm to a more conservative estimate of 2 MW/sqkm [#280].
- The inclusion of renewable carriers is now specified in the config entry `renewable_carriers`. Before this was done by commenting/uncommenting sub-sections in the `renewable` config section.
- Now, all carriers that should be extendable have to be listed in the config entry `extendable_carriers`. Before, renewable carriers were always set to be extendable. For backwards compatibility, the workflow is still looking at the listed carriers under the `renewable` key. In the future, all of them have to be listed under `extendable_carriers`.
- It is now possible to set conventional power plants as extendable by adding them to the list of extendable `Generator` carriers in the config.
- Listing conventional carriers in `extendable_carriers` but not in `conventional_carriers`, sets the corresponding conventional power plants as extendable without a lower capacity bound of today's capacities.
- Now, conventional carriers have an assigned capital cost by default.

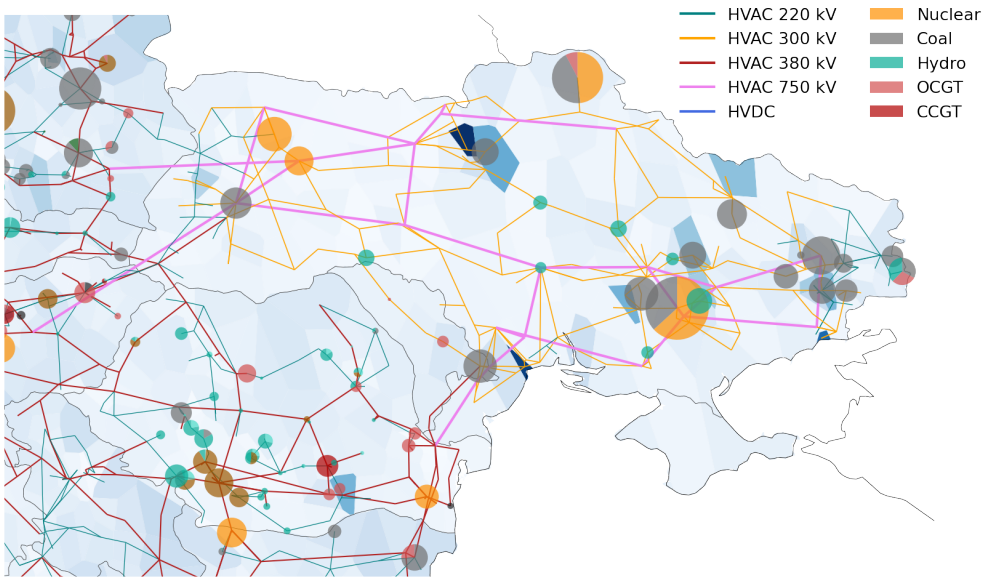
- The `build_year` and `lifetime` column are now defined for conventional power plants.
- Use updated SARAH-2 and ERA5 cutouts with slightly wider scope to east and additional variables.
- Resource definitions for memory usage now follow Snakemake standard resource definition `mem_mb` rather than `mem`.
- The powerplants that have been shut down by 2021 are filtered out.
- Updated historical EIA hydro generation data.
- Network building is made deterministic by supplying a fixed random state to network clustering routines.
- Clustering strategies for generator and bus attributes can now be specified directly in the `config/config.yaml`.
- Iterative solving with impedance updates is skipped if there are no expandable lines.
- The unused argument `simple_hvdc_costs` in *`add_electricity`* was removed.
- Switch from Germany to Belgium for continuous integration and tutorial to save resources.
- It is now possible to skip the progressbar for land eligibility calculations for additional speedup.

Bugs and Compatibility

- Fix crs bug. Change crs 4236 to 4326.
- `powerplantmatching>=0.5.1` is now required for `IRENASTATS`.
- Update rasterio version to correctly calculate exclusion raster.
- It is now possible to run the workflow with only landlocked countries.
- Bugfixes for manual load adjustments across years.
- Enable parallel computing with new dask version.
- Restore compatibility of `mock_snakemake` with latest Snakemake versions.
- Script `build_bus_regions`: move voronoi partition from vresutils to script.
- Script `add_electricity`: remove `vresutils.costdata.annuity` dependency.
- Fix the plot_network snakemake rule.
- Compatibility with pandas 1.4. Address deprecations.
- Restore Windows compatibility by using `shutil.move` rather than `mv`.

Synchronisation Release - Ukraine and Moldova (17th March 2022)

On March 16, 2022, the transmission networks of Ukraine and Moldova have successfully been synchronised with the continental European grid. We have taken this as an opportunity to add the power systems of Ukraine and Moldova to PyPSA-Eur. This includes:



- the transmission network topology from the ENTSO-E interactive map.
- existing power plants (incl. nuclear, coal, gas and hydro) from the powerplantmatching tool
- country-level load time series from ENTSO-E through the OPSD platform, which are then distributed heuristically to substations by GDP and population density.
- wind and solar profiles based on ERA5 and SARAH-2 weather data
- hydro profiles based on historical EIA generation data
- a simplified calculation of wind and solar potentials based on the Copernicus Land Cover dataset.
- electrical characteristics of 750 kV transmission lines

The Crimean power system is currently disconnected from the main Ukrainian grid and, hence, not included.

This release is not on the `master` branch. It can be used with

```
git clone https://github.com/pypsa/pypsa-eur
git checkout synchronisation-release
```

PyPSA-Eur 0.4.0 (22th September 2021)

New Features and Changes

- With this release, we change the license from copyleft GPLv3 to the more liberal MIT license with the consent of all contributors [#276].
- Switch to the new major `atlite` release v0.2. The version upgrade comes along with significant speed up for the rule `build_renewable_profiles.py` (~factor 2). A lot of the code which calculated the land-use availability is now outsourced and does not rely on `glaes`, `geokit` anymore. This facilitates the environment building and version compatibility of `gdal`, `libgdal` with other packages [#224].
- Implemented changes to `n.snapshot_weightings` in new PyPSA version v0.18 (cf. PyPSA/PyPSA/#227) [#259].
- Add option to pre-aggregate nodes without power injections (positive or negative, i.e. generation or demand) to electrically closest nodes or neighbors in `simplify_network`. Defaults to `False`. This affects nodes that are no substations or have no offshore connection.

- In *simplify_network*, bus columns with no longer correct entries are removed (symbol, tags, under_construction, substation_lv, substation_off) [#219]
- Add option to include marginal costs of links representing fuel cells, electrolysis, and battery inverters [#232].
- The rule and script `build_country_flh` are removed as they are no longer used or maintained.
- The connection cost of generators in *simplify_network* are now reported in `resources/connection_costs_s{simpl}.csv` [#261].
- The tutorial cutout was renamed from `cutouts/europe-2013-era5.nc` to `cutouts/be-03-2013-era5.nc` to accommodate tutorial and productive cutouts side-by-side.
- The flag `keep_all_available_areas` in the configuration for renewable potentials was deprecated and now defaults to `True`.
- Update dependencies in `envs/environment.yaml` [#257]
- Continuous integration testing switches to Github Actions from Travis CI [#252].
- Documentation on readthedocs.io is now built with `pip` only and no longer requires `conda` [#267].
- Use `Citation.cff` [#273].

Bugs and Compatibility

- Support for PyPSA v0.18 [#268].
- Minimum Python version set to `3.8`.
- Removed `six` dependency [#245].
- Update `plot_network` and *make_summary* rules to latest PyPSA versions [#270].
- Keep converter links to store components when using the `ATK` wildcard and only remove DC links [#214].
- Value for `co2base` in `config.yaml` adjusted to 1.487e9 t CO2-eq (from 3.1e9 t CO2-eq). The new value represents emissions related to the electricity sector for EU+UK+Balkan. The old value was too high and used when the emissions wildcard in `{opts}` was used [#233].
- Add escape in *base_network* if all TYNDP links are already contained in the network [#246].
- In *solve_operations_network* the optimised capacities are now fixed for all extendable links, not only HVDC links [#244].
- The `focus_weights` are now also considered when pre-clustering in the *simplify_network* rule [#241].
- in `build_renewable_profile` where offshore wind profiles could no longer be created [#249].
- Lower expansion limit of extendable carriers is now set to the existing capacity, i.e. `p_nom_min = p_nom` (0 before). Simultaneously, the upper limit (`p_nom_max`) is now the maximum of the installed capacity (`p_nom`) and the previous estimate based on land availability (`p_nom_max`) [#260].
- Solving an operations network now includes optimized store capacities as well. Before only lines, links, generators and storage units were considered [#269].
- With `load_shedding: true` in the solving options of `config.yaml` load shedding generators are only added at the AC buses, excluding buses for H2 and battery stores [#269].
- Delete duplicated capital costs at battery discharge link [#240].
- Propagate the solver log file name to the solver. Previously, the PyPSA network solving functions were not told about the solver logfile specified in the Snakemake file [#247]

PyPSA-Eur 0.3.0 (7th December 2020)

New Features

Using the `{opts}` wildcard for scenario:

- An option is introduced which adds constraints such that each country or node produces on average a minimal share of its total consumption itself. For example `EQ0.5c` set in the `{opts}` wildcard requires each country to produce on average at least 50% of its consumption. Additionally, the option `ATK` requires autarky at each node and removes all means of power transmission through lines and links. `ATKc` only removes cross-border transfer capacities. [#166].
- Added an option to alter the capital cost (`c`) or installable potentials (`p`) of carriers by a factor via `carrier+{c,p}factor` in the `{opts}` wildcard. This can be useful for exploring uncertain cost parameters. Example: `solar+c0.5` reduces the capital cost of solar to 50% of original values [#167, #207].
- Added an option to the `{opts}` wildcard that applies a time series segmentation algorithm based on renewables, hydro inflow and load time series to produce a given total number of adjacent snapshots of varying lengths. This feature is an alternative to downsampling the temporal resolution by simply averaging and uses the tsam package [#186].

More OPSD integration:

- Add renewable power plants from OPSD to the network for specified technologies. This will overwrite the capacities calculated from the heuristic approach in `estimate_renewable_capacities()` [#212].
- Electricity consumption data is now retrieved directly from the OPSD website using the rule *`build_electricity_demand`*. The user can decide whether to take the ENTSO-E power statistics data (default) or the ENTSO-E transparency data [#211].

Other:

- Added an option to use custom busmaps in rule *`cluster_network`*. To use this feature set `enable: custom_busmap: true`. Then, the rule looks for custom busmaps at `data/custom_busmap_elec_s{simpl}_{clusters}.csv`, which should have the same format as `resources/busmap_elec_s{simpl}_{clusters}.csv`. i.e. the index should contain the buses of `networks/elec_s{simpl}.nc` [#193].
- Line and link capacities can be capped in the `config.yaml` at `lines: s_nom_max:` and `links: p_nom_max:` [#166].
- Added Google Cloud Platform tutorial (for Windows users) [#177].

Changes

- Don't remove capital costs from lines and links, when imposing a line volume limit (`lv`) or a line cost limit (`lc`). Previously, these were removed to move the expansion in direction of the limit [#183].
- The mappings for clustered lines and buses produced by the *`simplify_network`* and *`cluster_network`* rules changed from Hierarchical Data Format (`.h5`) to Comma-Separated Values format (`.csv`) for ease of use. [#198]
- The N-1 security margin for transmission lines is now fixed to a provided value in `config.yaml`, removing an undocumented linear interpolation between 0.5 and 0.7 in the range between 37 and 200 nodes. [#199].
- Modelling hydrogen and battery storage with Store and Link components is now the default, rather than using StorageUnit components with fixed power-to-energy ratio [#205].
- Use `mamba` (https://github.com/mamba-org/mamba) for faster Travis CI builds [#196].
- Multiple smaller changes: Removed unused `{network}` wildcard, moved environment files to dedicated `envs` folder, removed sector-coupling components from configuration files, updated documentation colors, minor refactoring and code cleaning [#190].

Bugs and Compatibility

- Add compatibility for pyomo 5.7.0 in *cluster_network* and *simplify_network* [#172].
- Fixed a bug for storage units such that individual store and dispatch efficiencies are correctly taken account of rather than only their round-trip efficiencies. In the cost database (`data/costs.csv`) the efficiency of battery inverters should be stated as per discharge/charge rather than per roundtrip [#202].
- Corrected exogenous emission price setting (in `config: cost: emission price:`), which now correctly accounts for the efficiency and effective emission of the generators [#171].
- Corrected HVDC link connections (a) between Norway and Denmark and (b) mainland Italy, Corsica (FR) and Sardinia (IT) as well as for East-Western and Anglo-Scottish interconnectors [#181, #206].
- Fix bug of clustering `offwind-{ac,dc}` generators in the option of high-resolution generators for renewables. Now, there are more sites for `offwind-{ac,dc}` available than network nodes. Before, they were clustered to the resolution of the network (`elec_s1024_37m.nc`: 37 network nodes, 1024 generators) [#191].
- Raise a warning if `tech_colors` in the config are not defined for all carriers [#178].

PyPSA-Eur 0.2.0 (8th June 2020)

- The optimization is now performed using the `pyomo=False` setting in the `pypsa.lopf.network_lopf()`. This speeds up the solving process significantly and consumes much less memory. The inclusion of additional constraints were adjusted to the new implementation. They are all passed to the `network_lopf()` function via the `extra_functionality` argument. The rule `trace_solve_network` was integrated into the rule *solve_network* and can be activated via configuration with `solving: options: track_iterations: true`. The charging and discharging capacities of batteries modelled as store-link combination are now coupled [#116].
- An updated extract of the ENTSO-E Transmission System Map (including Malta) was added to the repository using the GridKit tool. This tool has been updated to retrieve up-to-date map extracts using a single script. The update extract features 5322 buses, 6574 lines, 46 links. [#118].
- Added FSFE REUSE compliant license information. Documentation now licensed under CC-BY-4.0 [#160].
- Added a 30 minute video introduction and a 20 minute video tutorial
- Networks now store a color and a nicely formatted name for each carrier, accessible via `n.carrier['color']` and `n.carrier['nice_name'] ``(networks after ``elec.nc`).
- Added an option to skip iterative solving usually performed to update the line impedances of expanded lines at `solving: options: skip_iterations:`.
- `snakemake` rules for retrieving cutouts and the natura raster can now be disabled independently from their respective rules to build them; via `config.*yaml` [#136].
- Removed the `id` column for custom power plants in `data/custom_powerplants.csv` to avoid custom power plants with conflicting ids getting attached to the wrong bus [#131].
- Add option `renewables: {carrier}: keep_all_available_areas:` to use all available weather cells for renewable profile and potential generation. The default ignores weather cells where only less than 1 MW can be installed [#150].
- Added a function `_helpers.load_network()` which loads a network with overridden components specified in `snakemake.config['override_components']` [#128].
- Bugfix in *base_network* which now finds all closest links, not only the first entry [#143].
- Bugfix in *cluster_network* which now skips recalculation of link parameters if there are no links [#149].
- Added information on pull requests to contribution guidelines [#151].

- Improved documentation on open-source solver setup and added usage warnings.
- Updated `conda` environment regarding `pypsa`, `pyproj`, `gurobi`, `lxml`. This release requires PyPSA v0.17.0.

PyPSA-Eur 0.1.0 (9th January 2020)

This is the first release of PyPSA-Eur, a model of the European power system at the transmission network level. Recent changes include:

- Documentation on installation, workflows and configuration settings is now available online at pypsa-eur.readthedocs.io [#65].
- The `conda` environment files were updated and extended [#81].
- The power plant database was updated with extensive filtering options via `pandas.query` functionality [#84 and #94].
- Continuous integration testing with Travis CI is now included for Linux, Mac and Windows [#82].
- Data dependencies were moved to zenodo and are now versioned [#60].
- Data dependencies are now retrieved directly from within the snakemake workflow [#86].
- Emission prices can be added to marginal costs of generators through the keywords `Ep` in the `{opts}` wildcard [#100].
- An option is introduced to add extendable nuclear power plants to the network [#98].
- Focus weights can now be specified for particular countries for the network clustering, which allows to set a proportion of the total number of clusters for particular countries [#87].
- A new rule *`add_extra_components`* allows to add additional components to the network only after clustering. It is thereby possible to model storage units (e.g. battery and hydrogen) in more detail via a combination of `Store`, `Link` and `Bus` elements [#97].
- Hydrogen pipelines (including cost assumptions) can now be added alongside clustered network connections in the rule *`add_extra_components`* . Set `electricity: extendable_carriers: Link: [H2 pipeline]` and ensure hydrogen storage is modelled as a `Store`. This is a first simplified stage [#108].
- Logfiles for all rules of the `snakemake` workflow are now written in the folder `log/` [#102].
- The new function `_helpers.mock_snakemake` creates a `snakemake` object which mimics the actual `snakemake` object produced by workflow by parsing the `Snakefile` and setting all paths for inputs, outputs, and logs. This allows running all scripts within a (I)python terminal (or just by calling `python <script-name>`) and thereby facilitates developing and debugging scripts significantly [#107].

7.1.7 PyPSA-Eur-Sec Releases (pre-merge)

PyPSA-Eur-Sec 0.7.0 (16th February 2023)

This release includes many new features. Highlights include new gas infrastructure data with retrofitting options for hydrogen transport, improved carbon management and infrastructure planning, regionalised potentials for hydrogen underground storage and carbon sequestration, new applications for biomass, and explicit modelling of methanol and ammonia as separate energy carriers.

This release is known to work with PyPSA-Eur Version 0.7.0 and Technology Data Version 0.5.0.

Gas Transmission Network

- New rule `retrieve_gas_infrastructure_data` that downloads and extracts the SciGRID_gas IGGIELGN dataset from zenodo. It includes data on the transmission routes, pipe diameters, capacities, pressure, and whether the pipeline is bidirectional and carries H-Gas or L-Gas.
- New rule `build_gas_network` processes and cleans the pipeline data from SciGRID_gas. Missing or uncertain pipeline capacities can be inferred by diameter.
- New rule `build_gas_input_locations` compiles the LNG import capacities (from the Global Energy Monitor's Europe Gas Tracker, pipeline entry capacities and local production capacities for each region of the model. These are the regions where fossil gas can eventually enter the model.
- New rule `cluster_gas_network` that clusters the gas transmission network data to the model resolution. Cross-regional pipeline capacities are aggregated (while pressure and diameter compatibility is ignored), intra-regional pipelines are dropped. Lengths are recalculated based on the regions' centroids.
- With the option `sector: gas_network:`, the existing gas network is added with a lossless transport model. A length-weighted k-edge augmentation algorithm can be run to add new candidate gas pipelines such that all regions of the model can be connected to the gas network. The number of candidates can be controlled via the setting `sector: gas_network_connectivity_upgrade:`. When the gas network is activated, all the gas demands are regionally disaggregated as well.
- New constraint allows endogenous retrofitting of gas pipelines to hydrogen pipelines. This option is activated via the setting `sector: H2_retrofit:`. For every unit of gas pipeline capacity dismantled, `sector: H2_retrofit_capacity_per_CH4` units are made available as hydrogen pipeline capacity in the corresponding corridor. These repurposed hydrogen pipelines have lower costs than new hydrogen pipelines. Both new and repurposed pipelines can be built simultaneously. The retrofitting option `sector: H2_retrofit:` also works with a copperplated methane infrastructure, i.e. when `sector: gas_network: false`.
- New hydrogen pipelines can now be built where there are already power or gas transmission routes. Previously, only the electricity transmission routes were considered.

Carbon Management and Biomass

- Add option to spatially resolve carrier representing stored carbon dioxide (`co2_spatial`). This allows for more detailed modelling of CCUTS, e.g. regarding the capturing of industrial process emissions, usage as feedstock for electrofuels, transport of carbon dioxide, and geological sequestration sites.
- Add option for regionally-resolved geological carbon dioxide sequestration potentials through new rule `build_sequestration_potentials` based on CO2StoP. This can be controlled in the section `regional_co2_sequestration_potential` of the `config.yaml`. It includes options to select the level of conservatism, whether onshore potentials should be included, the respective upper and lower limits per region, and an annualisation parameter for the cumulative potential. The defaults are preliminary and will be validated the next release.
- Add option to sweep the global CO2 sequestration potentials with keyword `seq200` in the `{sector_opts}` wildcard (for limit of 200 Mt CO2).
- Add option to include Allam cycle gas power plants (`allam_cycle`).
- Add option for planning a new carbon dioxide network (`co2network`).
- Separate option to regionally resolve biomass (`biomass_spatial`) from option to allow biomass transport (`biomass_transport`).
- Add option for biomass boilers (wood pellets) for decentral heating.
- Add option for BioSNG (methane from biomass) with and without carbon capture.
- Add option for BtL (biomass to liquid fuel/oil) with and without carbon capture.

Other new features

- Add regionalised hydrogen salt cavern storage potentials from Technical Potential of Salt Caverns for Hydrogen Storage in Europe. This data is compiled in a new rule `build_salt_cavern_potentials`.
- Add option to resolve ammonia as separate energy carrier with Haber-Bosch synthesis, ammonia cracking, storage and industrial demand. The ammonia carrier can be nodally resolved or copperplated across Europe (see `ammonia`).
- Add methanol as energy carrier, methanolisation as process, and option for methanol demand in shipping sector.
- Shipping demand now defaults to methanol rather than liquefied hydrogen until 2050.
- Demand for liquid hydrogen in international shipping is now geographically distributed by port trade volumes in a new rule `build_shipping_demand` using data from the World Bank Data Catalogue. Domestic shipping remains distributed by population.
- Add option to aggregate network temporally using representative snapshots or segments (with tsam).
- Add option for minimum part load for Fischer-Tropsch plants (default: 90%) and methanolisation plants (default: 50%).
- Add option to use waste heat of electrolysis in district heating networks (`use_electrolysis_waste_heat`).
- Add option for coal CHPs with carbon capture (see `coal_cc`).
- In overnight optimisation, it is now possible to specify a year for the technology cost projections separate from the planning horizon.
- New config options for changing energy demands in aviation (`aviation_demand_factor`) and HVC industry (`HVC_demand_factor`), as well as explicit ICE shares for land transport (`land_transport_ice_share`) and agriculture machinery (`agriculture_machinery_oil_share`).
- It is now possible to merge residential and services heat buses to reduce the problem size (see `cluster_heat_nodes`).
- Added option to tweak (almost) any configuration parameter through the `{sector_opts}` wildcard. The regional_co2_sequestration_potential is triggered by the prefix `CF+` after which it is possible to pipe to any setting that does not contain underscores (_). Example: `CF+sector+v2g+false` disables vehicle-to-grid flexibility.
- Option `retrieve_sector_databundle` to automatically retrieve and extract data bundle.
- Removed the need to clone `technology-data` repository in a parallel directory. The new approach automatically retrieves the technology data from remote in the rule `retrieve_cost_data`.
- Improved network plots including better legends, hydrogen retrofitting network display, and change to EqualEarth projection. A new color scheme for technologies was also introduced.
- Add two new rules `build_transport_demand` and `build_population_weighted_energy_totals` using code previously contained in `prepare_sector_network`.
- Rules that convert weather data with `atlite` now largely run separately for categories residential, rural and total.
- Units are assigned to the buses. These only provide a better understanding. The specifications of the units are not taken into account in the optimisation, which means that no automatic conversion of units takes place.
- Configuration file and wildcards are now stored under `n.meta` in every PyPSA network.
- Updated data bundle that includes the hydrogan salt cavern storage potentials.
- Updated and extended documentation in <https://pypsa-eur-sec.readthedocs.io/en/latest/>
- Added new rule `copy_conda_env` that exports a list of packages with which the workflow was executed.
- Add basic continuous integration using Github Actions.
- Add basic `rsync` setup.

Bugfixes

- The CO2 sequestration limit implemented as GlobalConstraint (introduced in the previous version) caused a failure to read in the shadow prices of other global constraints.
- Correct capital cost of Fischer-Tropsch according to new units in `technology-data` repository.
- Fix unit conversion error for thermal energy storage.
- For myopic pathway optimisation, set optimised capacities of power grid expansion of previous iteration as minimum capacity for next iteration.
- Further rather minor bugfixes for myopic optimisation code (see #256).

Many thanks to all who contributed to this release!

PyPSA-Eur-Sec 0.6.0 (4 October 2021)

This release includes improvements regarding the basic chemical production, the addition of plastics recycling, the addition of the agriculture, forestry and fishing sector, more regionally resolved biomass potentials, CO2 pipeline transport and storage, and more options in setting exogenous transition paths, besides many performance improvements.

This release is known to work with PyPSA-Eur Version 0.4.0, Technology Data Version 0.3.0 and PyPSA Version 0.18.0.

Please note that the data bundle has also been updated.

General

- With this release, we change the license from copyleft GPLv3 to the more liberal MIT license with the consent of all contributors.

New features and functionality

- Distinguish costs for home battery storage and inverter from utility-scale battery costs.
- Separate basic chemicals into HVC (high-value chemicals), chlorine, methanol and ammonia [#166].
- Add option to specify reuse, primary production, and mechanical and chemical recycling fraction of platics [#166].
- Include energy demands and CO2 emissions for the agriculture, forestry and fishing sector. It is included by default through the option `A` in the `sector_opts` wildcard. Part of the emissions (1.A.4.c) was previously assigned to "industry non-elec" in the `co2_totals.csv`. Hence, excluding the agriculture sector will now lead to a tighter CO2 limit. Energy demands are taken from the JRC IDEES database (missing countries filled with eurostat data) and are split into electricity (lighting, ventilation, specific electricity uses, pumping devices (electric)), heat (specific heat uses, low enthalpy heat) machinery oil (motor drives, farming machine drives, pumping devices (diesel)). Heat demand is assigned at "services rural heat" buses. Electricity demands are added to low-voltage buses. Time series for demands are constant and distributed inside countries by population [#147].
- Include today's district heating shares in myopic optimisation and add option to specify exogenous path for district heating share increase under `sector: district_heating:` [#149].
- Added option for hydrogen liquefaction costs for hydrogen demand in shipping. This introduces a new `H2 liquid` bus at each location. It is activated via `sector: shipping_hydrogen_liquefaction: true`.
- The share of shipping transformed into hydrogen fuel cell can be now defined for different years in the `config.yaml` file. The carbon emission from the remaining share is treated as a negative load on the atmospheric carbon dioxide bus, just like aviation and land transport emissions.
- The transformation of the Steel and Aluminium production can be now defined for different years in the `config.yaml` file.

- Include the option to alter the maximum energy capacity of a store via the `carrier+factor` in the `{sector_opts}` wildcard. This can be useful for sensitivity analyses. Example: `co2 stored+e2` multiplies the `e_nom_max` by factor 2. In this example, `e_nom_max` represents the CO2 sequestration potential in Europe.
- Use JRC ENSPRESO database to spatially disaggregate biomass potentials to PyPSA-Eur regions based on overlaps with NUTS2 regions from ENSPRESO (proportional to area) (#151).
- Add option to regionally disaggregate biomass potential to individual nodes (previously given per country, then distributed by population density within) and allow the transport of solid biomass. The transport costs are determined based on the JRC-EU-Times Bioenergy report in the new optional rule `build_biomass_transport_costs`. Biomass transport can be activated with the setting `sector: biomass_transport: true`.
- Add option to regionally resolve CO2 storage and add CO2 pipeline transport because geological storage potential, CO2 utilisation sites and CO2 capture sites may be separated. The CO2 network is built from zero based on the topology of the electricity grid (greenfield). Pipelines are assumed to be bidirectional and lossless. Furthermore, neither retrofitting of natural gas pipelines (required pressures are too high, 80-160 bar vs <80 bar) nor other modes of CO2 transport (by ship, road or rail) are considered. The regional representation of CO2 is activated with the config setting `sector: co2_network: true` but is deactivated by default. The global limit for CO2 sequestration now applies to the sum of all CO2 stores via an `extra_functionality` constraint.
- The myopic option can now be used together with different clustering for the generators and the network. The existing renewable capacities are split evenly among the regions in every country [#144].
- Add optional function to use `geopy` to locate entries of the Hotmaps database of industrial sites with missing location based on city and country, which reduces missing entries by half. It can be activated by setting `industry: hotmaps_locate_missing: true`, takes a few minutes longer, and should only be used if spatial resolution is coarser than city level.

Performance and Structure

- Extended use of `multiprocessing` for much better performance (from up to 20 minutes to less than one minute).
- Handle most input files (or base directories) via `snakemake.input`.
- Use of `mock_snakemake` from PyPSA-Eur.
- Update `solve_network` rule to match implementation in PyPSA-Eur by using `n.ilopf()` and remove outdated code using `pyomo`. Allows the new setting to skip iterated impedance updates with `solving: options: skip_iterations: true`.
- The component attributes that are to be overridden are now stored in the folder `data/override_component_attrs` analogous to `pypsa/component_attrs`. This reduces verbosity and also allows circumventing the `n.madd()` hack for individual components with non-default attributes. This data is also tracked in the Snakefile. A function `helper.override_component_attrs` was added that loads this data and can pass the overridden component attributes into `pypsa.Network()`.
- Add various parameters to `config.default.yaml` which were previously hardcoded inside the scripts (e.g. energy reference years, BEV settings, solar thermal collector models, geomap colours).
- Removed stale industry demand rules `build_industrial_energy_demand_per_country` and `build_industrial_demand`. These are superseded with more regionally resolved rules.
- Use simpler and shorter `gdf.sjoin()` function to allocate industrial sites from the Hotmaps database to onshore regions. This change also fixes a bug: The previous version allocated sites to the closest bus, but at country borders (where Voronoi cells are distorted by the borders), this had resulted in e.g. a Spanish site close to the French border being wrongly allocated to the French bus if the bus center was closer.
- Retrofitting rule is now only triggered if endogeneously optimised.
- Show progress in build rules with `tqdm` progress bars.

- Reduced verbosity of `Snakefile` through directory prefixes.
- Improve legibility of `config.default.yaml` and remove unused options.
- Use the country-specific time zone mappings from `pytz` rather than a manual mapping.
- A function `add_carrier_buses()` was added to the `prepare_network` rule to reduce code duplication.
- In the `prepare_network` rule the cost and potential adjustment was moved into an own function `maybe_adjust_costs_and_potentials()`.
- Use `matplotlibrc` to set the default plotting style and backend.
- Added benchmark files for each rule.
- Consistent use of `__main__` block and further unspecific code cleaning.
- Updated data bundle and moved data bundle to zenodo.org (10.5281/zenodo.5546517).

Bugfixes and Compatibility

- Compatibility with `atlite>=0.2`. Older versions of `atlite` will no longer work.
- Corrected calculation of "gas for industry" carbon capture efficiency.
- Implemented changes to `n.snapshot_weightings` in PyPSA v0.18.0.
- Compatibility with `xarray` version 0.19.
- New dependencies: `tqdm`, `atlite>=0.2.4`, `pytz` and `geopy` (optional). These are included in the environment specifications of PyPSA-Eur v0.4.0.

Many thanks to all who contributed to this release!

PyPSA-Eur-Sec 0.5.0 (21st May 2021)

This release includes improvements to the cost database for building retrofits, carbon budget management and wildcard settings, as well as an important bugfix for the emissions from land transport.

This release is known to work with PyPSA-Eur Version 0.3.0 and Technology Data Version 0.2.0.

Please note that the data bundle has also been updated.

New features and bugfixes:

- The cost database for retrofitting of the thermal envelope of buildings has been updated. Now, for calculating the space heat savings of a building, losses by thermal bridges and ventilation are included as well as heat gains (internal and by solar radiation). See the section retro for more details on the retrofitting module.
- For the myopic investment option, a carbon budget and a type of decay (exponential or beta) can be selected in the `config.yaml` file to distribute the budget across the `planning_horizons`. For example, `cb40ex0` in the `{sector_opts}` wildcard will distribute a carbon budget of 40 GtCO2 following an exponential decay with initial growth rate 0.
- Added an option to alter the capital cost or maximum capacity of carriers by a factor via `carrier+factor` in the `{sector_opts}` wildcard. This can be useful for exploring uncertain cost parameters. Example: `solar+c0.5` reduces the `capital_cost` of solar to 50% of original values. Similarly `solar+p3` multiplies the `p_nom_max` by 3.
- Rename the bus for European liquid hydrocarbons from `Fischer-Tropsch` to `EU oil`, since it can be supplied not just with the Fischer-Tropsch process, but also with fossil oil.
- Bugfix: The new separation of land transport by carrier in Version 0.4.0 failed to account for the carbon dioxide emissions from internal combustion engines in land transport. This is now treated as a negative load on the atmospheric carbon dioxide bus, just like aviation emissions.

- Bugfix: Fix reading in of `pypsa-eur/resources/powerplants.csv` to PyPSA-Eur Version 0.3.0 (use column attribute name `DateIn` instead of old `YearDecommissioned`).
- Bugfix: Make sure that `Store` components (battery and H2) are also removed from PyPSA-Eur, so they can be added later by PyPSA-Eur-Sec.

Thanks to Lisa Zeyen (KIT) for the retrofitting improvements and Marta Victoria (Aarhus University) for the carbon budget and wildcard management.

PyPSA-Eur-Sec 0.4.0 (11th December 2020)

This release includes a more accurate nodal disaggregation of industry demand within each country, fixes to CHP and CCS representations, as well as changes to some configuration settings.

It has been released to coincide with PyPSA-Eur Version 0.3.0 and Technology Data Version 0.2.0, and is known to work with these releases.

New features:

- The Hotmaps Industrial Database is used to disaggregate the industrial demand spatially to the nodes inside each country (previously it was distributed by population density).
- Electricity demand from industry is now separated from the regular electricity demand and distributed according to the industry demand. Only the remaining regular electricity demand for households and services is distributed according to GDP and population.
- A cost database for the retrofitting of the thermal envelope of residential and services buildings has been integrated, as well as endogenous optimisation of the level of retrofitting. This is described in the paper Mitigating heat demand peaks in buildings in a highly renewable European energy system. Retrofitting can be activated both exogenously and endogenously from the `config.yaml`.
- The biomass and gas combined heat and power (CHP) parameters `c_v` and `c_b` were read in assuming they were extraction plants rather than back pressure plants. The data is now corrected in Technology Data Version 0.2.0 to the correct DEA back pressure assumptions and they are now implemented as single links with a fixed ratio of electricity to heat output (even as extraction plants, they were always sitting on the backpressure line in simulations, so there was no point in modelling the full heat-electricity feasibility polygon). The old assumptions underestimated the heat output.
- The Danish Energy Agency released new assumptions for carbon capture in October 2020, which have now been incorporated in PyPSA-Eur-Sec, including direct air capture (DAC) and post-combustion capture on CHPs, cement kilns and other industrial facilities. The electricity and heat demand for DAC is modelled for each node (with heat coming from district heating), but currently the electricity and heat demand for industrial capture is not modelled very cleanly (for process heat, 10% of the energy is assumed to go to carbon capture) - a new issue will be opened on this.
- Land transport is separated by energy carrier (fossil, hydrogen fuel cell electric vehicle, and electric vehicle), but still needs to be separated into heavy and light vehicles (the data is there, just not the code yet).
- For assumptions that change with the investment year, there is a new time-dependent format in the `config.yaml` using a dictionary with keys for each year. Implemented examples include the CO2 budget, exogenous retrofitting share and land transport energy carrier; more parameters will be dynamised like this in future.
- Some assumptions have been moved out of the code and into the `config.yaml`, including the carbon sequestration potential and cost, the heat pump sink temperature, reductions in demand for high value chemicals, and some BEV DSM parameters and transport efficiencies.
- Documentation on *Supply and demand* options has been added.

Many thanks to Fraunhofer ISI for opening the hotmaps database and to Lisa Zeyen (KIT) for implementing the building retrofitting.

PyPSA-Eur-Sec 0.3.0 (27th September 2020)

This releases focuses on improvements to industry demand and the generation of intermediate files for demand for basic materials. There are still inconsistencies with CCS and waste management that need to be improved.

It is known to work with PyPSA-Eur v0.1.0 (commit bb3477cd69), PyPSA v0.17.1 and technology-data v0.1.0. Please note that the data bundle has also been updated.

New features:

- In previous version of PyPSA-Eur-Sec the energy demand for industry was calculated directly for each location. Now, instead, the production of each material (steel, cement, aluminium) at each location is calculated as an intermediate data file, before the energy demand is calculated from it. This allows us in future to have competing industrial processes for supplying the same material demand.
- The script `build_industrial_production_per_country_tomorrow.py` determines the future industrial production of materials based on today's levels as well as assumed recycling and demand change measures.
- The energy demand for each industry sector and each location in 2015 is also calculated, so that it can be later incorporated in the pathway optimization.
- Ammonia production data is taken from the USGS and deducted from JRC-IDEES's "basic chemicals" so that it ammonia can be handled separately from the others (olefins, aromatics and chlorine).
- Solid biomass is no longer allowed to be used for process heat in cement and basic chemicals, since the wastes and residues cannot be guaranteed to reach the high temperatures required. Instead, solid biomass is used in the paper and pulp as well as food, beverages and tobacco industries, where required temperatures are lower (see DOI:10.1002/er.3436 and DOI:10.1007/s12053-017-9571-y).
- National installable potentials for salt caverns are now applied.
- When electricity distribution grids are activated, new industry electricity demand, resistive heaters and micro-CHPs are now connected to the lower voltage levels.
- Gas distribution grid costs are included for gas boilers and micro-CHPs.
- Installable potentials for rooftop PV are included with an assumption of 1 kWp per person.
- Some intermediate files produced by scripts have been moved from the folder `data` to the folder `resources`. Now `data` only includes input data, while `resources` only includes intermediate files necessary for building the network models. Please note that the data bundle has also been updated.
- Biomass potentials for different years and scenarios from the JRC are generated in an intermediate file, so that a selection can be made more explicitly by specifying the biomass types from the `config.yaml`.

PyPSA-Eur-Sec 0.2.0 (21st August 2020)

This release introduces pathway optimization over many years (e.g. 2020, 2030, 2040, 2050) with myopic foresight, as well as outsourcing the technology assumptions to the technology-data repository.

It is known to work with PyPSA-Eur v0.1.0 (commit bb3477cd69), PyPSA v0.17.1 and technology-data v0.1.0.

New features:

- Option for pathway optimization with myopic foresight, based on the paper Early decarbonisation of the European Energy system pays off (2020). Investments are optimized sequentially for multiple years (e.g. 2020, 2030, 2040, 2050) taking account of existing assets built in previous years and their lifetimes. The script uses data on the existing assets for electricity and building heating technologies, but there are no assumptions yet for existing transport and industry (if you include these, the model will greenfield them). There are also some outstanding issues on e.g. the distribution of existing wind, solar and heating technologies within each country. To use myopic foresight, set `foresight : 'myopic'` in the `config.yaml` instead of the default `foresight`

`: 'overnight'`. An example configuration can be found in `config.myopic.yaml`. More details on the implementation can be found in myopic.

- Technology assumptions (costs, efficiencies, etc.) are no longer stored in the repository. Instead, you have to install the technology-data database in a parallel directory. These assumptions are largely based on the Danish Energy Agency Technology Data. More details on the installation can be found in *Installation*.
- Logs and benchmarks are now stored with the other model outputs in `results/run-name/`.
- All buses now have a `location` attribute, e.g. bus `DE0 3 urban central heat` has a `location` of `DE0 3`.
- All assets have a `lifetime` attribute (integer in years). For the myopic foresight, a `build_year` attribute is also stored.
- Costs for solar and onshore and offshore wind are recalculated by PyPSA-Eur-Sec based on the investment year, including the AC or DC connection costs for offshore wind.

Many thanks to Marta Victoria for implementing the myopic foresight, and Marta Victoria, Kun Zhu and Lisa Zeyen for developing the technology assumptions database.

PyPSA-Eur-Sec 0.1.0 (8th July 2020)

This is the first proper release of PyPSA-Eur-Sec, a model of the European energy system at the transmission network level that covers the full ENTSO-E area.

It is known to work with PyPSA-Eur v0.1.0 (commit bb3477cd69) and PyPSA v0.17.0.

We are making this release since in version 0.2.0 we will introduce changes to allow myopic investment planning that will require minor changes for users of the overnight investment planning.

PyPSA-Eur-Sec builds on the electricity generation and transmission model PyPSA-Eur to add demand and supply for the following sectors: transport, space and water heating, biomass, industry and industrial feedstocks. This completes the energy system and includes all greenhouse gas emitters except waste management, agriculture, forestry and land use.

PyPSA-Eur-Sec was initially based on the model PyPSA-Eur-Sec-30 (Version 0.0.1 below) described in the paper Synergies of sector coupling and transmission reinforcement in a cost-optimised, highly renewable European energy system (2018) but it differs by being based on the higher resolution electricity transmission model PyPSA-Eur rather than a one-node-per-country model, and by including biomass, industry, industrial feedstocks, aviation, shipping, better carbon management, carbon capture and usage/sequestration, and gas networks.

PyPSA-Eur-Sec includes PyPSA-Eur as a snakemake subworkflow. PyPSA-Eur-Sec uses PyPSA-Eur to build the clustered transmission model along with wind, solar PV and hydroelectricity potentials and time series. Then PyPSA-Eur-Sec adds other conventional generators, storage units and the additional sectors.

PyPSA-Eur-Sec 0.0.2 (4th September 2020)

This version, also called PyPSA-Eur-Sec-30-Path, built on PyPSA-Eur-Sec 0.0.1 (also called PyPSA-Eur-Sec-30) to include myopic pathway optimisation for the paper Early decarbonisation of the European energy system pays off (2020). The myopic pathway optimisation was then merged into the main PyPSA-Eur-Sec codebase in Version 0.2.0 above.

This model has its own github repository and is archived on Zenodo.

PyPSA-Eur-Sec 0.0.1 (12th January 2018)

This is the first published version of PyPSA-Eur-Sec, also called PyPSA-Eur-Sec-30. It was first used in the research paper Synergies of sector coupling and transmission reinforcement in a cost-optimised, highly renewable European energy system (2018). The model covers 30 European countries with one node per country. It includes demand and supply for electricity, space and water heating in buildings, and land transport.

It is archived on Zenodo.

7.1.8 Release Process

- Checkout a new release branch `git checkout -b release-v0.x.x`.
- Finalise release notes at `doc/release_notes.rst`.
- Update `envs/environment.fixed.yaml` via `conda env export -n pypsa-eur -f envs/environment.fixed.yaml --no-builds` from an up-to-date `pypsa-eur` environment.
- Update version number in `doc/conf.py`, `CITATION.cff` and `*config.*.yaml`.
- Make a `git commit`.
- Open, review and merge pull request for branch `release-v0.x.x`. Make sure to close issues and PRs or the release milestone with it (e.g. closes #X).
- Tag a release on Github via `git tag v0.x.x`, `git push`, `git push --tags`. Include release notes in the tag message.
- Make a GitHub release, which automatically triggers archiving to the zenodo code repository with MIT license.
- Send announcement on the PyPSA mailing list.

7.2 Licenses

PyPSA-Eur is released under multiple licenses:

- All original source code is licensed as free software under MIT.
- The documentation is licensed under CC-BY-4.0.
- Configuration files are mostly licensed under CC0-1.0.
- Data files are licensed under CC-BY-4.0.

See the individual files and the dep5 file for license details.

Additionally, different licenses and terms of use also apply to the various input data for both electricity-only and sector-coupled modelling exercises, which are summarised below.

7.2.1 Electricity Systems Databundle

Note: More details are included in the description of the data bundles on zenodo.

Files	BY	NC	SA	Mark Changes	Detail
corine/*	x			x	https://land.copernicus.eu/pan-european/corine-land-cover/clc-2012?tab=metadata
eez/*	x	x	x		http://www.marineregions.org/disclaimer.php
natura/*	x				https://www.eea.europa.eu/data-and-maps/data/natura-10#tab-metadata
naturalearth/*					http://www.naturalearthdata.com/about/terms-of-use/
NUTS_2013_60M_SH/*	x	x		x	https://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units
cantons.csv	x		x		https://en.wikipedia.org/wiki/Data_codes_for_Switzerland
eia_hydro_annual_genera	x				https://www.eia.gov/about/copyrights_reuse.php
GEBCO_2014_2D.nc	x				https://www.gebco.net/data_and_products/gridded_bathymetry_data/documents/gebco_2014_historic.pdf
hydro_capacities.csv	x				
je-e-21.03.02.xls	x	x			https://www.bfs.admin.ch/bfs/en/home/fso/swiss-federal-statistical-office/terms-of-use.html
nama_10r_3 gdp.tsv.gz	x			x	https://ec.europa.eu/eurostat/about/policies/copyright
nama_10r_3 popgdp.tsv.gz	x			x	https://ec.europa.eu/eurostat/about/policies/copyright
time_series_60min_singlein-dex_filtered.csv	x				https://data.open-power-system-data.org/time_series/2019-06-05/README.md

- BY: Attribute Source
- NC: Non-Commercial Use Only
- SA: Share Alike

7.2.2 Sector-Coupled Systems Databundle

description	file/folder	licence	source
JRC IDEES database	jrc-idees-2015/	CC BY 4.0	https://ec.europa.eu/jrc/en/potencia/jrc-idees
urban/rural fraction	urban_percent.csv	unknown	unknown
JRC biomass potentials	biomass/	unknown	https://doi.org/10.2790/39014
JRC ENSPRESO biomass potentials	remote	CC BY 4.0	https://data.jrc.ec.europa.eu/dataset/74ed5a04-7d74-4807-9eab-b94774309d9f
EEA emission statistics	eea/UNFCCC_v23.csv	EEA standard re-use policy	https://www.eea.europa.eu/data-and-maps/data/national-emissions-reported-to-the-unfccc-and-to- …u-greenhouse-gas-
Eurostat Energy Balances	eurostat-energy_balances-*/	Eurostat	https://ec.europa.eu/eurostat/web/energy/data/energy-balances
Swiss energy statistics from Swiss Federal Office of Energy	switzerland-sfoe/	unknown	http://www.bfe.admin.ch/themen/00526/00541/00542/02167/index.html?dossier_id=02169
BASt emobility statistics	emobility/	unknown	http://www.bast.de/DE/Verkehrstechnik/Fachthemen/v2-verkehrszaehlung/Stundenwerte.html?nn=626916
BDEW heating profile	heat_load_profile_BDE	unknown	https://github.com/oemof/demandlib
heating profiles for Aarhus	heat_load_profile_DK_	unknown	Adam Jensen MA thesis at Aarhus University
co2 budgets	co2_budget.csv	CC BY 4.0	https://arxiv.org/abs/2004.11009
existing heating potentials	existing_infrastructure/existi	unknown	https://energy.ec.europa.eu/publications/mapping-and-analyses-current-and-future-2020-2(…eatingcooling-fuel en
IRENA existing VRE capacities	existing_infrastructure/{sola	unknown	https://www.irena.org/Statistics/Download-Data
USGS ammonia production	myb1-2017-nitro.xls	unknown	https://www.usgs.gov/centers/nmic/nitrogen-statistics-and-information
hydrogen salt cavern potentials	h2_salt_caverns_GWh_	CC BY 4.0	https://doi.org/10.1016/j.ijhydene.2019.12.161 https://doi.org/10.20944/preprints201910.0187.v1
international port trade volumes	attributed_ports.json	CC BY 4.0	https://datacatalog.worldbank.org/search/dataset/0(…3/Global—Interna Ports
hotmaps industrial site database	Industrial_Database.csv	CC BY 4.0	https://gitlab.com/hotmaps/industrial_sites/industrial_sites_Industrial_Database
Hotmaps building stock data	data_building_stock.csv	CC BY 4.0	https://gitlab.com/hotmaps/building-stock
U-values Poland	u_values_poland.csv	unknown	https://data.europa.eu/euodp/de/data/dataset/building-stock-observatory
Floor area missing in hotmaps building stock data	floor_area_missing.csv	unknown	https://data.europa.eu/euodp/de/data/dataset/building-stock-observatory
Comparative level investment	comparative_level_investment.c	Eurostat	https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Comparative_price_levels_for_investment
Electricity taxes	electricity_taxes_eu.csv	Eurostat	https://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=nrg_pc_204&lang=en
Building topologies and corresponding standard	tabula-calculator-calcsetbuilding.csv	unknown	https://episcope.eu/fileadmin/tabula/public/calc/tabula-calculator.xlsx

7.3 Validation

The PyPSA-Eur model workflow provides a built-in mechanism for validation. This allows users to contrast the outcomes of network optimization against the historical behaviour of the European power system. The snakemake rule `validate_elec_networks` enables this by generating comparative figures that encapsulate key data points such as dispatch carrier, cross-border flows, and market prices per price zone.

These comparisons utilize data from the 2019 ENTSO-E Transparency Platform. To enable this, an ENTSO-E API key must be inserted into the `config.yaml` file. Detailed steps for this process can be found in the user guide here.

Once the API key is set, the validation workflow can be triggered by running the following command:

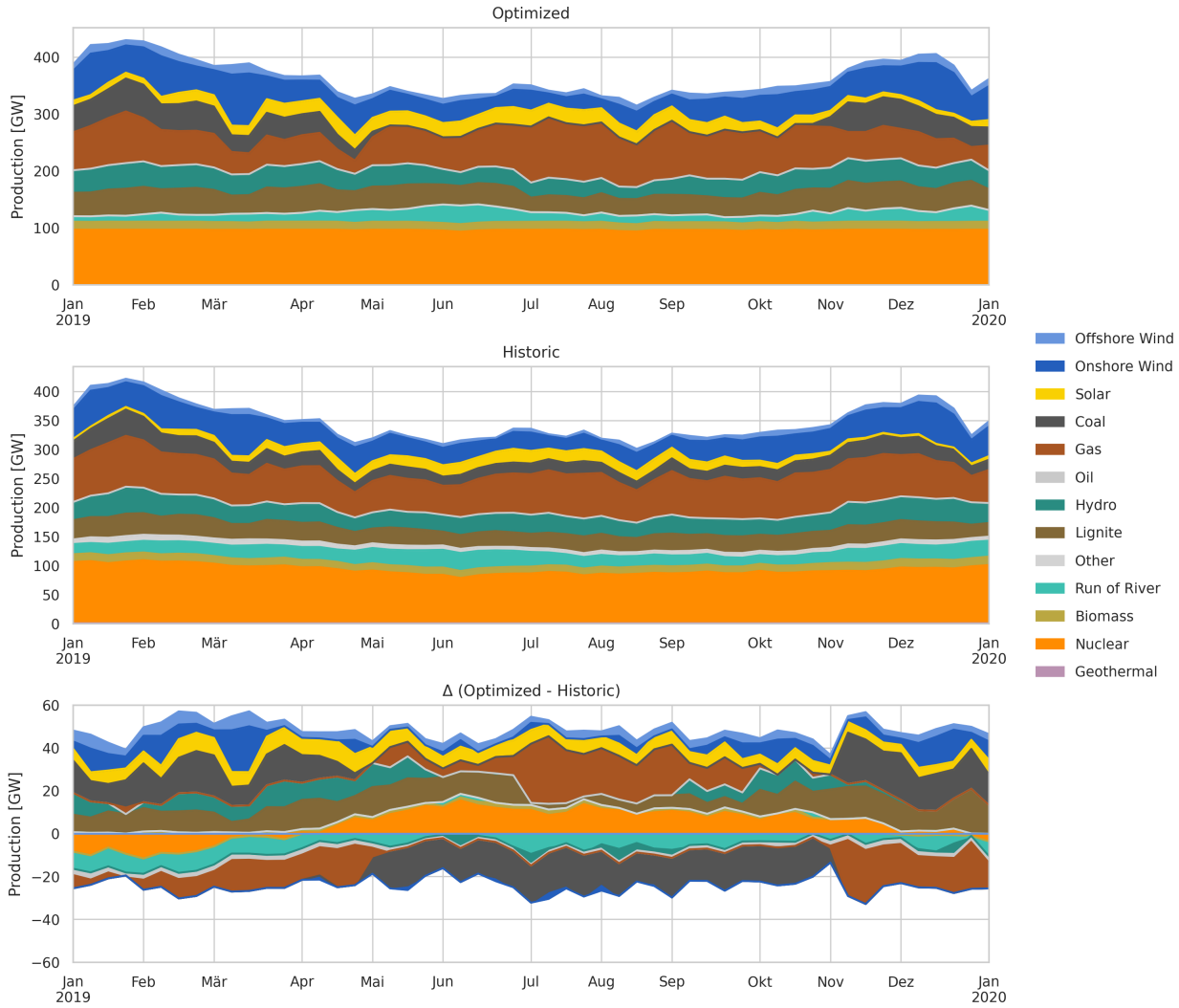
snakemake validate_elec_networks –configfile config/config.validation.yaml -c8

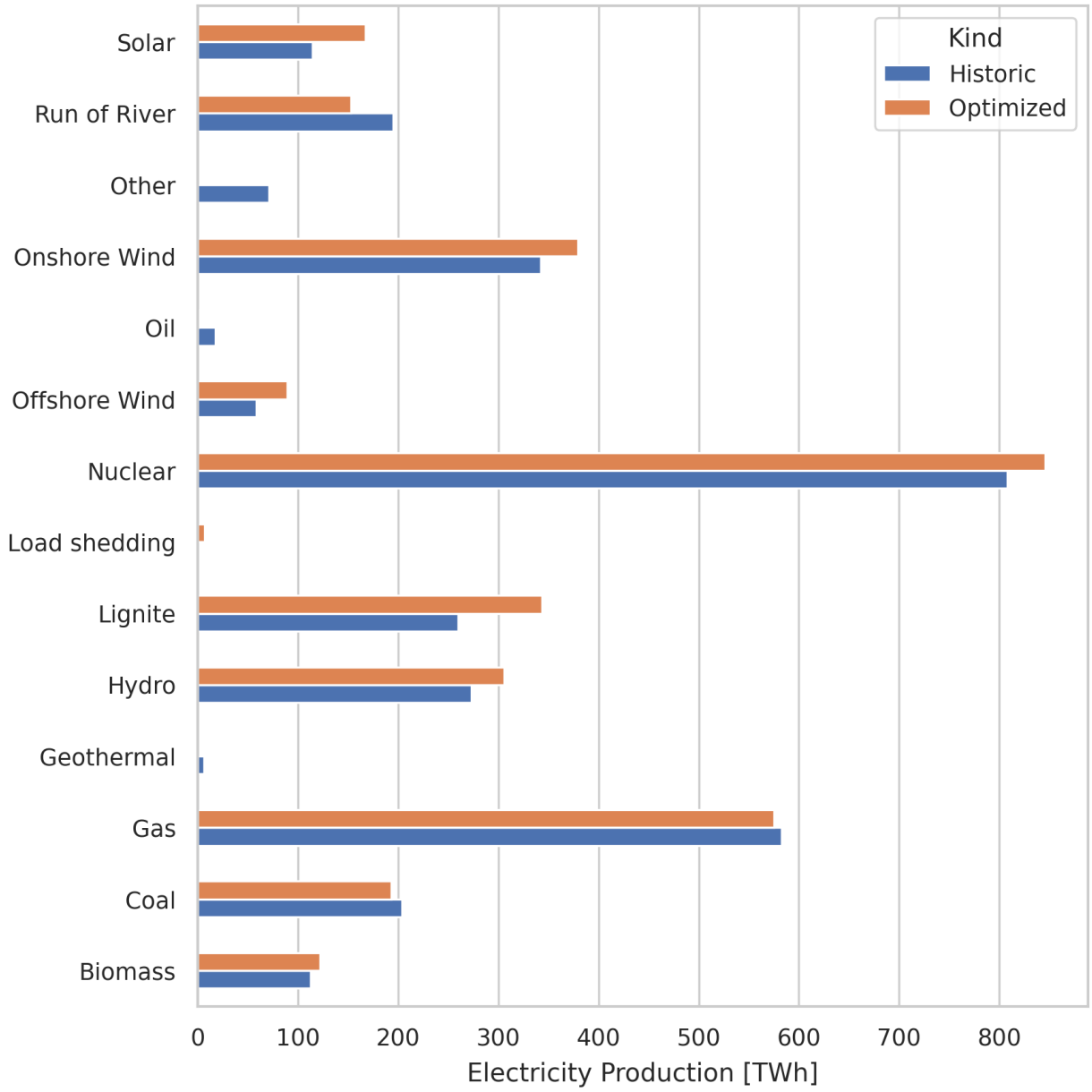
The configuration file *config/config.validation.yaml* contains the following parameters:

The setup uses monthly varying fuel prices for gas, lignite, coal and oil as well as CO2 prices, which are created by the script `build_monthly_prices`. Upon completion of the validation process, the resulting network and generated figures will be stored in the `results/validation` directory for further analysis.

7.3.1 Results

By the time of writing the comparison with the historical data shows partially accurate, partially improvable results. The following figures show the comparison of the dispatch of the different carriers.


Optimized
Historic
Δ (Optimized - Historic)
Production [GW]
Jan 2019
Feb
Mär
Apr
Mai
Jun
Jul
Aug
Sep
Okt
Nov
Dez
Jan 2020
Offshore Wind
Onshore Wind
Solar
Coal
Gas
Oil
Hydro
Lignite
Other
Run of River
Biomass
Nuclear
Geothermal



Issues and possible improvements

Overestimated dispatch of wind and solar: Renewable potentials of wind and solar are slightly overestimated in the model. This leads to a higher dispatch of these carriers than in the historical data. In particular, the solar dispatch during winter is overestimated.

Coal - Lignite fuel switch: The model has a fuel switch from coal to lignite. This might result from non-captured subsidies for lignite and coal in the model. In order to fix the fuel switch from coal to lignite, a manual cost correction was added to the script `build_monthly_prices`.

Planned outages of nuclear power plants: Planned outages of nuclear power plants are not captured in the model. This leads to a underestimated dispatch of nuclear power plants in winter and a overestimated dispatch in summer. This point is hard to fix, since the planned outages are not published in the ENTSO-E Transparency Platform.

False classification of run-of-river power plants: Some run-of-river power plants are classified as hydro power plants in the model. This leads to a general overestimation of the hydro power dispatch. In particular, Swedish hydro power plants are overestimated.

Load shedding: Due to constraint NTC's (crossborder capacities), the model has to shed load in some regions. This leads to a high market prices in the regions which drive the average market price up. Further fine-tuning of the NTC's is needed to avoid load shedding.

7.4 Limitations

While the benefit of an openly available, functional and partially validated model of the European energy system is high, many approximations have been made due to missing data. The limitations of the dataset are listed below, both as a warning to the user and as an encouragement to assist in improving the approximations.

Warning: This list of limitations is incomplete and will be added to over time.

See also:

See also the GitHub repository issues.

- **Electricity transmission network topology:** The grid data is based on a map of the ENTSO-E area that is known to contain small distortions to improve readability. Since the exact impedances of the lines are unknown, approximations based on line lengths and standard line parameters were made that ignore specific conductoring choices for particular lines. There is no openly available data on busbar configurations, switch locations, transformers or reactive power compensation assets.
- **Assignment of electricity demand to transmission nodes:** Using Voronoi cells to aggregate load and generator data to transmission network substations ignores the topology of the underlying distribution network, meaning that assets may be connected to the wrong substation.
- **Incomplete information on existing assets:** Approximations have been made for missing data, including: existing distribution grid capacities and costs, existing space and water heating supply, existing industry facilities, existing transport vehicle fleets.
- **Exogenous pathways for transformation of transport and industry:** To avoid penny-switching the transformation of transport and industry away from fossil fuels is determined exogenously.
- **Industry materials production constant and inelastic:** For industry, the production of different materials per country is assumed to remain constant and no industry demand elasticity is included in the modelled.
- **Energy demand distribution within countries:** Assumptions have been made about the distribution of demand in each country proportional to population and GDP that may not reflect local circumstances. Openly available data on load time series may not correspond to the true vertical load and is not spatially disaggregated; assuming, as we have done, that the load time series shape is the same at each node within each country ignores local differences.
- **Currently installed renewable capacities:** Information on existing wind, solar and small hydro, geothermal, marine and biomass power plants are excluded from the dataset because of a lack of data availability in many countries. Approximate distributions of wind and solar plants in each country can be generated that are proportional to the capacity factor at each location.
- **Hydro-electric power plants:** The database of hydro-electric power plants does not include plant-specific energy storage information, so that blanket values based on country storage totals have been used. Inflow time series are based on country-wide approximations, ignoring local topography and basin drainage; in principle a full hydrological model should be used.

- **International interactions:** Border connections and power flows to Russia, Belarus, Ukraine, Turkey and Morocco have not been taken into account; islands which are not connected to the main European system, such as Malta, Crete and Cyprus, are also excluded from the model.
- **Demand sufficiency:** Further measures of demand reduction may be possible beyond the assumptions made here.

7.5 Contributing

We welcome anyone interested in contributing to this project, be it with new ideas, suggestions, by filing bug reports or contributing code to our GitHub repository.

- If you already have some code changes, you can submit them directly as a pull request.
- If you are wondering where we would greatly appreciate your efforts, check out the `help wanted` tag in the issues list and initiate a discussion there.
- If you start working on a feature in the code, let us know by opening an issue or a draft pull request. This helps all of us to keep an overview on what is being done and helps to avoid a situation where we are doing the same work twice in parallel.

For linting, formatting and checking your code contributions against our guidelines (e.g. we use Black as code style use pre-commit:

1. Installation `mamba install -c conda-forge pre-commit` or `pip install pre-commit`
2. **Usage:**
    - To automatically activate `pre-commit` on every `git commit`: Run `pre-commit install`
    - To manually run it: `pre-commit run --all`

Note: Note that installing `pre-commit` locally is not strictly necessary. If you create a Pull Request the `pre-commit CI` will be triggered automatically and take care of the checks.

For all code contributions we follow the four eyes principle (two person principle), i.e. all suggested code including our own are reviewed by a second person before they are incorporated into our repository.

If you are unfamiliar with pull requests, the GitHub help pages have a nice guide.

To ask and answer general usage questions, join the PyPSA mailing list.

7.6 Support

- In case of code-related **questions**, please post on stack overflow.
- For non-programming related and more general questions please refer to the mailing list.
- To **discuss** with other PyPSA users, organise projects, share news, and get in touch with the community you can use the discord server.
- For **bugs and feature requests**, please use the issue tracker.
- We strongly welcome anyone interested in providing **contributions** to this project. If you have any ideas, suggestions or encounter problems, feel invited to file issues or make pull requests on Github. For further information on how to contribute, please refer to *Contributing*.

7.7 Publications

BIBLIOGRAPHY

[1] Martha Maria Frysztacki, Jonas Hörsch, Veit Hagenmeyer, and Tom Brown. The strong effect of network resolution on electricity system models with high shares of wind and solar. *Applied Energy*, 291:116726, 2021. doi:10.1016/j.apenergy.2021.116726.

[2] Jonas Hörsch, Fabian Hofmann, David Schlachtberger, and Tom Brown. Pypsa-eur: an open optimisation model of the European transmission system. *Energy Strategy Reviews*, 22:207–215, 2018. arXiv:1806.01613, doi:10.1016/j.esr.2018.08.012.

[3] Fabian Neumann, Elisabeth Zeyen, Marta Victoria, and Tom Brown. The potential role of a hydrogen network in Europe. 2022. arXiv:2207.05816.

[4] T. Brown, D. Schlachtberger, A. Kies, S. Schramm, and M. Greiner. Synergies of sector coupling and transmission reinforcement in a cost-optimised, highly renewable European energy system. *Energy*, 160:720–739, 2018. doi:10.1016/j.energy.2018.06.222.

[5] Marta Victoria, Elisabeth Zeyen, and Tom Brown. Speed of technological transformations required in Europe to achieve different climate goals. *Joule*, 6(5):1066–1086, 2022. arXiv:2109.09563, doi:10.1016/j.joule.2022.04.016.

[6] Marta Victoria, Kun Zhu, Tom Brown, Gorm B. Andresen, and Martin Greiner. Early decarbonisation of the European energy system pays off. *Nature Communications*, 11(1):6223, 2020. doi:10.1038/s41467-020-20015-4.

[7] David P. Schlachtberger, Tom Brown, Mirko Schäfer, Stefan Schramm, and Martin Greiner. Cost optimal scenarios of a future highly renewable European electricity system: Exploring the influence of weather data, cost parameters and policy constraints. *Energy*, 163:100–114, 2018. arXiv:http://arxiv.org/abs/1803.09711, doi:10/gfk5cj.

[8] Elisabeth Zeyen, Veit Hagenmeyer, and Tom Brown. Mitigating heat demand peaks in buildings in a highly renewable European energy system. *Energy*, 231:120784, 2021. URL: http://arxiv.org/abs/2012.01831, doi:10.1016/j.energy.2021.120784.

[9] Elisabeth Zeyen, Marta Victoria, and Tom Brown. Endogenous learning for green hydrogen in a sector-coupled energy model for Europe. 2022. URL: http://arxiv.org/abs/2205.11901.

[10] M. Millinger, L. Reichenberg, F. Hedenus, G. Berndes, E. Zeyen, and T. Brown. Are biofuel mandates cost-effective? - an analysis of transport fuels and biomass usage to achieve emissions targets in the european energy system. *Applied Energy*, 326:120016, 2022. doi:https://doi.org/10.1016/j.apenergy.2022.120016.

[11] Martha Maria Frysztacki, Veit Hagenmeyer, and Tom Brown. Inverse methods: How feasible are spatially low-resolved capacity expansion modeling results when dis-aggregated at high resolution? 2022. URL: http://arxiv.org/abs/2209.02364.

[12] Martha Frysztacki and Tom Brown. Modeling Curtailment in Germany: How Spatial Resolution Impacts Line Congestion. In *2020 17th International Conference on the European Energy Market (EEM)*, 1–7. IEEE, 2020. doi:10.1109/EEM49802.2020.9221886.

[13] Martha Maria Frysztacki, Gereon Recht, and Tom Brown. A comparison of clustering methods for the spatial reduction of renewable electricity optimisation models of Europe. *Energy Informatics*, 5(1):4, 2022. URL: https://energyinformatics.springeropen.com/articles/10.1186/s42162-022-00187-7, doi:10.1186/s42162-022-00187-7.

[14] Fabian Neumann and Tom Brown. The near-optimal feasible space of a renewable power system model. *Electric Power Systems Research*, 190:106690, 2021. doi:10.1016/j.epsr.2020.106690.

[15] Fabian Neumann, Veit Hagenmeyer, and Tom Brown. Assessments of linear power flow and transmission loss approximations in coordinated capacity expansion problems. *Applied Energy*, 2022. doi:10.1016/j.apenergy.2022.118859.

[16] Fabian Neumann. Costs of regional equity and autarky in a renewable European power system. *Energy Strategy Reviews*, 2021. doi:10.1016/j.esr.2021.100652.

[17] Philipp K. Rose and Fabian Neumann. Hydrogen refueling station networks for heavy-duty vehicles in future power systems. *Transportation Research Part D: Transport and Environment*, 83:102358, 2020. doi:10.1016/j.trd.2020.102358.

[18] Fabian Neumann and Tom Brown. Heuristics for Transmission Expansion Planning in Low-Carbon Energy System Models. In *2019 16th International Conference on the European Energy Market (EEM)*, 1–8. IEEE, 2019. doi:10.1109/EEM.2019.8916411.

[19] Fabian Neumann and Tom Brown. Broad Ranges of Investment Configurations for Renewable Power Systems, Robust to Cost Uncertainty and Near-Optimality. 2021. URL: http://arxiv.org/abs/2111.14443.

[20] Amin Shokri Gazafroudi, Elisabeth Zeyen, Martha Frysztacki, Fabian Neumann, and Tom Brown. Long-Term Benefits for Renewables Integration of Network Boosters for Corrective Grid Security. 2021. URL: http://arxiv.org/abs/2112.06667.

[21] Amin Shokri Gazafroudi, Fabian Neumann, and Tom Brown. Topology-based approximations for N - 1 contingency constraints in power transmission networks. *International Journal of Electrical Power & Energy Systems*, 137:107702, 2022. doi:10.1016/j.ijepes.2021.107702.

[22] Jonas Horsch and Tom Brown. The role of spatial scale in joint optimisations of generation and transmission for European highly renewable scenarios. In *2017 14th International Conference on the European Energy Market (EEM)*, 1–7. IEEE, 2017. doi:10.1109/EEM.2017.7982024.

[23] D.P. Schlachtberger, T. Brown, S. Schramm, and M. Greiner. The benefits of cooperation in a highly renewable European electricity network. *Energy*, 134:469–481, 2017. doi:10.1016/j.energy.2017.06.004.

[24] Philipp Glaum and Fabian Hofmann. Enhancing the German Transmission Grid Through Dynamic Line Rating. 2022. URL: http://arxiv.org/abs/2208.04716.

[25] Maximilian Parzen, Hazem Abdel-Khalek, Ekaterina Fedorova, Matin Mahmood, Martha Maria Frysztacki, Johannes Hampp, Lukas Franken, Leon Schumm, Fabian Neumann, Davide Poli, Aristides Kiprakis, and Davide Fioriti. PyPSA-Earth. A New Global Open Energy System Optimization Model Demonstrated in Africa. 2022. URL: http://arxiv.org/abs/2209.04663.

PYTHON MODULE INDEX

INDEX

C

D

M

P

R

S