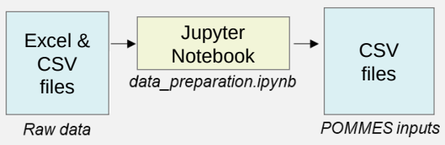
Modelling aspects of *pommesdata* and *pommesinvest*

General Overview

pommesdata



Aim

- Prepare data for …
  - *pommesdispatch* and
  - *pommesinvest*
- *Provide transparency*
  - *Start with (mostly) raw data*
  - *end with POMMES inputs*

Scope

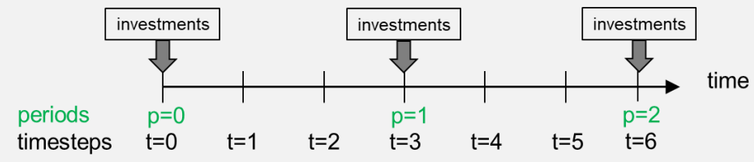
- Time frame: 2020-2045
- Resolution: Hourly (infeed) or annually (costs)

- Bidding zone / country-level for DE + neighbours + IT (DE, AT, CH, FR, BE, NL, CZ, PL, DK, NO, SE, IT)
- no georeference in prepared data; but in some raw data

- Existing
  - RES: wind onshore / offshore, solar PV, biomass, run of river, others
  - conventionals: nuclear, lignite, hard coal, biomass, oil, other / mixed
  - Storages: pumped hydro, reservoir
- Expansion
  - Backup: natural gas, hydrogen, biomass, oil
  - Storages: pumped hydro, (lithium-ion) batteries, electrolysers
  - Demand response: households, commercial and industrial

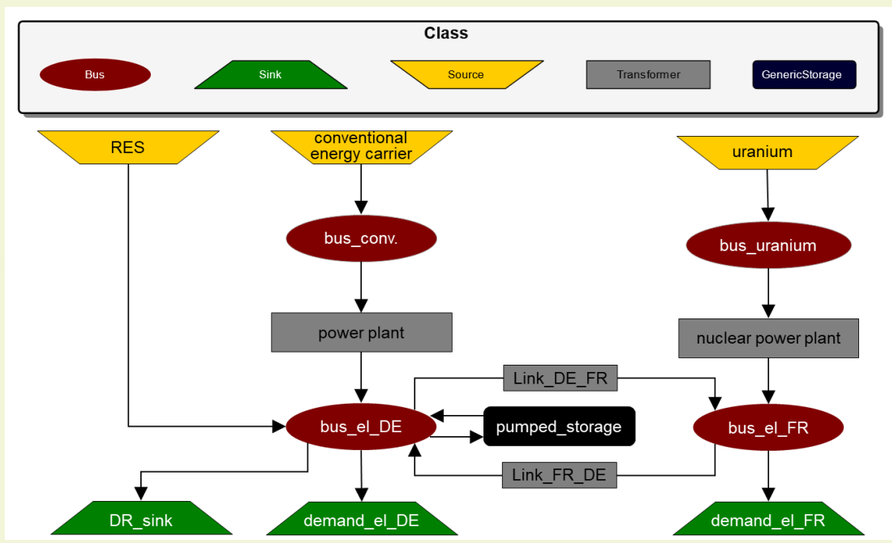
https://github.com/pommes-public/pommesdata

pommesinvest



Sketch

- multi-period investments
- time frame 2020-2045 in hourly resolution
- DE + electrical neighbours + IT

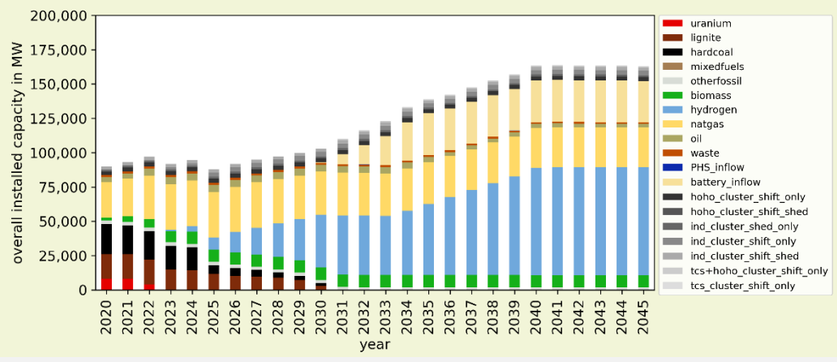


DE: Germany
FR: Fance
IT: Italy

el: electrical
DR: demand response

total: around 600 nodes

Results

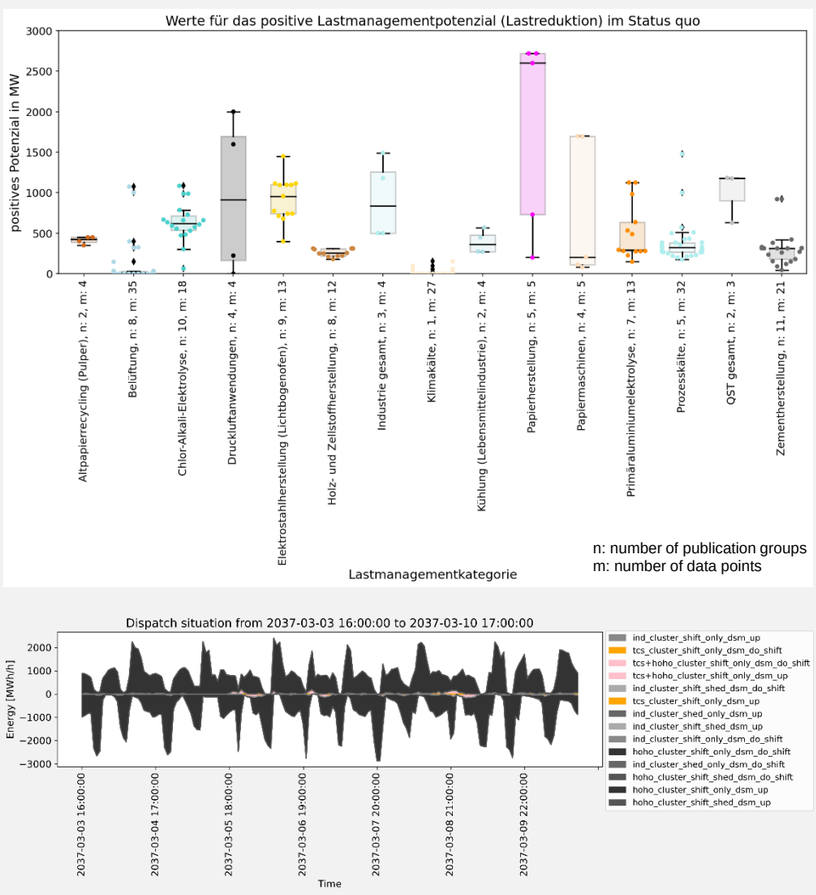


https://github.com/pommes-public/pommesinvest

Selected modelling aspects

Demand Response

- Applying enhanced *SinkDSM* component
- Depicting investments in demand response and distincting between load shifting and load shedding
- Applying adapated modelling approach from Gils (2015)
- Parameterization based on primary sources from Kochems (2020)



Multi-period investments

Periods

Extract years from index

Cut leap years to ensure 8,760 hours per year

```python
def determine_periods(self, datetimeindex):
    """Explicitly define and return periods of the energy system..."""
    years = sorted(list(set(getattr(datetimeindex, "year"))))
    periods = []
    filter_series = datetimeindex.to_series()
    for number, year in enumerate(years):
        start = filter_series.loc[filter_series.index.year == year].min()
        if not is_leap_year(year):
            end = filter_series.loc[filter_series.index.year == year].max()
        else:
            # Exclude last day / resp. leap day
            end = filter_series.loc[
                (filter_series.index.year == year)
                & (filter_series.index.month == 12)
                & (filter_series.index.day != 31)
            ].max()
        periods.append(pd.date_range(start, end, freq=self.freq))

    return periods
```

Lifetime tracking (in oemof.solph)

$$P_{total}(p) = P_{invest}(p) + P_{total}(p-1) - P_{old}(p) \quad \forall p > 0$$

$$P_{total}(p) = P_{invest}(p) + P_{existing} \quad \forall p = 0$$

$$P_{old,end}(p) = P_{invest}(p-n) \quad \forall p \geq n$$

$$P_{old,end}(p) = 0 \quad else$$

$$P_{old,exo}(p) = P_{existing} \quad \forall p = n - age$$

$$P_{old,exo}(p) = 0 \quad else$$

$$P_{old}(p) = P_{old,end}(p) + P_{old,exo}(p)$$

P: capacity
p: period
n: lifetime
end: endogenous
exo: exogenous
$invest$: investments
old: decommissionings

- Calculate commissioning year by mapping periods to years
- Use boolean flag to detect if existing capacity was decommissioned

Discounting and annuities (in oemof.solph)

$$P_{invest}(p) \cdot annuity(c_{invest}(p), n, i) \cdot n \cdot DF(p) \quad \forall p \in PERIODS$$

$$annuity(c_{invest}(p), n, i) = \frac{(1+i)^n \cdot i}{(1+i)^n - 1} \cdot c_{invest}(p)$$

$$DF(p) = (1+d)^{-p}$$

- Use a technology-specific interest rate i (WACC)
- Use a discounting factor d of 2 % p.a. for nominal costs

Complexity & Maintenance

pommesdata

- Maintenance is tough!
- Reasons
  - *Structure*: one large jupyter notebook (JSON format)
    → hard to read diffs
    → hard to identify code dependencies
  - *Historically grown*: a lot of features / data sets that have accumulated over time
  - *Data sets*: Data sets itself can become outdated
    → good thing there are no API requests that can break
    → bad thing that this means manual reworks

pommesinvest

- Complexity is high!
- Reasons
  - High number of units and time steps
  - Processing not optimized for speed / memory usage

- Specifications of model runs

with DR	DR scenario	shifting times limited	run-time	memory usage (in GB)	rows after presolve	columns after presolve	nonzeros after presolve
no	-	-	09:44	342.7	20,368,023	70,158,635	125,253,822
yes	5	no	12:24	400.58	28,174,251	75,947,844	185,928,478
yes	50	no	15:55	454.58	30,036,656	86,238,461	271,180,925
yes	95	yes	18:25	427.43	30,516,218	83,562,641	243,343,059

Machine:
- Dell PowerEdge R630; Intel Xeon E5-2600 v4 processor
- 36 cores, 72 threads; 512 GB RAM

image sources:
https://reiner-lemoine-institut.de/wp-content/uploads/2019/05/oemof_header.jpg, accessed 19.05.2023
oemof documentation, oemof-solph - usage, available online at https://oemof-solph.readthedocs.io/en/latest/usage.html, accessed 19.05.2023.
https://raw.githubusercontent.com/oemof/oemof-solph/492e3f5a0dda7065be30d33a37b0625027847518/docs/_logo/logo_oemof_solph_FULL.svg, accessed 19.05.2023.

Works cited:
Gils, Hans Christian (2015): Balancing of Intermittent Renewable Power Generation by Demand Response and Thermal Energy Storage. Dissertation. Universität Stuttgart.
Kochems, Johannes (2020): Lastflexibilisierungspotenziale in Deutschland – Bestandsaufnahme und Entwicklungsprojektionen, Langfassung: In: Tagungsband 16. Symposium Energieinnovation, 12.-14.02.2020, Graz.

Contact: Johannes Kochems; johannes.kochems@dlr.de; Github: @jokochems