ZNES
ZENTRUM FÜR NACHHALTIGE ENERGIESYSTEME
HOCHSCHULE | EUROPA-UNIVERSITÄT FLENSBURG

RLI
REINER LEMOINE INSTITUT

OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

oemof
['ø:mɔf]

Using the oemof cosmos - de21

oemof – a community project to make energy modelling transparent and shareable

Uwe Krien

10. Mai 2017, oemof user meeting 2017

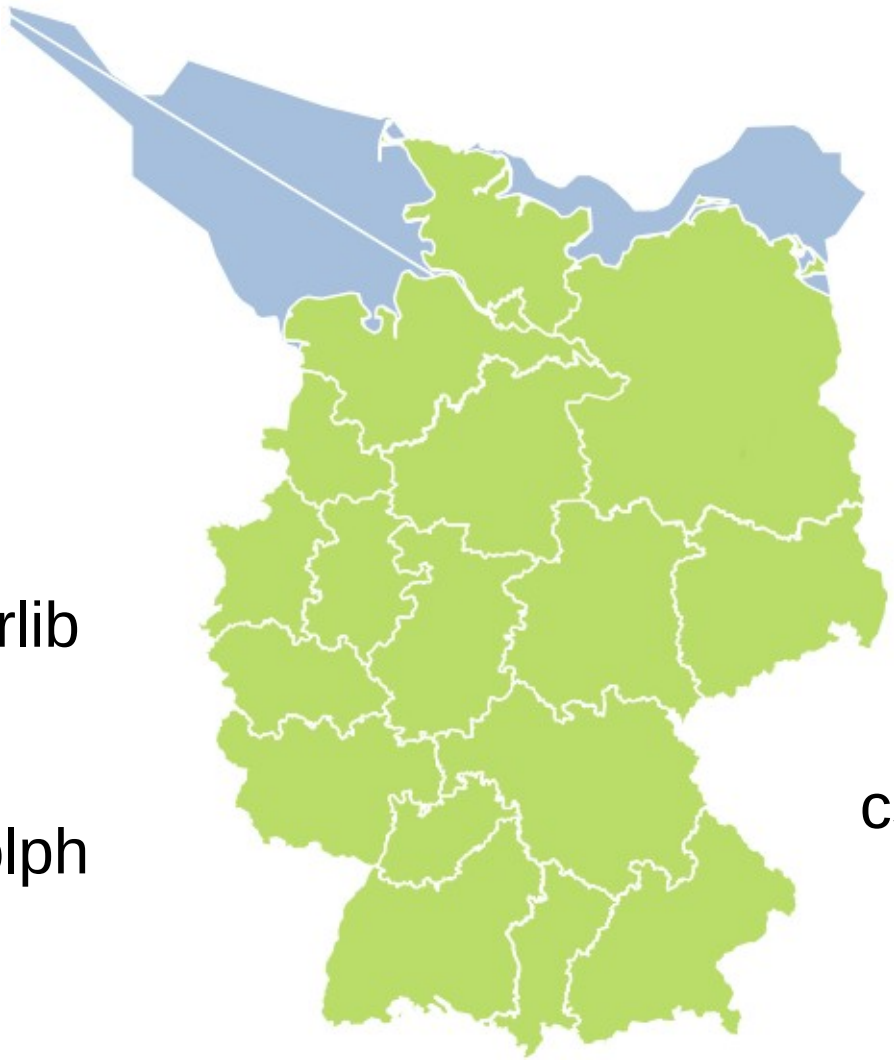
de21– python as connector

pvlib

demandlib

windpowerlib

oemof.solph



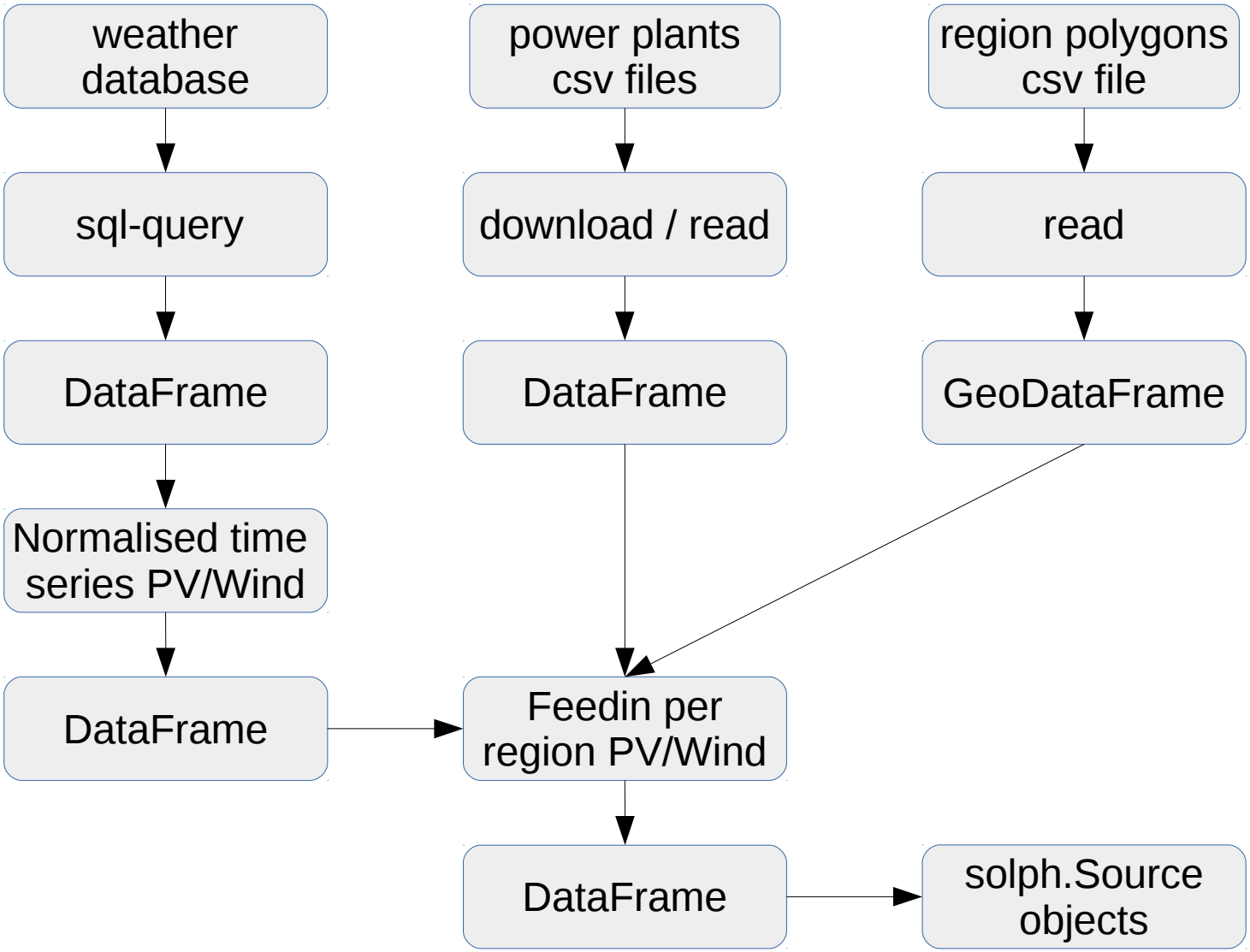
weather data

open power data

open energy db

csv-scenario

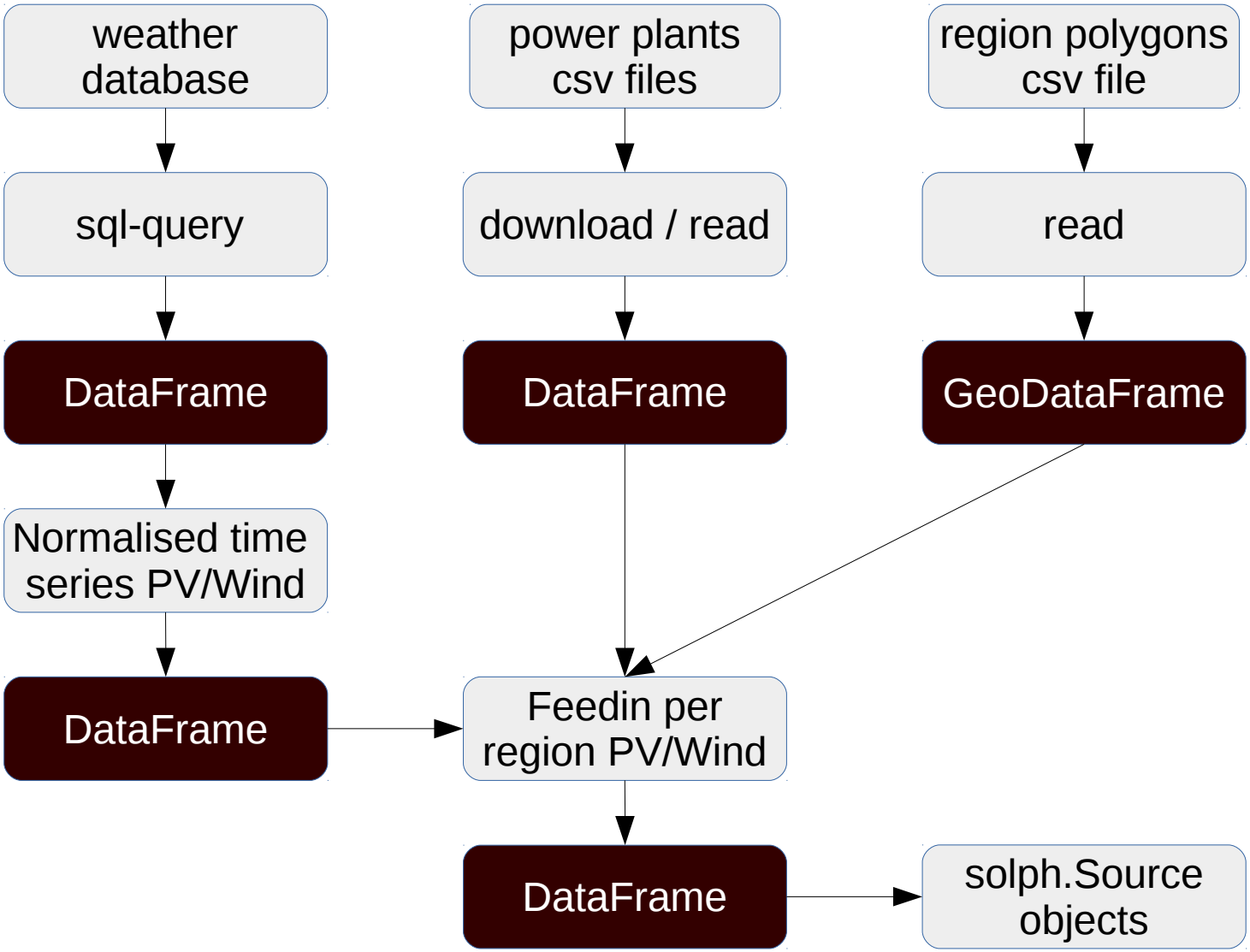
de21 – solph.Source object from open data



de21 – using python

- Database
  - SQLAlchemy
  - Psycopg
- Downloads
  - Requests
- GIS operations
  - Shapely
  - Postgis (see database)
  - GeoPandas
  - PyQGIS
- Input/output
  - pandas

de21 – solph.Source object from open data

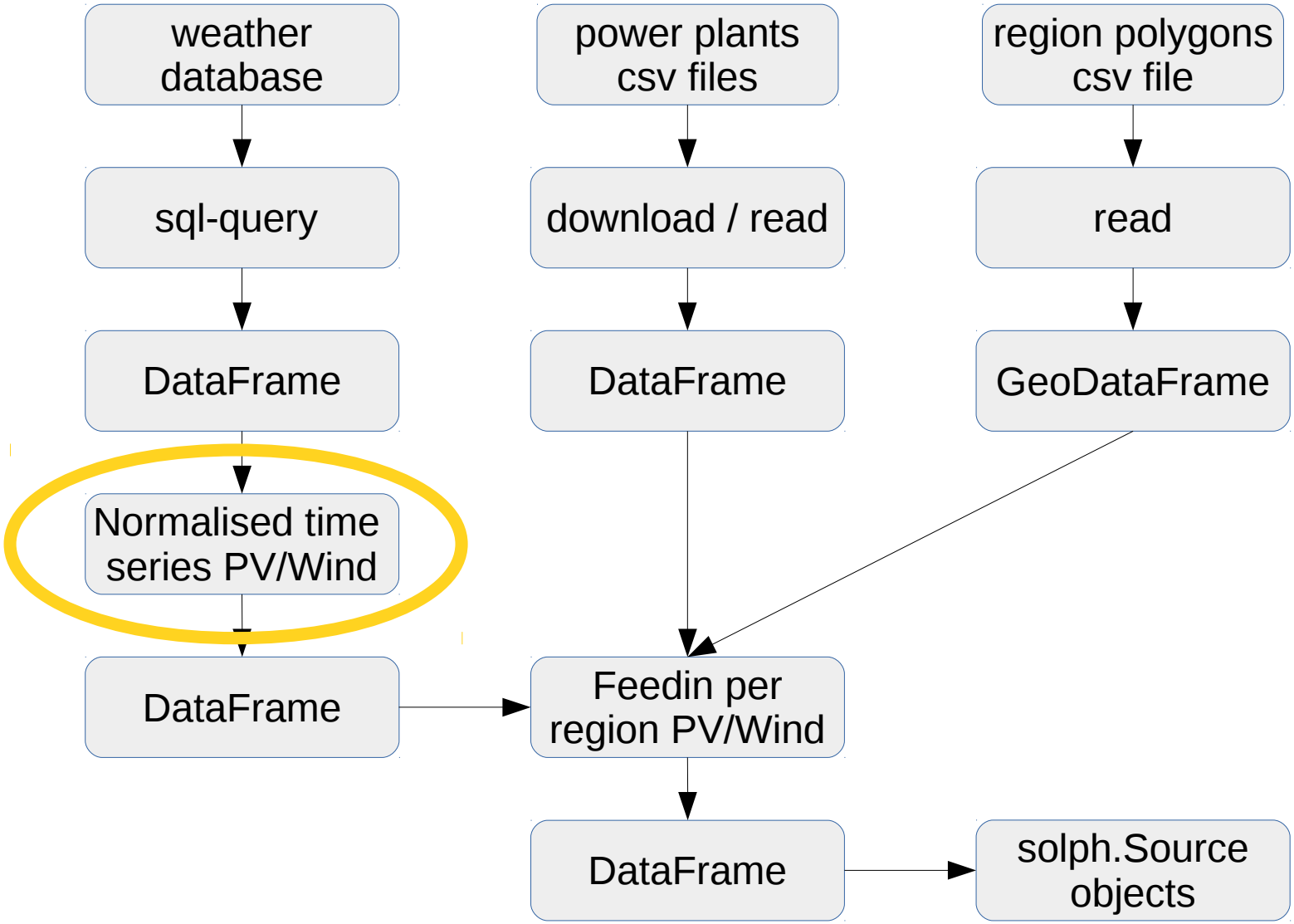


oemof ['ø:məf]

de21 – pandas.DataFrame

- CSV: `read_csv, to_csv`
- JSON: `read_json, to_json`
- HTML: `read_html, to_html`
- HDF5 : `read_hdf, to_hdf`
- Clipboard: `read_clipboard, to_clipboard`
- MS Excel: `read_excel, to_excel`
- Python Pickle: `read_pickle, to_pickle`
- SQL: `read_sql, to_sql`
- Google Big Query: `read_gbq, to_gbq`

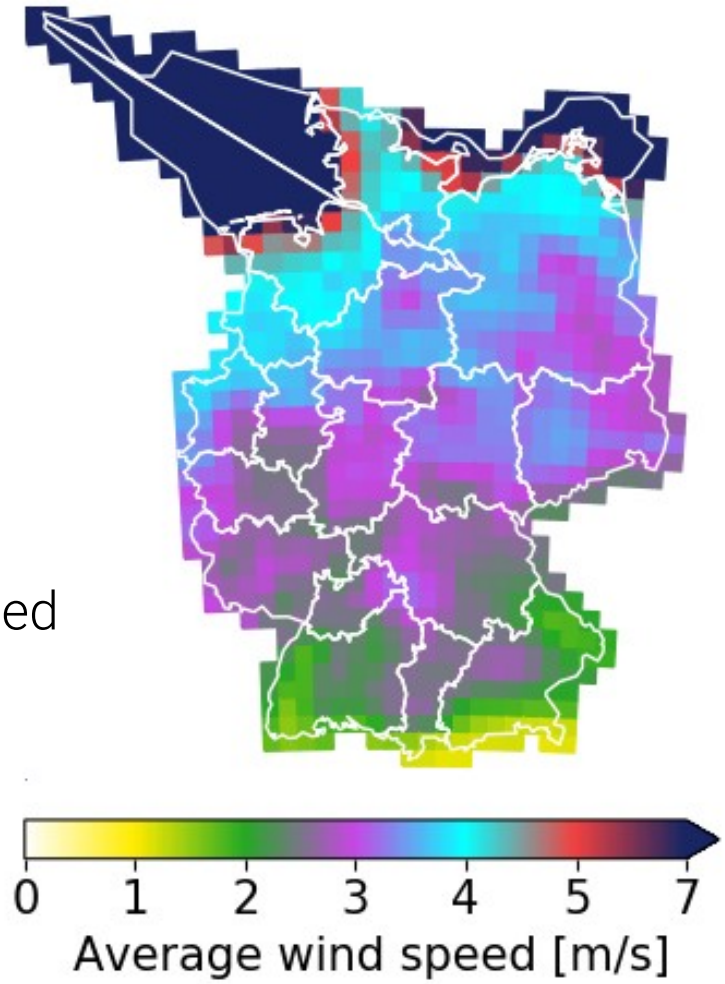
de21 – solph.Source object from open data



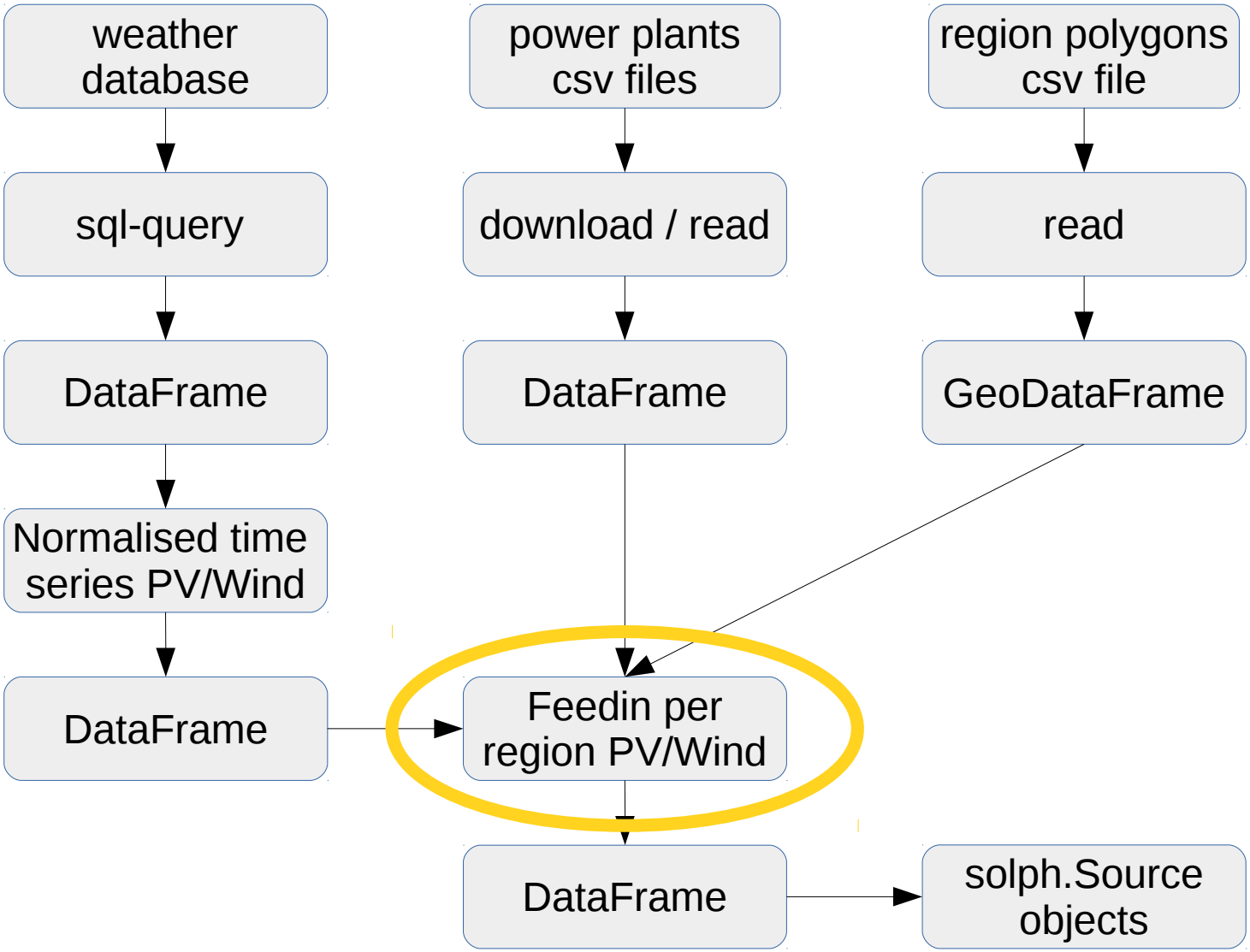
oemof ['ø:məf]

de21 – feedinlin (windpowerlib, pvlib)

- pvlib
  - surface azimuth
  - surface tilt
  - module type
- windpowerlib
  - selection by average wind speed
  - type of turbine
  - hub height
  - diameter of rotor

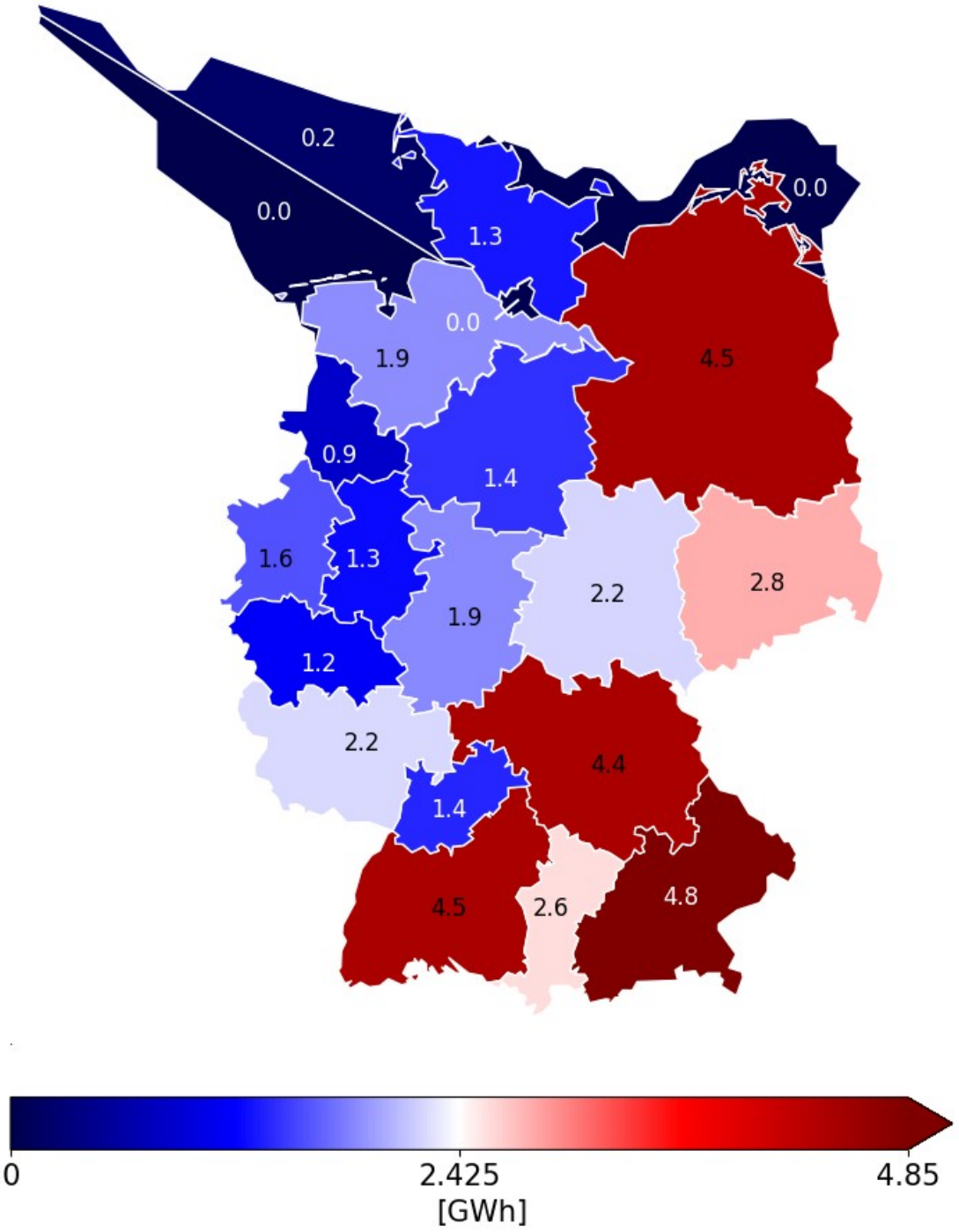


de21 – solph.Source object from open data



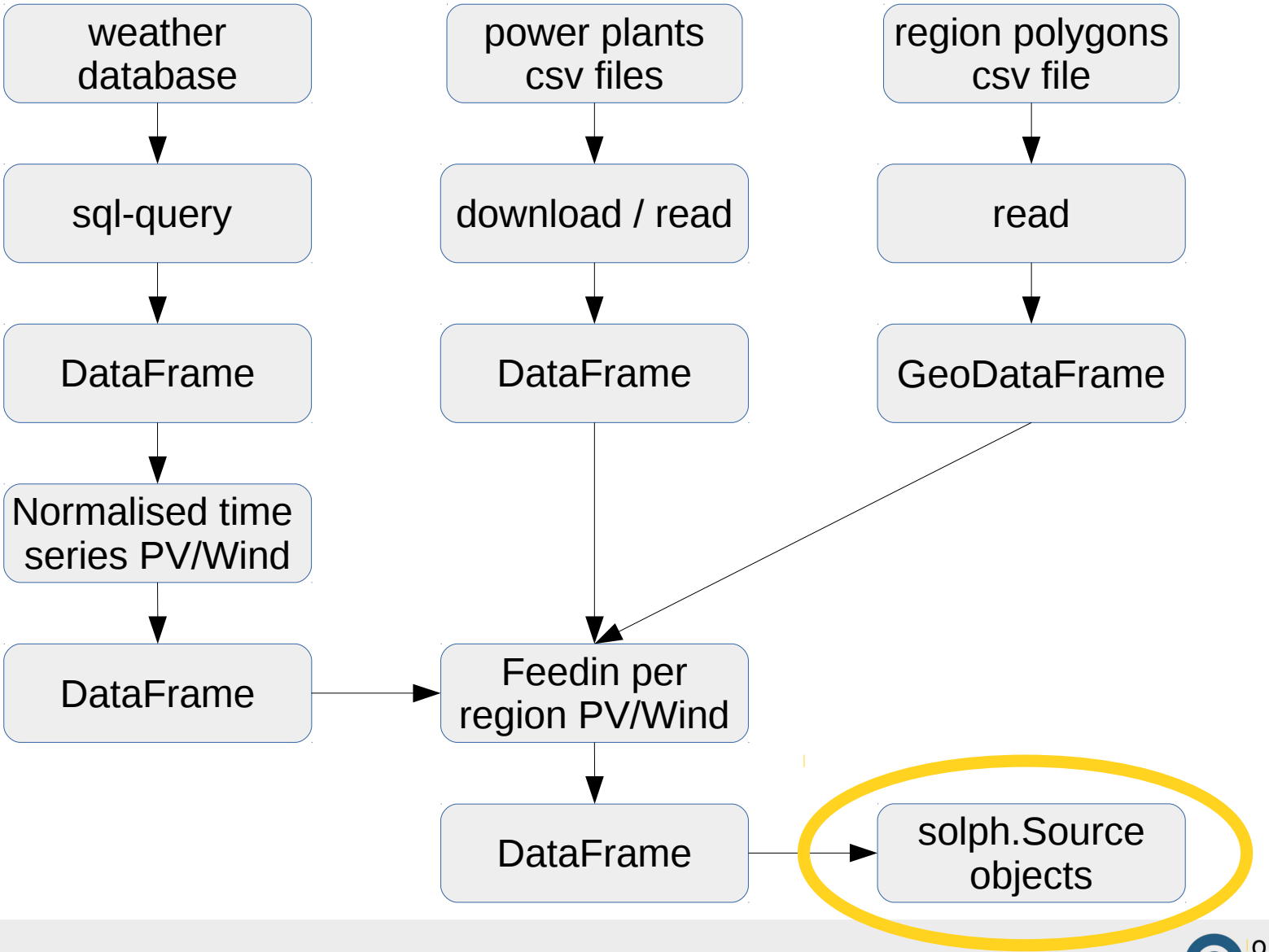
oemof ['ø:məf]

de21



oemof ['ø:məf]

de21 – solph.Source object from open data



oemof ['ø:məf]

de21 – solph.Source object from open data

```
feedin = pandas.read_csv('feedin_de21.csv')
capacity = pandas.read_csv('re_capacities_de21.csv')

regions = ['de01', 'de02', ...]
pp_types = ['pv', 'wind']

for region in regions:
    for pp_type in pp_types:

        name = region + '_' + pp_type
        time_series = feedin[(pp_type, reg)]
        capacity = capacities[(pp_type, reg)]

        solph.Source(label=name, outputs={bus_elec: solph.Flow(
            actual_value=time_series, nominal_value=capacity, fixed=True)})
```

de21 – solph.Source object from open data

```
feedin = pandas.read_csv('feedin_de21.csv')
capacity = pandas.read_csv('re_capacities_de21.csv')

regions = ['de01', 'de02', ...]
pp_types = ['pv', 'wind']

for region in regions:
    for pp_type in pp_types:

        name = region + '_' + pp_type
        time_series = feedin[(pp_type, reg)]
        capacity = capacities[(pp_type, reg)]

        solph.Source(label=name, outputs={bus_elec: solph.Flow(
            actual_value=time_series, nominal_value=capacity, fixed=True)})
```

de21 – `feedin_21.csv` (libreoffice)

	A	B	C	D	E
1	pp_type	wind		pv	
2	region	DE01	DE02	DE01	DE02
3	01.01.14 00:00	0,161431589	0,2424160846	0	0
4	01.01.14 01:00	0,0964284853	0,182236562	0	0
5	01.01.14 02:00	0,10718897	0,1817105899	0	0
6	01.01.14 03:00	0,1150653444	0,273519003	0	0
7	01.01.14 04:00	0,116135532	0,2463081924	0	0
8	01.01.14 05:00	0,1149060915	0,2267081491	0	0
9	01.01.14 06:00	0,1134285888	0,2312934719	0	0
10	01.01.14 07:00	0,1108836687	0,1750679727	0	0
11	01.01.14 08:00	0,1078779072	0,1455857641	0	0
12	01.01.14 09:00	0,1025639489	0,1405771328	6,33701E-005	0
13	01.01.14 10:00	0,0950407562	0,1220165933	0,0422880793	0
14	01.01.14 11:00	0,0868960669	0,111012614	0,0987953417	0,0437733545
15	01.01.14 12:00	0,0739333742	0,1298543867	0,1176598025	0,3623594147
16	01.01.14 13:00	0,0658132647	0,1446836859	0,0891018535	0,494561146
17	01.01.14 14:00	0,0588620834	0,138828364	0,0691240805	0,4745171844
18	01.01.14 15:00	0,0518779613	0,1525743349	0,0562799208	0,2542793827
19	01.01.14 16:00	0,0516459129	0,1868380521	0	0
20	01.01.14 17:00	0,0539846809	0,1712215399	0	0
21	01.01.14 18:00	0,0591474281	0,1513662636	0	0
22	01.01.14 19:00	0,0701826957	0,1714315178	0	0
23	01.01.14 20:00	0,0898451981	0,2421234482	0	0
24	01.01.14 21:00	0,1088405555	0,2847195641	0	0
25	01.01.14 22:00	0,1233745647	0,2982942605	0	0
26	01.01.14 23:00	0,1470443566	0,3530740347	0	0
27	02.01.14 00:00	0,1771054668	0,4696898035	0	0
28	02.01.14 01:00	0,2030055484	0,4447462026	0	0
29	02.01.14 02:00	0,220925328	0,4866946671	0	0
30	02.01.14 03:00	0,2329795597	0,6251752399	0	0

de21 – solph.Source object from open data

```
feedin = pandas.read_csv('feedin_de21.csv')
capacity = pandas.read_csv('re_capacities_de21.csv')

regions = ['de01', 'de02', ...]
pp_types = ['pv', 'wind']

for region in regions:
    for pp_type in pp_types:

        name = region + '_' + pp_type
        time_series = feedin[(pp_type, reg)]
        capacity = capacities[(pp_type, reg)]

        solph.Source(label=name, outputs={bus_elec: solph.Flow(
            actual_value=time_series, nominal_value=capacity, fixed=True)})
```

de21 – solph.Source object from open data

```
feedin = pandas.read_csv('feedin_de21.csv')
capacity = pandas.read_csv('re_capacities_de21.csv')

regions = ['de01', 'de02', ...]
pp_types = ['pv', 'wind']

for region in regions:
    for pp_type in pp_types:

        name = region + '_' + pp_type
        time_series = feedin[(pp_type, reg)]
        capacity = capacities[(pp_type, reg)]

        solph.Source(label=name, outputs={bus_elec: solph.Flow(
            actual_value=time_series, nominal_value=capacity, fixed=True)})
```

de21 – solph.Source object from open data

```
feedin = pandas.read_csv('feedin_de21.csv')
capacity = pandas.read_csv('re_capacities_de21.csv')

regions = ['de01', 'de02', ...]
pp_types = ['pv', 'wind']

for region in regions:
    for pp_type in pp_types:

        name = region + '_' + pp_type
        time_series = feedin[(pp_type, reg)]
        capacity = capacities[(pp_type, reg)]

        solph.Source(label=name, outputs={bus_elec: solph.Flow(
            actual_value=time_series, nominal_value=capacity, fixed=True)})
```

de21 – solph.Source object from open data

```
feedin = pandas.read_csv('feedin_de21.csv')
capacity = pandas.read_csv('re_capacities_de21.csv')

regions = ['de01', 'de02', ...]
pp_types = ['pv', 'wind']

for region in regions:
    for pp_type in pp_types:

        name = region + '_' + pp_type
        time_series = feedin[(pp_type, reg)]
        capacity = capacities[(pp_type, reg)]

        solph.Source(label=name, outputs={bus_elec: solph.Flow(
            actual_value=time_series, nominal_value=capacity, fixed=True)})
```

de21 – solph csv file from open data

```
de21 = solph.Scenario(path='scenarios', name='cool_scenario')
de21.create_tables()

feedin = pandas.read_csv('feedin_de21.csv')
capacity = pandas.read_csv('re_capacities_de21.csv')

regions = ['de01', 'de02', ...]
pp_types = ['pv', 'wind']

for region in regions:
    for pp_type in pp_types:
        name = region + '_' + pp_type
        time_series = feedin[(pp_type, reg)]
        capacity = capacities[(pp_type, reg)]
        target = region + '_bus_el'
        idx = ('Source', name, name, target)
        cols = ['nominal_value', 'actual_value', 'fixed']
        values = [capacity, 'seq', 1]
        de21.add_parameters(idx, cols, values)

        idx = ['Source', name, name, target, 'actual_value']
        de21.add_sequences(idx, time_series)
de21.write_tables()
```

Scenario file (csv)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	class	label	source	target	conversion_factors	nominal_value	min	max	summed_max	actual_value	fixed_costs	variable_costs	fixed
2	### GLOBAL RESOURCES												
3	Source	GL_resource_waste	GL_resource_waste	GL_bus_waste								30	
4	Source	GL_resource_other_fossil_fuels	GL_resource_other_fossil_fuels	GL_bus_other_fossil_fuels								27,24696	
5	Source	GL_resource_oil	GL_resource_oil	GL_bus_oil								86,86674	
6	Source	GL_resource_nuclear	GL_resource_nuclear	GL_bus_nuclear								5,961744	
7	Source	GL_resource_natural_gas	GL_resource_natural_gas	GL_bus_natural_gas								54,05328	
8	Source	GL_resource_hydro	GL_resource_hydro	GL_bus_hydro								0	
9	Source	GL_resource_hard_coal	GL_resource_hard_coal	GL_bus_hard_coal								45,86634	
10	Source	GL_resource_biomass_and_biog	GL_resource_biomass_and_biog	GL_bus_biomass_and_biogas								27,73476	
11	Source	GL_resource_lignite	GL_resource_lignite	GL_bus_lignite								27,5949	
12	### DE01												
13	LinearTransformer	DE01_pp_biomass_and_biogas	DE01_pp_biomass_and_biogas	DE01_bus_el	0,38	268							
14	LinearTransformer	DE01_pp_biomass_and_biogas	GL_bus_biomass_and_biogas	DE01_pp_biomass_and_biogas									
15	LinearTransformer	DE01_pp_hard_coal	DE01_pp_hard_coal	DE01_bus_el	0,4	1291							
16	LinearTransformer	DE01_pp_hard_coal	GL_bus_hard_coal	DE01_pp_hard_coal									
17	LinearTransformer	DE01_pp_lignite	DE01_pp_lignite	DE01_bus_el	0,36	209							
18	LinearTransformer	DE01_pp_lignite	GL_bus_lignite	DE01_pp_lignite									
19	LinearTransformer	DE01_pp_natural_gas	DE01_pp_natural_gas	DE01_bus_el	0,41	2164							
20	LinearTransformer	DE01_pp_natural_gas	GL_bus_natural_gas	DE01_pp_natural_gas									
21	LinearTransformer	DE01_pp_oil	DE01_pp_oil	DE01_bus_el	0,37	660							
22	LinearTransformer	DE01_pp_oil	GL_bus_oil	DE01_pp_oil									
23	LinearTransformer	DE01_pp_other_fossil_fuels	DE01_pp_other_fossil_fuels	DE01_bus_el	0,33	124							
24	LinearTransformer	DE01_pp_other_fossil_fuels	GL_bus_other_fossil_fuels	DE01_pp_other_fossil_fuels									
25	LinearTransformer	DE01_pp_waste	DE01_pp_waste	DE01_bus_el	0,33	187							
26	LinearTransformer	DE01_pp_waste	GL_bus_waste	DE01_pp_waste									
27	Source	DE01_wind	DE01_wind	DE01_bus_el		8433				seq			1
28	Source	DE01_solar	DE01_solar	DE01_bus_el		3513				seq			1
29	Sink	DE01_load	DE01_bus_el	DE01_load		3161				seq			1
30	Source	DE01_shortage	DE01_shortage	DE01_bus_el								1000	
31	Sink	DE01_excess	DE01_bus_el	DE01_excess									
32	### DE02												
33	LinearTransformer	DE02_pp_biomass_and_biogas	DE02_pp_biomass_and_biogas	DE02_bus_el	0,38	20							
34	LinearTransformer	DE02_pp_biomass_and_biogas	GL_bus_biomass_and_biogas	DE02_pp_biomass_and_biogas									

27	Source	DE01_wind	DE01_wind	DE01_bus_el
28	Source	DE01_solar	DE01_solar	DE01_bus_el

oemof ['ø:məf]

Using the oemof cosmos – de21

- pandas for i/o conversions
- different python packages to fetch/process data
- windpowerlib / pvlib / feedinlib
- creating solph objects by looping DataFrames

Any questions?