Powerplantmatching Tool

Information on power plants, particularly European ones is scattered over a few different projects and databases that are introducing their own different standards. The open-source package package **powerplantmatching** provides functions to vertically clean databases and convert them into one coherent standard. It provides functions to horizontally merge different databases in order to improve the reliability.

1 Structure

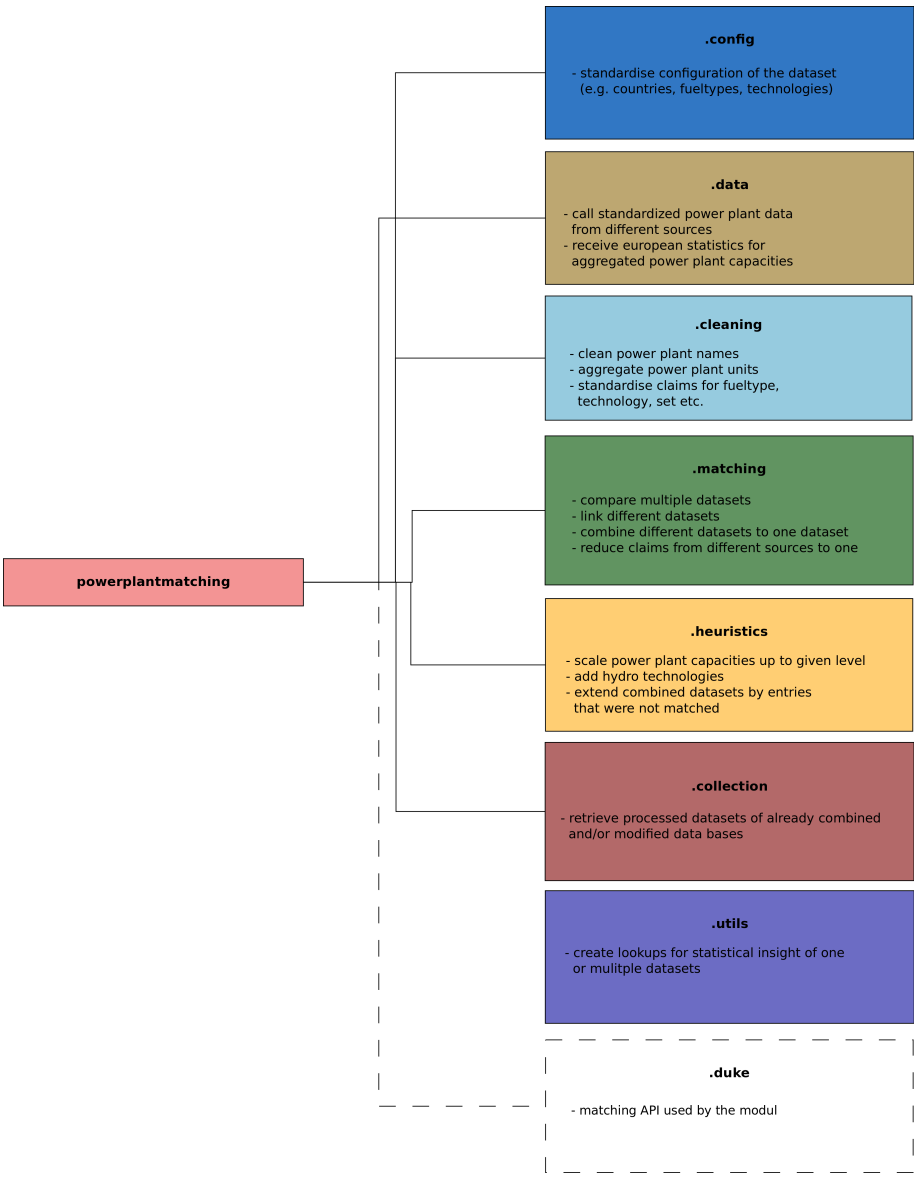

Figure 1: Structure of the powerplant-module. The modules stand for different task areas

The package **powerplantmatching** consists of eight different modules, shown in Figure 1 for different tasks helping to standardize and combine power plant data sets and providing already processed data collections for the research community.

1.1 Configuration - powerplantmatching.config

The module **configuration** sets the standard of the data structure. It defines the exact set of properties (columns) and arguments that should be given per power plant as well as the set of countries that should be covered by the final data set. The following Table 1 shows the configuration set of arguments.

Column	Argument	Python Format
Power plant name	original name of the database	str
Fueltype	{Bioenergy, Geothermal, Hard Coal, Hydro, Lignite, Nuclear, Natural Gas, Oil, Solar, Wind, Other }	str
Technology	{CCGT, OCGT, Steam Turbine, Combustion Engine, Run-Of-River, Pumped Storage, Reservoir }	str
Set	{PP, CHP}	str
Capacity	Brutto or Net Generation Capacity in MW	float
lat	Latitude	float
lon	Longitude	float
Country	{EU-27 + CH + NO (+ UK) minus Cyprus and Malta}	str
YearCommissioned	Commissioning year of the power plants	int
File	Source file of the data entry, if available	str
projectID	Identifier of the power plant in the original source file	str

Table 1: Standardizing data structure for the package.

The set of technological arguments as 'Fueltype', 'Technology' and 'Set' is

mainly based on the standard of the OPSD project [5]. However, we have reduced the number of possible combinations between 'Fueltype'-arguments and 'Technology'-arguments.

The module consists of of five commands, each returning a python list, determining the configuration. The command `europeancountries` returns a list of all considered countries. The command `target_columns` returns a list of all column names stated in Table 1. The commands `target_fueltypes`, `target_technologies` and `target_sets` return the corresponding list of arguments given in the Table.

1.2 Available data - powerplantmatching.data

The data base of the package consists of six different open-source power plant datasets from different projects that are either open or at least available free-off-charge:

- **Open Power System Data** (OPSD) [5], free license power plant data for EU-28 countries. The dataset can be called through the function `OPSD`. It consists of two datasets provided by the project, one German specific dataset, including data from the Bundesnetzagentur [6] and the Umwelt Bundesamt [7], and one European dataset, including the Czech Republic, France, Netherlands, Poland and Switzerland so far. Both datasets originally have different formats and can be obtained in raw version by setting *rawDE=True* or *rawEU=True*

- **Global Energy Observatory**(GEO) [8], free license power plant data for all countries, can be obtained from an sqlite scraper [9]

- **World Resource Institute** (WRI) [10], free license power plant data for all countries, available on their powerwatch repository [11]

- **Carbon Monitoring for Action** (CARMA) [12, 13], free license power plant data for all countries

- **Energy Storage Exchange** (ESE) [14], licensed storage units data for all countries, but the data can be downloaded separately.

- **European Network of Transmission System Operators for Electricity** (ENTSO-E) [15], free license power plant data for EU-28, additional statistics about aggregated power plant capacities, which can

be used as a validation reference. We further use their annual energy generation report from 2010 [16] as an input for the hydro power plant technology.

Available power plant data sources do not cover the set of all existing power plants in Europe but rather reveal gaps for specific countries or fuel types. The package provides all the different datasets in a processed format. Therefore all original columns and all arguments are renamed according to Table 1. To bring all datasets into one coherent standard implies a lot of individual adjustment.

All datasets are stored in csv format in the 'data'-file of the repository, most of them in their original form. The calling functions read in the original files and process a precleaning such that the dataset is returned in a standardized form. The cleaning functions which are used, are situated in the **cleaning** module.

1.3 Vertical cleaning - powerplantmatching.cleaning

In order to compare and combine information from multiple databases, uniform standards must be guaranteed. That is, the datasets should be based on the same set of arguments having consistent formats. As explained in section 1.1, we reduce the set of arguments to eleven columns. This module aims to easily handle the data alignment, that is, after renaming the basic columns of an unprocessed dataset, one simply has to apply several provided functions. Furthermore, it aims to aggregate power plant units from the same power plant and with same fuel type together. Since the cleaning process is mainly modifying string expression, this module uses regex expressions for properly matching and searching for part of strings.

Shifting Information

The first step of the cleaning process implies a rearrangement of information. Datasets often state more than one information in one data column, *e.g.* as a fuel type description 'Hydro Run-of-river' or 'CHP Hard Coal'. Simply replacing such expression with general fuel type expressions would discard important information about each power plant. Therefore, a set of functions helps to parse relevant columns and if specified keywords are found, equivalent (and coherent) expressions are written into the designated columns. In

the illustrated case, this means changing 'Hydro Run-of-river' into 'Hydro' in the 'Fueltype'-column and adding 'Run-Of-River' into the 'Technology'-column, respectively, changing 'CHP Hard Coal' into 'Hard Coal' in the 'Fueltype'-column and adding 'CHP' to the 'Set'-column. The following three functions are separately dedicated to the three technological classification columns:

The command `gather_fueltype_info` primarily aims to determine a classification between 'Hard Coal' and 'Lignite' where not automatically provided. It parses the columns 'Name' and 'Technology' by default but can be applied to any set of columns. Datasets like CARMA do not differ between varieties of coal but give keywords like 'brown coal' 'lignite' in the 'Name' and 'Technology' column. If none of the expressions are found, the fuel type is automatically changed to 'Hard Coal'. This bias is motivated by a general superior number of hard coal power plants.

The command `gather_technology_info` tries to catch all technology expressions (stated in Table 1) from the 'Name' and 'Fueltype' column by default, disregarding upper-case letters. All matching expressions are uniquely added to the technology column.

The command `gather_set_info` parses the 'Name' and 'Fueltype' columns by default, searching for patterns that signify a combined heat and power plant, *e.g.* 'hkw', 'combined heat and power' or 'cogeneration'. Where those patterns occur, the 'Set'-column is set to 'CHP' otherwise, if not such patterns were found, to 'PP' which indicates an ordinary power plant.

Cleaning Expressions

It is crucial to clean string expressions in a way that does not discard important information but still reduces the expression to its essential information. In principle only the columns 'Name' and 'Fueltype' have to be cleaned in such a way.

The command `clean_powerplantname` cleans the column 'Name' of the database by deleting very frequent words, numericals and nonalphanumerical characters of the column. It returns a reduced dataset with a non-empty

'Name'-column. Power plants entries with no specific name, that is no alphabetical character in the name, are deleted, since these cannot not be considered in the matching process in section 1.4.

The command `clean_technology` returns the dataset with uniquely stated technology expressions according to the configuration set (section 1.1). Expressions as 'combined cycle', 'critical thermal' are converted into the corresponding configuration expression. Hydro technology expressions can be converted optionally, since it might be in specific cases useful to keep descriptions as 'Reservoir with natural inflow' which would be converted to 'Reservoir' otherwise.

Aggregating Power Plant Units

The aggregation of power plant units is mainly based on the module **duke**, section 1.8, which calls the java application DUKE for detecting duplicated or, in our case, similar, entries within the dataset. According to a predefined configuration file the application compares every column of every power plant entry with each other, to find entries which belong to the same power plant.

The command `aggregate_units` runs DUKE and groups together power plant units by applying aggregation rules to the different arguments, which are determined in the subfunction `prop_for_groups`. Capacities are summed up, latitude and longitude are averaged. The name of the power plant is determined by the most frequent occurring name within the set of combined units. By default, no units are brought together if their fuel type descriptions are differing.

The function provides the feature to skip the duke process and instead to use an cached aggregation file from a previous aggregation process, triggered with the argument *use_saved_aggregation*. The file is stored in data/out/-aggregation_groups_XX.csv with XX being the name for the dataset, which has to be passed to the function. Skipping the DUKE process can save time in case one needs to aggregate power plants without wanting to run the aggregation algorithm again.

1.4 Combining Data From Different Sources - powerplantmatching.matching

Once a set of uniformly standardized databases has been created, one can merge them, such that entries from different sources, that represent the same power plant, are brought together. Thus, the datasets can complement each other and improve their reliability. The package **matching** supplies several modes of linking and combining different datasets. Similar to the module **cleaning** 1.3, these functions are mainly based on the module **duke** 1.8 which is set to record linkage mode [17].

Combining Model

One main task of this package is to extent the matching capability of DUKE, which restricts the number of datasets to maximally two. Therefore, it is only possible to create different sets of pair-links between the different combinations of the datasets.
Assume, we have three datasets to combine, $\mathbf{a} = \{a_1, \cdots, a_l\}$, $\mathbf{b} = \{b_1, \cdots, b_m\}$ and $\mathbf{c} = \{c_1, \cdots, c_n\}$. In order to assemble all to one dataset **a-b-c**, we separately link **a-b**, **a-c** and **b-c**. We have to guarantee that each entry has maximally one link to another data set, *e.g.* a_i only has maximally one link to $\{b_1, \cdots, b_m\}$. These three sets of pair-links build the base of the combined set **a-b-c**. Every pair appearing in the original sets has to reappear in the combined set. Furthermore, also indirect links are created in case two entries from different datasets, which are not directly linked, match the same entry from a third dataset. The following Table 2 shows an short example.

a-b	
a_1	b_1
a_2	b_3
$\vdots$	$\vdots$

$+$

a-c	
a_1	c_2
a_2	c_3
$\vdots$	$\vdots$

$+$

b-c	
b_3	c_3
b_4	c_5
$\vdots$	$\vdots$

$\rightarrow$

a-b-c		
a_1	b_1	c_2
a_2	b_3	c_3
	b_4	c_5
$\vdots$	$\vdots$	$\vdots$

Table 2: Example of combining sets of pair-links

One easily can see that in **a-b-c** the indices b_1 and c_2 are linked despite they have not matched in **b-c**. In the second row, however, all entries of the combined set **a-b-c** have matched directly among each other. In the third row, it

remains a single pair, which has no further links. This process is applicable for any number of data sets.

For implementing the previously described process the following functions are available:

In order to guarantee a one-to-one link for two matched datasets, the function `best_matches` takes the link output from DUKE and only returns the matches with the highest matching scores if multiple links per power plant occur.

The function `cross_matches` combines multiple sets of pairs and returns one consistent dataframe, according to the procedure in Table 2. Hence, identifiers of two datasets can appear in one row, even though they did not match directly but indirectly through a connecting identifier of another database.

Main Use

In order to create a combined dataset, the module provides three convenient functions that barely differ.

The function `link_multiple_datasets` initiates a DUKE-based horizontal matching of multiple databases. It returns the indices pointing out the matches between the datasets. All properties of the columns 'Name', 'Fueltype', 'Country', 'Capacity' and 'Geoposition' are being compared, in order to work out the same powerplant in different datasets. The function uses are mainly based on the functions `duke`, `best_matches` and `cross_matches`. As mentioned before, the match is in one-to-one mode.

The function `combine_multiple_datasets` works the similar way but instead of returning all matching indices, it returns all information from each dataset combined in one multi-indexed pandas.Dataframe. The first column level indicates the property given in 1, the second column level gives the dataset name.

The function `reduce_matched_dataframe` applies different aggregation rules, given in Table 3, to the output of `combine_multiple_datasets`. Thereby, the large combined dataset is reduced to a small dataset, such that each power plant has only one claim per property.

Column	Rule
Name	Every name of the different databases
Fueltype	Most frequent claimed one
Technology	All different Technology in a row
Country	Take the uniquely stated country
Capacity	Median/Mean
lat	Mean
lon	Mean
File	All files in a row
projectID	Python dictionary referencing all original powerplants that are included

Table 3: Aggregation rules for reducing the matched entries of multiple data sets

Note that for latitude and longitude we aggregate the claims by averaging. As for the capacity, it turned out that taking the median or the mean achieves reasonable results. The claims for country cannot differ, otherwise the power plants do not match.

Example

As for an practical insight, we will step-by-step combine two subsets to one. Therefore, we first of all call define two datasets, 'Dataset1' and 'Dataset2', from our data sources GEO and CARMA:

```
import powerplantmatching as pm
Dataset1 = pm.data.GEO().head()
Dataset2 = pm.data.CARMA().loc[[8,13,14,15,19,21]]
```

which, in reduced form, initiates the datasets

	Name	Fueltype	Technology	Country	Capacity	lat	lon
0	Aarberg	Hydro	Run-Of-River	CH	15.5	47.0378	7.272
1	Aberthaw	Coal	Thermal	UK	1500	51.3873	-3.4049
2	Abono	Coal	Thermal	ES	921.7	43.5528	-5.7231
3	Abwinden asten	Hydro	nan	AT	168	48.248	14.4305
4	Aceca	Oil	CHP	ES	629	39.941	-3.8569
5	Aceca fenosa	Natural Gas	CCGT	ES	400	39.9427	-3.8548

Table 4: 'Dataset1': Extract from the GEO dataset

and

	Name	Fueltype	Technology	Country	Capacity	lat	lon
8	Aarberg	Hydro	nan	CH	14.609	47.044	7.2758
13	Abbey mills	Oil	nan	UK	6.4	51.687	-0.0042
14	Abertay	Other	nan	UK	8	57.178	-2.1868
15	Aberthaw	Coal	nan	UK	1552.5	51.387	-3.4067
19	Ablass	Wind	nan	DE	18	51.233	12.95
21	Abono	Coal	nan	ES	921.7	43.559	-5.7228

Table 5: 'Dataset2': Extract from the CARMA dataset

Apparently the entries 0,1 and 2 of 'Dataset1' relate to the same power plants as the entries 8, 15 and 21 of 'Dataset2'. Applying the matching algorithm with

```
pm.matching.link_multiple_datasets([Dataset1,Dataset2],
                                   ['Dataset 1', 'Dataset 2'])
```

returns

	Dataset 1	Dataset 2
0	0	8
1	1	15
2	2	21

Table 6: Links for 'Dataset1' and 'Dataset2'

Whereas in order to obtain the full combined dataset, we use

```
matched = pm.matching.combine_multiple_datasets([Dataset1,Dataset2],
                          ['Dataset 1', 'Dataset 2'])
print matched
```

which returns the following output.

	Name		Fueltype		Technology	
	Dataset 1	**Dataset 2**	**Dataset 1**	**Dataset 2**	**Dataset 1**	**Dataset 2**
0	Aarberg	Aarberg	Hydro	Hydro	Run-Of-River	nan
1	Aberthaw	Aberthaw	Coal	Coal	Thermal	nan
2	Abono	Abono	Coal	Coal	Thermal	nan

	Country		Capacity		lat	
	Dataset 1	**Dataset 2**	**Dataset 1**	**Dataset 2**	**Dataset 1**	**Dataset 2**
0	CH	CH	15.5	14.609	47.0378	47.0444
1	UK	UK	1500	1552.5	51.3873	51.3875
2	ES	ES	921.7	921.7	43.5528	43.5588

	lon		File		projectID	
	Dataset 1	**Dataset 2**	**Dataset 1**	**Dataset 2**	**Dataset 1**	**Dataset 2**
0	7.272	7.27578	nan	nan	1180	64
1	-3.4049	-3.40675	nan	nan	246	123
2	-5.7231	-5.72287	nan	nan	203	139

Table 7: Combined dataset from the two extracts 'Dataset1' and 'Dataset2'

Now, one can easily reduce the dataframe by applying aggregation rules from Table 3:

```
pm.matching.reduce_matched_dataframe(matched)
```

This returns the following more handy Table

	Dataset 1	Dataset 2	Fueltype	Technology	Country
0	Aarberg	Aarberg	Hydro	Run-Of-River	CH
1	Aberthaw	Aberthaw	Coal	Thermal	UK
2	Abono	Abono	Coal	Thermal	ES

	Capacity	lat	lon	projectID
0	15.5	47.04	7.273	{'Dataset 1': 1180, 'Dataset 2': 64}
1	1552.5	51.38	-3.405	{'Dataset 1': 246, 'Dataset 2': 123}
2	921.7	43.55	-5.722	{'Dataset 1': 203, 'Dataset 2': 139}

Table 8: Reduced merged data set from the two extracts. Whereas both original power plant names are kept, the multiple statements of the other arguments are combined through the rules in Table 3 (page 10).

Note, that the names from the different sources are kept for ease of referencing, whereas the claims about the other plant parameters have been reduced an aggregate value using the rules.

1.5 Fine Tuning - powerplantmatching.heuristics

For final adjustments the module **heuristics** provides functions to manipulate data. Data completeness is one of the most important features for reliable simulations. Even through the matching process of multiple data sets, not all existing power plants are included. Two functions help to manually fine tune the (matched) data:

The function `rescale_capacities_to_country_totals` returns an extra column 'Scaled Capacity' with a linearly up or down scaled capacity for each power plant, to match the statistics of the ENTSO-E country totals. The latter provides country-wise information about the aggregated capacity for each fuel type. This allows a capacity adjustment for power plants of a selected fuel type but can also applied to all fuel types. The scaling factor is determined by the ratio between the data set's capacity totals per fueltype and country and the ENTSO-E capacity totals per fueltype and country.

The function `extend_by_non_matched` supports an non-artificial data extension for matched datasets. It adds non-matched power plants of a selected

source to the matched dataset. This is convenient in the case of uneven reliable input data sources. Reliable data sources will then appear completely in the matched dataset.

The package aims to provide a further function for filling the information about missing hydro technology, which will also be part of the **heuristics**. By now this in done externally, and will be presented in section 2

1.6 Processed Data - powerplantmatching.collection

For simplifying the access to processed data, the module **collection** provides multiple functions for read-in or updating the matched data.

The core function `Collection` takes a list of data sources names, *e.g.* "['GEO', 'OPSD']", as an argument and reads-in the resulting matched dataset from the stored file in powerplantmatching/data/out if available. In case no stored file is found, the function automatically starts a matching process, including vertical cleaning and horizontal matching of the dataset combination. After the process, the matched dataset is returned and added to the data file. By setting *reduced=True*, the reduced dataset, resulting from `reduce_matched_dataframe`, is returned. In order to speed up the vertical cleaning, one can set *use_saved_aggregation=True*, which takes stored information about vertical cleaning and avoids to rerunning DUKE.

Convenience functions as `Carma_ENTSOE_GEO_OPSD_matched_reduced` are based on the same concept, but represent a shortcut for the given combination in the function name.

1.7 Auxiliary Functions - powerplantmatching.utils

Data processing requires constant inspection and validation. For the power plant data collection it is convenient and helpful to double-check the aggregated capacities for each country, fuel type or both. The module **powerplantmatching.utils** contains functions for data inspection and plotting, for revising the property of the data or to compare two or more data sets against each other. The function `lookup()` returns a lookup Table for the dataset. By setting the keyword '*by*' one can define the aggregation type:

```
import powerplantmatching as pm
geo = pm.data.GEO()
print pm.utils.lookup(geo, by='Country')
print pm.utils.lookup(geo, by='Fueltype')
print pm.utils.lookup(geo, by='Country, Fueltype')
```

returns

Country	Capacity
Austria	12280
Belgium	10317
Bulgaria	9481
Croatia	3274
Czech Republic	9537
⋮	⋮
Total	515226

Table 9: Aggregated by country, third output

Fueltype	Capacity
Coal	150638
Geothermal	562
Hydro	86775
Natural Gas	154617
Nuclear	103040
Oil	19070
Waste	522
Total	515226

Table 10: Aggregated by fuel type, second output

	Austria	Belgium	Bulgaria	Croatia	Czech Republic	⋯
Coal	1003	889	4970	330	8033	⋯
Geothermal	0	0	0	0	0	⋯
Hydro	8553	1393	2512	1623	1505	⋯
Natural Gas	2724	4906	0	582	0	⋯
Nuclear	0	3129	2000	0	0	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋱

Table 11: Aggregated by country and fuel-type, third output

Passing a list of data sets to the function extends the Table such that one can easily compare the data sets. As a means of comparison, it is convenient to plot the aggregated totals, as done by the following code:

```
import powerplantmatching as pm

geo = pm.utils.set_uncommon_fueltypes_to_other(pm.data.GEO())
```

```
carma = pm.utils.set_uncommon_fueltypes_to_other(pm.data.CARMA())
entsoe = pm.utils.set_uncommon_fueltypes_to_other(pm.data.ENTSOE())

pm.utils.lookup([geo, carma, entsoe],
                    keys=['GEO', 'CARMA', 'ENTSO-E'],
                     by='Fueltype').plot(kind='bar')
```

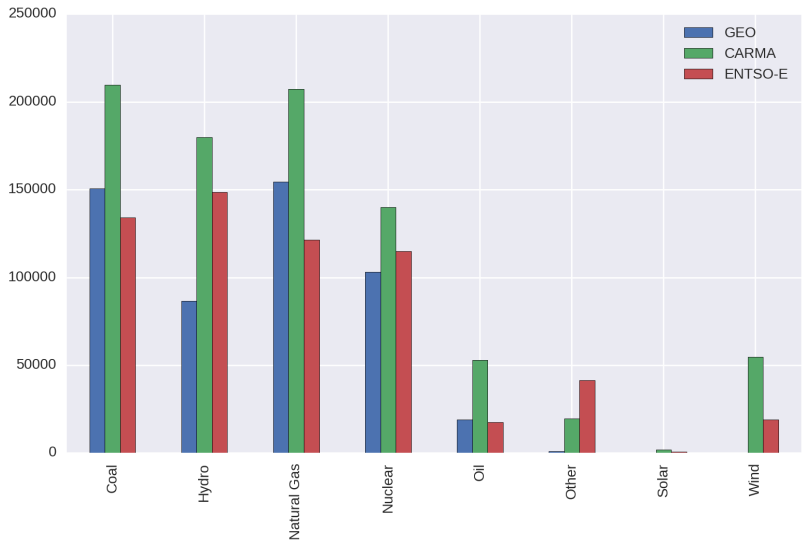

Figure 2: Plot created from the preceding code

Note that in order to avoid uncommon fuel type descriptions the function `set_uncommon_fueltypes_to_other` changes those into "Other".

For convenient data read-in, the function `read_csv_if_string` was defined, which reads in the csv file of a given path and returns it in the correct format.

For data sets which only provide location descriptions, such as city, street or postcode, the function `parse_Geoposition` sets a nominatim request for the geoposition of a specific location in a country. It returns a tuples with (latitude, longitude), if the request was successful, and None otherwise. In particular, this was used for the WRI data and can be used for the PLATTS data, which initially does not provide geopositions.

1.8 Matching Engine - powerplantmatching.duke

The cleaning and matching process, presented in sections 1.3 and 1.4, heavily relies on DUKE [18], a java application specialized for deduplicating and linking data. The module **duke** allows calling the java application from the python environment.

DUKE provides many built-in comparators such as numerical, string or geoposition comparators. According to the settings in the configuration file, the engine does a detailed comparison for each single argument (power plant name, fuel-type etc.) using adjusted comparators and probability boundaries. The latter gives the boundary for the score that DUKE states for the match of two power plant attributes. For example, if the upper boundary for the country column is at 0.7, and two power plants have exact the same country string, than DUKE states a 'country score' of 0.7. In this way scores are stated for each column within the probability boundaries. From the individual columns scores it computes a compound score for the likeliness that the two power plant records refer to the same power plant. If the score exceeds a given threshold, the two records of the power plant are linked and merged into one data set. Table 12 shows the set of compared columns with the corresponding DUKE comparators and probability ranks for the horizontal matching (two different sources). The threshold for two power plants to be linked is 0.985.

column	comparator	probability boundary
Power plant name	JaroWinklerTokenized	[0.09, 0.99]
Fueltype	QGramComparator	[0.09, 0.7]
Capacity	NumericComparator	[0.1, 0.75]
lat	GeoComparator	[0.1, 0.8]
lon	GeoComparator	[0.1, 0.8]
Country	QGramComparator	[0.0, 0.53]

Table 12: DUKE comparison configuration. The table shows the compared columns and applied comparators with probability boundaries when matching two different datasets.

2 Hydro Technology

Whereas all of the power plants have unique fuel type specifications, the technology argument is not necessarily given for each power plant entry. However, for modeling hydro power plants its makes a big difference whether one can store excess energy as in pumped hydro storage, or only gain power dependent from the inflow, as for run-of-river power plants. In order to fill the missing gap an package extern script helps to identify the technology of hydro power plants. It mainly consists of a supervised learning algorithm, of the *scikit-learn* python package [19], which takes the already classified hydro plants as a training set in order to determine the technology of the unspecified plants.

The combined data, matching the sources Carma, ENTSOE, ESE, GEO, OPSD, WRI, includes 1737 hydro power plants. Of those, 1310 power plants have specified technologies and 427 are not specified. Since our data already covers most of the pumped storage power plants in Europe (provided by the ESE data base), we only keep run-of-river power plants and reservoirs, which reduces the training set to 1157 power plants. Thereby, we assume that the 427 unspecified hydro power plants are either run-of-river power plants or reservoirs.

A European height map, retrieved from [20], is used to align altitude and slope of the surrounding to the training set. Those properties are passed to the classifier. Therefore, the classifier can identify relations as flat surroundings and low altitude with run-of-river power plants, steep surrounding and high altitude with reservoir. The missing technology data is then fitted by the classifier. For further improvement, the fitting weights of the classifier are adjusted for each country, such that the aggregated capacity values roughly match the stated ENTSOE net generation capacity totals [16]. In Figure 3, we show the resulting technology classifications for each country. Mountainous countries as Switzerland and Norway include a strong amount of hydro power plants, most of them pumped hydro or reservoirs. Countries as France additionally have a lot of run-of-river power plants in the flatter regions.

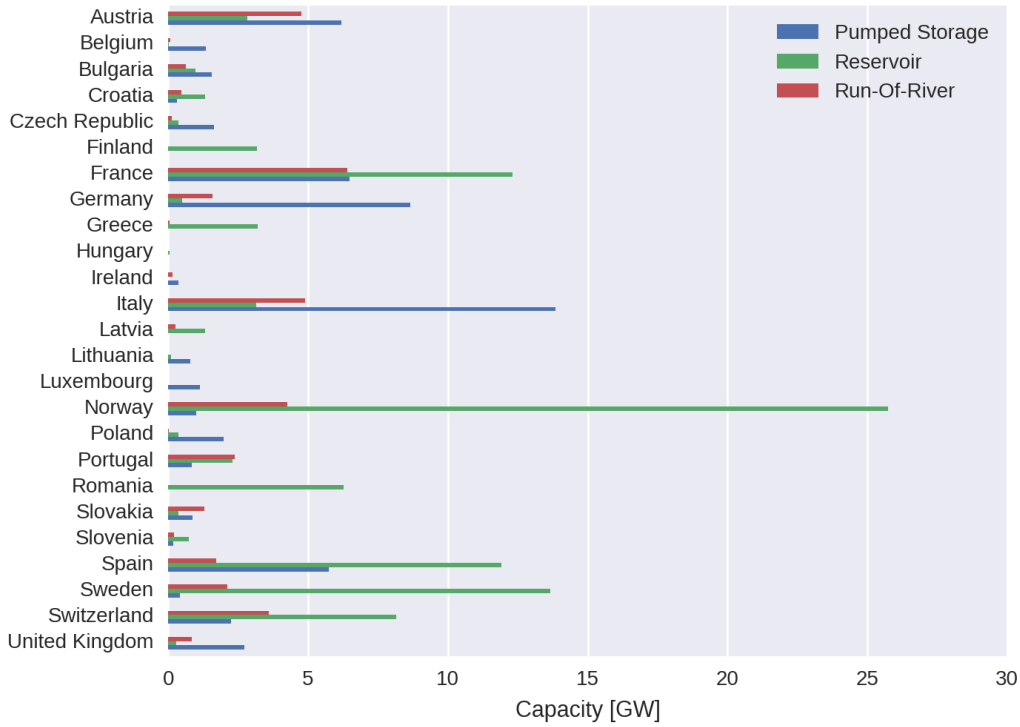

Figure 3: Distribution of the hydro technologies within the countries. The missing technologies are fitted to the total capacities stated in by the ENTOSE net generation capacities [16].

3 Resulting Power Plant Data

The processed data, provided by **powerplantmatching.collection**, combines the standardized data of the different input datasets described in section 1.2. In this section, we compare the aggregated totals of the resulting data with the original input data and the ENTSOE-statistics. This serves as a rough validation of the data and reveals missing gaps.

In Figure 4, we show the fuel type aggregated totals of the input databases in comparison to the matched database. As one can see, the heterogenous distribution of the input data averages out through the matching process. Note that the input data hardly provides information about solar and wind power generators. Furthermore, the CARMA database includes more power plants for all fuel types except for lignite, as the CARMA database does not

distinguish between lignite and hard coal and all coal power plants are labeled as hard coal. For hydro power plants all databases provide information as pumped hydro power plants of the ESE database are also included.

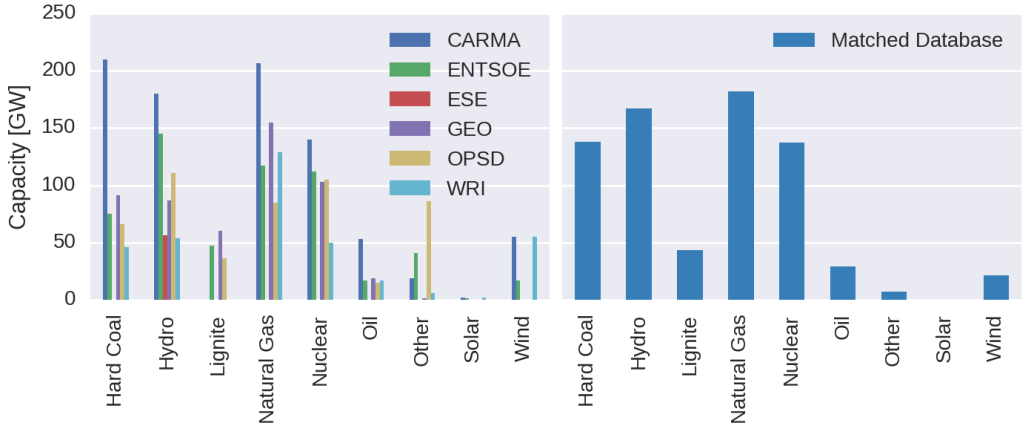

Figure 4: Comparison of the single databases capacity totals with the matched database.

The spatial distribution of the power plants and their according capacity size is shown in Figure 5. The circle size in the legend refers to a capacity of 4 GW. The largest European power plant is the 'Gravelines Nuclear Power Station' in northern France with 5706 MW. As one can see, most of the hydro power plants are located around the Alps and Scandinavia. A high conventional generator density occurs around Belgium and North West Germany.

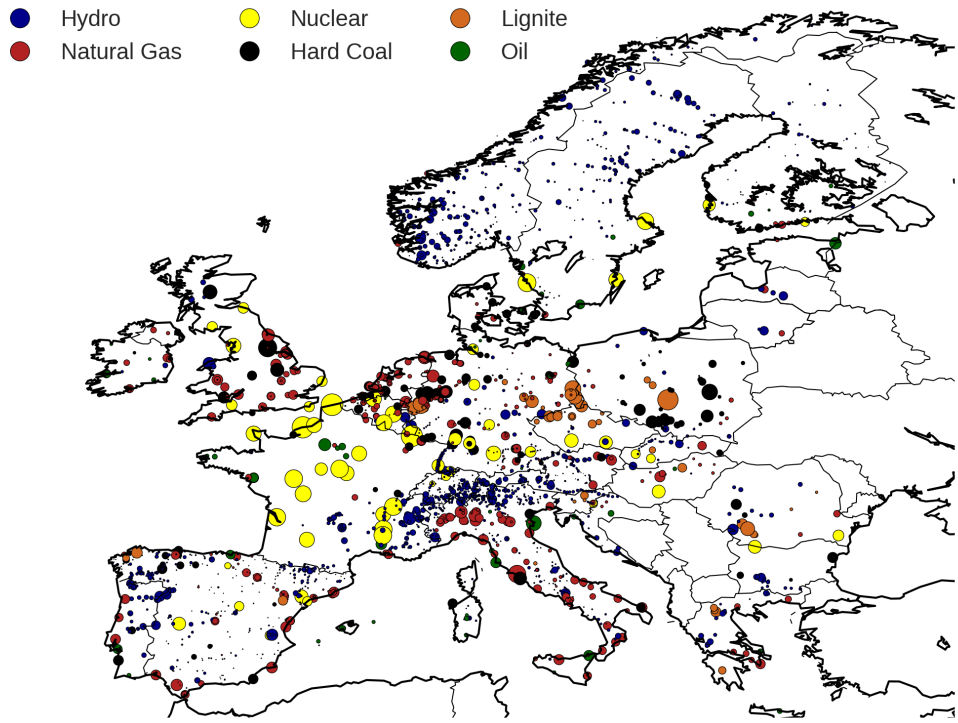

Figure 5: Map of power plants, the size of the circles are proportional to the capacity of the power plants. The circles in the legend correspond to a capacity of 4 GW.

A more detailed description of the fuel type and country aggregated capacity totals is shown in Figure 6. Here, the aggregated totals for each fuel type, except for wind and solar, are displayed for each country in comparison with the ENTSOE statistics, whereas all corresponding values are linked together. Uncommon fuel type descriptions as 'Bioenergy' or 'Waste' are added up to 'Other'. The country totals are added on top.

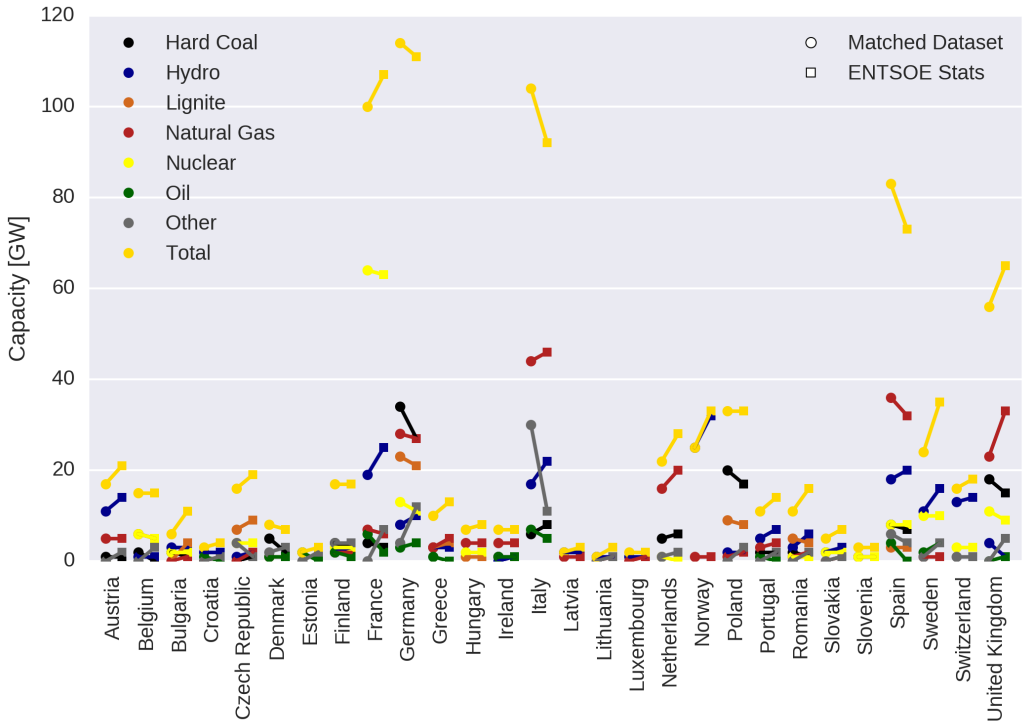

Figure 6: Country and fuel type specific comparison of the matched database with the aggregated capacities of the ENTSOE statistics.

As one can see, most of the country totals are nearly covered. The biggest gaps occur in France, Germany, Sweden and UK. The fuel type 'Other' is often underestimated by the matched dataset, especially in Austria, France, Germany and Italy, as well as 'Natural Gas' in Germany. On the other hand the dataset reveal some (small) overestimations as for Germany with fuel type 'Hard Coal' and Italy with fuel type 'Other'. Note that some of these errors may occur because of different fuel type specifications of the initial datasets.

In the simple fuel type aggregation, see Figure 7, the biggest gaps occur for wind and solar power generators. Also the fuel types 'Other' and 'Hydro' are not well covered yet. Thus, it is to assume that the matched dataset covers most of the conventional European power plants. Excluding wind and solar, the matched dataset represents a total European capacity of 682 GW whereas the ENTSO-E statistics claim 769 GW.

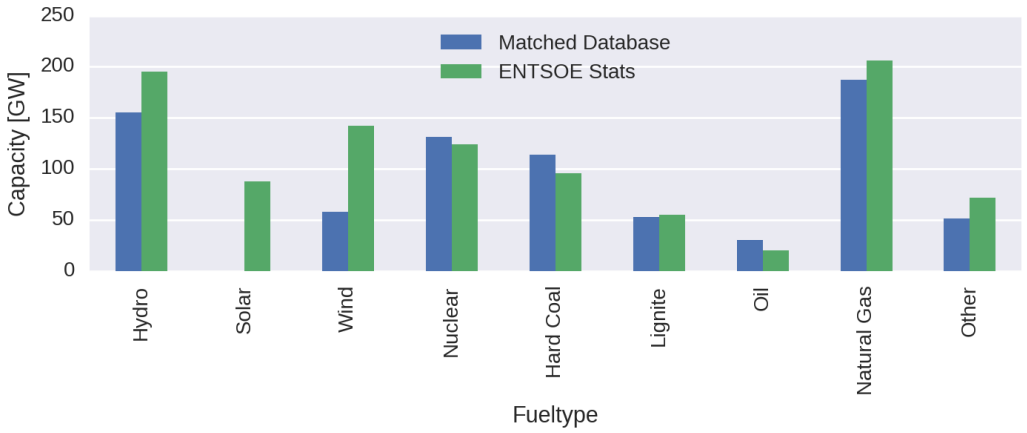

Figure 7: Fuel type specific comparison of the matched database and the ENTSOE statistics. The total capacity is at 72% (88% without wind and solar) of the ENTSOE total.

3.1 Upcoming Tasks

Even though the resulting dataset covers most of the conventional power plants so far, the **powerplantmatching** package is still in process. In the following, we briefly discuss the upcoming tasks and planned changes.

In order to further improve the resulting data, the gaps mentioned in the previous section should be dealt with. Since wind and solar data is difficult to obtain, the priority is set on conventional data. A proper analysis of the input data should clarify whether the gaps emerge from missing input information or from inaccurate standardizing or matching.

Except for single power plant entries, which were double checked manually, the data could not yet be validated on plant level. We plan to (let) compare the processed data (by a third party) with a commercial European power plant database like PLATTS [21].

The aggregation of power plant units during the vertical cleaning was not yet validated. With a validation set that provides correct power plant information on unit level and on aggregated level, the resulting set of `aggregate_units` in section 1.3 could straightforwardly be double-checked.

For usability of the data, it is crucial to Figure out how to distinguish and deal with capacities given as net and gross values. Since most of the original databases do not include information about the classification of the capacity value, taking the mean or the maximum of the claimed capacities when aggregating power plants leads to incorrect values. An idea of a workaround is to fix a priority range of the databases according to reliability of the database giving the net capacity. When aggregating claims together, this range determines which database information should be given priority.

References

[1] FRESNA - FIAS Renewable Energy System and Network Analysis, "Powerplantmatching," https://github.com/FRESNA/powerplantmatching/commits/master, 2016, Online, retrieved July 2016.

[2] "Collective Nonlinear Dynamics of Complex Networks," www.condynet.de/.

[3] "Federal Ministry of Education and Research," https://www.bmbf.de/en/.

[4] "Stromnetze - Forschungsinitiative der Bundesregierung," http://forschung-stromnetze.info/projekte/grundlagen-und-konzepte-fuer-effiziente-dezentrale-stromnetze/.

[5] Schill, Wolf-Peter et. al., "Open Power System Data," http://data.open-power-system-data.org/, 2016, Online, retrieved September 2016.

[6] "Bundesnetzagentur," https://www.bundesnetzagentur.de/DE/Sachgebiete/ElektrizitaetundGas/Unternehmen_Institutionen/Versorgungssicherheit/Erzeugungskapazitaeten/Kraftwerksliste/kraftwerksliste-node.html.

[7] "Umwelt Bundesamt," https://www.umweltbundesamt.de/daten/energiebereitstellung-verbrauch/kraftwerke.

[8] Rajan Gupta, Harihar Shankar et. al., "Global Energy Observatory," http://globalenergyobservatory.org/, 2016, Online, retrieved September 2016.

[9] Hörsch, Jonas and David, Chris, "GEO data sqlite scraper," https://morph.io/coroa/global_energy224_observatory_power_plants, 2016, Online, retrieved September 2016.

[10] World Resources Institute, "PowerWatch: An open-source global power plant database project," https://github.com/Arjay7891/WRI-Powerplant, 2016, Online, retrieved September 2016.

[11] World Resources Institute , "PowerWatch data file," https://github.com/Arjay7891/WRI-Powerplant/blob/master/output_database/powerwatch_data.csv, 2016, Online, retrieved September 2016.

[12] D. Wheeler and K. Ummel, "Calculating CARMA: global estimation of CO2 emissions from the power sector," *Available at SSRN 1138690*, 2008.

[13] K. Ummel, "CARMA revisited: an updated database of carbon dioxide emissions from power plants worldwide," *Center for Global Development Working Paper*, no. 304, 2012.

[14] Sandia National Laboratories, "DOE Global Energy Storage Database," http://www.energystorageexchange.org/projects, 2016, Online, retrieved September 2016.

[15] European Transmission System Operators, "Country-specific hourly load data," https://www.entsoe.eu/, 2011.

[16] ——, "Net Generating Capacity," https://www.entsoe.eu/db-query/miscellaneous/net-generating-capacity, 2011.

[17] Lars Marius Garshol, "Duke - recordlinkagemode," https://github.com/larsga/Duke/wiki/RecordLinkageMode, 2016, Online, retrieved July 2016.

[18] ——, "Duplicate Killer," https://github.com/larsga/Duke, 2016, Online, retrieved July 2016.

[19] "Scikit-learn, support vector machines," http://scikit-learn.org/stable/modules/svm.html, 2016, Online, retrieved Mai 2017.

[20] National Oceanic and Atmospheric Administration, "National Centers for Environmental Information," http://www.ngdc.noaa.gov/mgg/global/global.html, 2016, Online, retrieved September 2016.

[21] S&P Global Platts, "Powerplantmatching," http://www.platts.com/products/world-electric-power-plants-database, 2016, Online, retrieved April 2017.