eGo$^N Data$

Release 0.0.0

Jun 30, 2022

Contents

CHAPTER 1

Getting Started

1.1 Pre-requisites

In addition to the installation of Python packages, some non-Python packages are required too. Right now these are:

- Docker: Docker is used to provide a PostgreSQL database (in the default case).

  Docker provides extensive installation instruction. Best you consult their docs and choose the appropriate install method for your OS.

  Docker is not required if you use a local PostreSQL installation.

- The *psql* executable. On Ubuntu, this is provided by the *postgresql-client-common* package.
- Header files for the `libpq5` PostgreSQL library. These are necessary to build the `psycopg2` package from source and are provided by the `libpq-dev` package on Ubuntu.
- osm2pgsql On recent Ubuntu version you can install it via `sudo apt install osm2pgsql`.
- postgis On recent Ubuntu version you can install it via `sudo apt install postgis`.
- osmTGmod resp. osmosis needs java. On recent Ubuntu version you can install it via `sudo apt install default-jre` and `sudo apt install default-jdk`.
- conda is needed for the subprocess of running pypsa-eur-sec. For the installation of miniconda, check out the conda installation guide.
- pypsa-eur-sec resp. Fiona needs the additional library `libtbb2`. On recent Ubuntu version you can install it via `sudo apt install libtbb2`
- gdal On recent Ubuntu version you can install it via `sudo apt install gdal-bin`.
- curl is required. You can install it via `sudo apt install curl`.
- To download ERA5 weather data you need to register at the CDS registration page and install the CDS API key as described here You also have to agree on the terms of use
- Make sure you have enough free disk space (~350 GB) in your working directory.

1.2 Installation

Since no release is available on PyPI and installations are probably used for development, cloning it via

```
git clone git@github.com:openego/eGon-data.git
```

and installing it in editable mode via

```
pip install -e eGon-data/
```

are recommended.

In order to keep the package installation isolated, we recommend installing the package in a dedicated virtual environment. There's both, an external tool and a builtin module which help in doing so. I also highly recommend spending the time to set up virtualenvwrapper to manage your virtual environments if you start having to keep multiple ones around.

If you run into any problems during the installation of `egon.data`, try looking into the list of known installation problems we have collected. Maybe we already know of your problem and also of a solution to it.

1.3 Run the workflow

The `egon.data` package installs a command line application called `egon-data` with which you can control the workflow so once the installation is successful, you can explore the command line interface starting with `egon-data --help`.

The most useful subcommand is probably `egon-data serve`. After running this command, you can open your browser and point it to *localhost:8080*, after which you will see the web interface of Apache Airflow with which you can control the eGo^n data processing pipeline.

If running `egon-data` results in an error, we also have collected a list of known runtime errors, which can consult in search of a solution.

Warning: A complete run of the workflow might require much computing power and can't be run on laptop. Use the *test mode* for experimenting.

Warning: A complete run of the workflow needs loads of free disk space (~350 GB) to store (temporary) files.

1.3.1 Test mode

The workflow can be tested on a smaller subset of data on example of the federal state of Schleswig-Holstein. Data is reduced during execution of the workflow to represent only this area.

Warning: Right now, the test mode is set in *egon.data/airflow/pipeline.py*.

CHAPTER 2

Workflow

2.1 Project background

egon-data provides a transparent and reproducible open data based data processing pipeline for generating data models suitable for energy system modeling. The data is customized for the requirements of the research project eGo_n. The research project aims to develop tools for an open and cross-sectoral planning of transmission and distribution grids. For further information please visit the eGo_n project website. egon-data is a further development of the Data processing developed in the former research project open_eGo. It aims for an extensions of the data models as well as for a better replicability and manageability of the data preparation and processing. The resulting data set serves as an input for the optimization tools eTraGo, ding0 and eDisGo and delivers for example data on grid topologies, demands/demand curves and generation capacities in a high spatial resolution. The outputs of egon-data are published under open source and open data licenses.

2.2 Data

egon-data retrieves and processes data from several different external input sources which are all freely available and published under an open data license. The process handles data with different data types, such as spatial data with a high geographical resolution or load/generation time series with an hourly resolution.

2.3 Execution

In principle egon-data is not limited to the use of a specific programming language as the workflow integrates different scripts using Apache Airflow, but Python and SQL are widely used within the process. Apache Airflow organizes the order of execution of processing steps through so-called operators. In the default case the SQL processing is executed on a containerized local PostgreSQL database using Docker. For further information on Docker and its installation please refer to their documentation. Connection information of our local Docker database are defined in the corresponding docker-compose.yml

The egon-data workflow is composed of four different sections: database setup, data import, data processing and data export to the OpenEnergy Platform. Each section consists of different tasks, which are managed by Apache Airflow and correspond with the local database. Only final datasets which function as an input for the optimization tools or selected interim results are uploaded to the Open Energy Platform. The data processing in egon-data needs to be performed locally as calculations on the Open Energy Platform are prohibited. More information on how to run the workflow can be found in the getting started section of our documentation.

2.4 Versioning

Warning: Please note, the following is not implemented yet, but we are working on it.

Source code and data are versioned independendly from each other. Every data table uploaded to the Open Energy Platform contains a column 'version' which is used to identify different versions of the same data set. The version number is maintained for every table separately. This is a major difference to the versioning concept applied in the former data processing where all (interim) results were versioned under the same version number.

CHAPTER 3

Troubleshooting

Having trouble installing or running `eGon-data`? Here's a list of known issues including a solution.

3.1 Installation Errors

These are some errors you might encounter while trying to install *egon.data*.

3.1.1 importlib_metadata.PackageNotFoundError: No package metadata ...

It might happen that you have installed *importlib-metadata=3.1.0* for some reason which will lead to this error. Make sure you have *importlib-metadata>=3.1.1* installed. For more information read the discussion in issue #60.

3.2 Runtime Errors

These are some of the errors you might encounter while trying to run `egon-data`.

3.2.1 ERROR: Couldn't connect to Docker daemon ...

To verify, please execute `docker-compose -f <(echo {"service": {"image": "hellow-world"}}) ps` and you should see something like

```
ERROR: Couldn't connect to Docker daemon at http+docker://localunixsocket - is it␣
↪running?

If it's at a non-standard location, specify the URL with the DOCKER_HOST environment
variable.
```

This can have at least two possible reasons. First, the docker daemon might not be running. On Linux Systems, you can check for this by running `ps -e | grep dockerd`. If this generates no output, you have to start the docker daemon, which you can do via `sudo systemctl start docker.service` on recent Ubuntu systems.

Second, your current user might not be a member of the *docker* group. On Linux, you can check this by running `groups $(whoami)`. If the output does not contain the word *docker*, you have to add your current user to the *docker* group. You can find more information on how to do this in the docker documentation. Read the initial discussion for more context.

3.2.2 [ERROR] Connection in use ...

This error might arise when running `egon-data serve` making it shut down early with `ERROR - Shutting down webserver`. The reason for this is that the local webserver from a previous `egon-data serve` run didn't shut down properly and is still running. This can be fixed by running `ps -eo pid,command | grep "gunicorn: master" | grep -v grep` which should lead to output like `NUMBER gunicorn: master [airflow-webserver]` where `NUMBER` is a varying number. Once you got this, run `kill -s INT NUMBER`, substituting `NUMBER` with what you got previously. After this, `egon-data serve` should run without errors again.

3.2.3 [ERROR] Cannot create container for service egon-data-local-database ...

During building the docker container for the Postgres database, you might encounter an error like

```
ERROR: for egon-data-local-database  Cannot create container for service
egon-data-local-database: Conflict. The container name
"/egon-data-local-database" is already in use by container
"1ff9aadef273a76a0acbf850c0da794d0fb28a30e9840f818cca1a47d1181b00".
You have to remove (or rename) that container to be able to reuse that name.
```

If you're ok with deleting the data, stop and remove the container by

```
docker stop egon-data-local-database
docker rm -v egon-data-local-database
```

The container and its data can be kept by renaming the docker container.

```
docker rename egon-data-local-database NEW_CONTAINER_NAME
```

3.2.4 Working with multiple instances of egon-data

To make sure parallel installations of egon-data are not conflicting each other users have to set different values for the following options in the configuration:

```
--airflow-port
--compose-project-name
--database-port
--docker-container-name
```

3.3 Other import or incompatible package version errors

If you get an `ImportError` when trying to run `egon-data`, or the installation complains with something like

```
first-package a.b.c requires second-package>=q.r.r, but you'll have
second-package x.y.z which is incompatible.
```

you might have run into a problem of earlier `pip` versions. Either upgrade to a `pip` version >=20.3 and reinstall `egon.data`, or reinstall the package via `pip install -U --use-feature=2020-resolver`. The `-U` flag is important to actually force a reinstall. For more information read the discussions in issues #36 and #37.

CHAPTER 4

Data

4.1 Scenarios

4.2 Published data

CHAPTER 5

Literature

CHAPTER 6

Contributing

The research project eGo_n and egon-data are collaborative projects with several people contributing to it. The following section gives an overview of applicable guidelines and rules to enable a prospering collaboration. Any external contributions are welcome as well, and they are greatly appreciated! Every little bit helps, and credit will always be given.

6.1 Bug reports and feature requests

The best way to report bugs, inform about intended developments, send feedback or propose a feature is to file an issue at https://github.com/openego/eGon-data/issues.

Please tag your issue with one of the predefined labels as it helps others to keep track of unsolved bugs, open tasks and questions.

To inform others about intended developments please include: * a describtion of the purpose and the value it adds * outline the required steps for implementation * list open questions

When reporting a bug please include all information needed to reproduce the bug you found. This may include information on

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.

6.2 Contribution guidelines

6.2.1 Development

Adding changes to the egon-data repository should follow some guidelines:

1. Create an issue in our repository to describe the intended developments briefly
2. Create a branch for your issue related development from the dev-branch following our branch naming convention:

   ```
   git checkout -b `<prefix>/#<issue-id>-very-brief-description`
   ```

   where *issue-id* is the issue number on GitHub and *prefix* is one of

   - features
   - fixes
   - refactorings

   depending on which one is appropriate. This command creates a new branch in your local repository, in which you can now make your changes. Be sure to check out our *style conventions* so that your code is in line with them. If you don't have push rights to our repository, you need to fork it via the "Fork" button in the upper right of the repository page and work on the fork.
3. Make sure to update the documentation along with your code changes
4. When you're done making changes run all the checks and docs builder with tox one command:

   ```
   tox
   ```

5. Commit your changes and push your branch to GitHub:

   ```
   git add -p
   git commit
   git push origin features/#<issue-id>-very-brief-description
   ```

   Note that the `-p` switch will make `git add` iterate through your changes and prompt for each one on whether you want to include it in the upcoming commit. This is useful if you made multiple changes which should conceptually be grouped into different commits, like e.g. fixing the documentation of one function and changing the implementation of an unrelated one in parallel, because it allows you to still make separate commits for these changes. It has the drawback of not picking up new files though, so if you added files and want to put them under version control, you have to add them explicitly by running `git add FILE1 FILE2 ...` instead.
6. Submit a pull request through the GitHub website.

6.2.2 Code and Commit Style

We try the adhere to the PEP 8 Style Guide wherever possible. In addition to that, we use *a code formatter* to have a consistent style, even in cases where PEP 8 leaves multiple degrees of freedom. So please run your code through `black` before committing it.[1] PEP 8 also specifies a way to group imports, onto which we put the additional constraint that the imports within each group are ordered alphabetically. Once again, you don't have to keep track of this manually, but you can use isort to have imports sorted automatically. Note that *pre-commit* hooks are configured

[1] If you want to be really nice, run any file you touch through `black` before making changes, and commit the result separately from other changes.. The repository may contain wrongly formatted legacy code, and this way you commit eventually necessary style fixes separated from your actually meaningful changes, which makes the reviewers job a lot easier.

for this repository, so you can just `pip install pre-commit` followed by `pre-commit install` in the repository, and every commit will automatically be checked for style violations.

Unfortunately these tools don't catch everything, so here's a short list of things you have to keep track of manually:

- `Black` can't automatically break up overly long strings, so make use of Python's automatic string concatenation feature by e.g. converting

```
something = "A really really long string"
```

  into the equivalent:

```
something = (
    "A really really"
    " long string"
)
```

- `Black` also can't check whether you're using readable names for your variables. So please don't use abbreviations. Use readable names.
- `Black` also can't reformat your comments. So please keep in mind that PEP 8 specifies a line length of 72 for free flowing text like comments and docstrings. This also extends to the documentation in reStructuredText files.

Last but not least, commit messages are a kind of documentation, too, which should adhere to a certain style. There are quite a few documents detailing this style, but the shortest and easiest to find is probably https://commit.style. If you have 15 minutes instead of only five to spare, there's also a very good and only slightly longer article on this subject, containing references to other style guides, and also explaining why commit messages are important.

At the very least, try to only commit small, related changes. If you have to use an "and" when trying to summarize your changes, they should probably be grouped into separate commits.

6.2.3 Pull Request Guidelines

We use pull requests (PR) to integrate code changes from branches. PRs always need to be reviewed (exception proves the rule!). Therefore, ask one of the other developers for reviewing your changes. Once approved, the PR can be merged. Please delete the branch after merging.

Before requesting a review, please

1. Include passing tests (run `tox`).[2]
2. Let the workflow run in *Test mode* once from scratch to verify successful execution
3. Make sure that your changes are tested in integration with other tasks and on a complete run at least once by merging them into the continuous-integration/run-everything-over-the-weekend branch. This branch will regularly be checked out and tested on a complete workflow run on friday evening.
4. Update documentation when there's new API, functionality etc.
5. Add a note to `CHANGELOG.rst` about the changes and refer to the corresponding Github issue.
6. Add yourself to `AUTHORS.rst`.

When requesting reviews, please keep in mind it might be a significant effort to review the PR. Try to make it easier for them and keep the overall effort as low as possible. Therefore,

- asking for reviewing specific aspects helps reviewers a lot to focus on the relevant parts

[2] If you don't have all the necessary Python versions available locally you can rely on CI via GitHub actions - it will run the tests for each change you add in the pull request.
It will be slower though ...

- when multiple people are asked for a review it should be avoided that they check/test the same things. Be even more specific what you expect from someone in particular.

6.2.4 What needs to be reviewed?

Things that definitely should be checked during a review of a PR:

- *Is the code working?* The contributor should already have made sure that this is the case. Either by automated test or manual execution.
- *Is the data correct?* Verifying that newly integrated and processed data is correct is usually not possible during reviewing a PR. If it is necessary, please ask the reviewer specifically for this.
- *Do tests pass?* See automatic checks.
- *Is the documentation up-to-date?* Please check this.
- *Was* `CHANGELOG.rst` *updated accordingly?* Should be the case, please verify.
- *Is metadata complete and correct (in case of data integration)?* Please verify. In case of a pending metadata creation make sure an appropriate issue is filed.

6.3 Extending the data workflow

The egon-data workflow uses Apache Airflow which organizes the order of different processing steps and their execution.

6.3.1 How to add Python scripts

To integrate a new Python function to the egon-data workflow follow the steps listed:

1. Add your well documented script to the egon-data repository
2. Integrate functions which need to be called within the workflow to pipeline.py, which organzies and calls the different tasks within the workflow
3. Define the interdependencies between the scripts by setting the task downstream to another required task
4. The workflow can now be triggered via Apache Airflow

6.3.2 Where to save (downloaded) data?

If a task requires to retrieve some data from external sources which needs to be saved locally, please use *CWD* to store the data. This is achieved by using

```
from pathlib import Path
from urllib.request import urlretrieve

filepath = Path(".") / "filename.csv"
urlretrieve("https://url/to/file", filepath)
```

6.3.3 Add metadata

Add a metadata for every dataset you create for describing data with machine-readable information. Adhere to the OEP Metadata v1.4.1, you can follow the example to understand how the fields are used. Field are described in detail in the Open Energy Metadata Description.

You can obtain the metadata string from a table you created in SQL via

```
SELECT obj_description('<SCHEMA>.<TABLE>'::regclass);
```

Alternatively, you can write the table comment directly to a JSON file by

```
psql -h <HOST> -p <PORT> -d <DB> -U <USER> -c "\COPY (SELECT obj_description('<SCHEMA>
↪.<TABLE>'::regclass)) TO '/PATH/TO/FILE.json';"
```

For bulk export of all DB's table comments you can use this script. Please verify that your metadata string is in compliance with the OEP Metadata standard version 1.4.1 using the OMI tool (tool is shipped with eGon-data):

```
omi translate -f oep-v1.4 -t oep-v1.4 metadata_file.json
```

If your metadata string is correct, OMI puts the keys in the correct order and prints the full string (use *-o* option for export).

You may omit the fields *id* and *publicationDate* in your string as it will be automatically set at the end of the pipeline but you're required to set them to some value for a complete validation with OMI. For datasets published on the OEP *id* will be the URL which points to the table, it will follow the pattern *https://openenergy-platform.org/dataedit/view/SCHEMA/TABLE*.

For previous discussions on metadata, you may want to check PR 176.

Helpers

You can use the Metadata creator **GUI**. Fill the fields and hit *Edit JSON* to get the metadata string. Vice versa, you can paste a metadata string into this box and the fields will be filled automatically which may be helpful if you want to amend existing strings.

There are some **licence templates** provided in `egon.data.metadata` you can make use of for fields 11.4 and 12 of the Open Energy Metadata Description. Also, there's a template for the **metaMetadata** (field 16).

There are some functions to quickly generate a template for the **resource fields** (field 14.6.1 in Open Energy Metadata Description) from a SQLA table class or a DB table. This might be especially helpful if your table has plenty of columns.

- From SQLA table class: `egon.data.metadata.generate_resource_fields_from_sqla_model()`
- From database table: `egon.data.metadata.generate_resource_fields_from_db_table()`

Sources

The **sources** (field 11) are the most important parts of the metadata which need to be filled manually. You may also add references to tables in eGon-data (e.g. from an upstream task) so you don't have to list all original sources again. Make sure you include all upstream attribution requirements.

The following example uses various input datasets whose attribution must be retained:

```
"sources": [
    {
        "title": "eGo^n - Medium voltage grid districts",
        "description": (
            "Medium-voltage grid districts describe the area supplied by "
            "one MV grid. Medium-voltage grid districts are defined by one "
            "polygon that represents the supply area. Each MV grid district "
            "is connected to the HV grid via a single substation."
        ),
        "path": "https://openenergy-platform.org/dataedit/view/"
                "grid/egon_mv_grid_district", # "id" in the source dataset
        "licenses": [
            license_odbl(attribution=
                "© OpenStreetMap contributors, 2021; "
                "© Statistische Ämter des Bundes und der Länder, 2014; "
                "© Statistisches Bundesamt, Wiesbaden 2015; "
                "(Daten verändert)"
            )
        ]
    },
    # more sources...
]
```

6.3.4 Adjusting test mode data

When integrating new data or data processing scripts, make sure the *Test mode* still works correctly on a limited subset of data. In particular, if a new external data sources gets integrated make sure the data gets cut to the region of the test mode.

6.4 Documentation

eGon-data could always use more documentation, whether as part of the official eGon-data docs, in docstrings, or even in articles, blog posts or similar resources. Always keep in mind to update the documentation along with your code changes though.

The changes of the documentation in a feature branch get visible once a pull request is opened.

6.4.1 How to document Python scripts

Use docstrings to document your Python code. Note that PEP 8 also contains a section on docstrings and that there is a whole PEP dedicated to docstring conventions. Try to adhere to both of them. Additionally every Python script needs to contain a header describing the general functionality and objective and including information on copyright, license and authors.

```
""" Provide an example of the first line of a module docstring.

This is an example header describing the functionalities of a Python
script to give the user a general overview of what's happening here.
"""

__copyright__ = "Example Institut"
__license__ = "GNU Affero General Public License Version 3 (AGPL-3.0)"
```

(continues on next page)

(continued from previous page)

```
__url__ = "https://github.com/openego/eGon-data/blob/main/LICENSE"
__author__ = "github_alias1, github_alias2"
```

6.4.2 How to document SQL scripts

Please also add a similar header to your SQL scripts to give users and fellow developers an insight into your scripts and the methodologies applied. Please describe the content and objectives of the script briefly but as detailed as needed to allow other to comprehend how it works.

```
/*
This is an example header describing the functionalities of a SQL
script to give the user a general overview what's happening here

__copyright__ = "Example Institut"
__license__ = "GNU Affero General Public License Version 3 (AGPL-3.0)"
__url__ = "https://github.com/openego/eGon-data/blob/main/LICENSE"
__author__ = "github_alias1, github_alias2"
*/
```

You can build the documentation locally with (executed in the repos root directory)

```
sphinx-build -E -a docs docs/_build/
```

Eventually, you might need to install additional dependencies for building the documenmtation:

```
pip install -r docs/requirements.txt
```

6.4.3 Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel*:

```
tox -p auto
```

CHAPTER 7

Authors

- Guido Pleßmann, Ilka Cußmann, Stephan Günther, Jonathan Amme, Julian Endres, Kilian Helfenbein - https://github.com/openego/eGon-data

CHAPTER 8

Changelog

8.1 Unreleased

8.1.1 Added

- Include description of the egon-data workflow in our documentation #23
- There's now a wrapper around *subprocess.run* in *egon.data.subprocess.run*. This wrapper catches errors better and displays better error messages than Python's built-in function. Use this wrapper wenn calling other programs in Airflow tasks.
- You can now override the default database configuration using command line arguments. Look for the switches starting with `--database` in `egon-data --help`. See PR #159 for more details.
- Docker will not be used if there is already a service listening on the HOST:PORT combination configured for the database.
- You can now supply values for the command line arguments for `egon-data` using a configuration file. If the configuration file doesn't exist, it will be created by `egon-data` on it's first run. Note that the configuration file is read from and written to the directtory in which `egon-data` is started, so it's probably best to run `egon-data` in a dedicated directory. There's also the new function *egon.data.config.settings* which returns the current configuration settings. See PR #159 for more details.
- You can now use tasks which are not part of a `Dataset`, i.e. which are unversioned, as dependencies of a dataset. See **'PR #318'_** for more details.
- You can now force the tasks of a `Dataset` to be always executed by giving the version of the `Dataset` a `".dev"` suffix. See **'PR #318'_** for more details.
- OSM data import as done in open_ego #1 which was updated to the latest long-term data set of the 2021-01-01 in #223 <https://github.com/openego/eGon-data/issues/223>'_
- Verwaltungsgebiete data import (vg250) more or less done as in open_ego #3
- Zensus population data import #2
- Zensus data import for households, apartments and buildings #91

- DemandRegio data import for annual electricity demands #5
- Download cleaned open-MaStR data from Zenodo #14
- NEP 2021 input data import #45
- Option for running workflow in test mode #112
- Abstraction of hvmv and ehv substations #9
- Filter zensus being inside Germany and assign population to municipalities #7
- RE potential areas data import #124
- Heat demand data import #101
- Demographic change integration #47
- Creation of voronoi polygons for hvmv and ehv substations #9
- Add hydro and biomass power plants eGon2035 #127
- Creation of the ehv/hv grid model with osmTGmod, see issue #4 and PR #164
- Identification of medium-voltage grid districts #10
- Distribute electrical demands of households to zensus cells #181
- Distribute electrical demands of cts to zensus cells #210
- Include industrial sites' download, import and merge #117
- Integrate scenario table with parameters for each sector #177
- The volume of the docker container for the PostgreSQL database is saved in the project directory under *docker/database-data*. The current user (*$USER*) is owner of the volume. Containers created prior to this change will fail when using the changed code. The container needs to be re-created. #228
- Extract landuse areas from OSM #214
- Integrate weather data and renewable feedin timeseries #19
- Create and import district heating areas #162
- Integrate electrical load time series for cts sector #109
- Assign voltage level and bus_id to power plants #15
- Integrate solar rooftop for etrago tables #255
- Integrate gas bus and link tables #198
- Integrate data bundle #272
- Add household electricity demand time series, mapping of demand profiles to census cells and aggregated household electricity demand time series at MV grid district level #256
- Integrate power-to-gas installation potential links #293
- Integrate distribution of wind onshore and pv ground mounted generation #146
- Integrate dynamic line rating potentials #72
- Integrate gas voronoi polygons #308
- Integrate supply strategies for individual and district heating #232
- Integrate gas production #321
- Integrate industrial time series creation #237

- Merge electrical loads per bus and export to etrago tables #328
- Insert industial gas demand #321
- Integrate existing CHP and extdended CHP > 10MW_el #266
- Add random seed to CLI parameters #351
- Extend zensus by a combined table with all cells where there's either building, apartment or population data #359
- Include allocation of pumped hydro units #332
- Add example metadata for OSM, VG250 and Zensus VG250. Add metadata templates for licences, context and some helper functions. Extend docs on how to create metadata for tables. #139
- Integrate DSM potentials for CTS and industry #259
- Assign weather cell id to weather dependant power plants #330
- Distribute wind offshore capacities #329
- Add CH4 storages #405
- Include allocation of conventional (non CHP) power plants #392
- Fill egon-etrago-generators table #485
- Include time-dependent coefficient of performance for heat pumps #532
- Limit number of parallel processes per task #265
- Include biomass CHP plants to eTraGo tables #498
- Include Pypsa default values in table creation #544
- Include PHS in eTraGo tables #333
- Include feedin time series for wind offshore #531
- Include carrier names in eTraGo table #551
- Include hydrogen infrastructure for eGon2035 scenario #474
- Include downloaded pypsa-eur-sec results #138
- Create heat buses for eGon100RE scenario #582
- Filter for DE in gas infrastructure deletion at beginning of respective tasks #567
- Insert open cycle gas turbines into eTraGo tables #548
- Preprocess buildings and amenities for LV grids #262
- Assign household profiles to OSM buildings #435
- Add link to meta creator to docs #599
- Add extendable batteries and heat stores #566
- Add efficiency, capital_cost and marginal_cost to gas related data in etrago tables #596
- Add wind onshore farms for the eGon100RE scenario #690
- The shared memory under *"/dev/shm"* is now shared between host and container. This was done because Docker has a rather tiny default for the size of *"/dev/shm"* which caused random problems. Guessing what size is correct is also not a good idea, so sharing between host and container seems like the best option. This restricts using *egon-data* with docker to Linux and MacOS, if the latter has *"/dev/shm"* but seems like the best course of action for now. Done via PR #703 and hopefully prevents issues #702 and #267 from ever occurring again.

- Provide wrapper to catch DB unique violation #514
- Add electric scenario parameters for eGon100RE #699
- Introduce Sanity checks for eGon2035 #382
- Add motorized individual travel #553

8.1.2 Changed

- Adapt structure of the documentation to project specific requirements #20
- Switch from Travis to GitHub actions for CI jobs #92
- Rename columns to id and zensus_population_id in zensus tables #140
- Revise docs CONTRIBUTING section and in particular PR guidelines #88 and #145
- Drop support for Python3.6 #148
- Improve selection of zensus data in test mode #151
- Delete tables before re-creation and data insertation #166
- Adjust residential heat demand in unpopulated zenus cells #167
- Introduce mapping between VG250 municipalities and census cells #165
- Delete tables if they exist before re-creation and data insertation #166
- Add gdal to pre-requisites #185
- Update task zensus-inside-germany #196
- Update installation of demandregio's disaggregator #202
- Update etrago tables #243 <https://github.com/openego/eGon-data/issues/243>'_and '#285
- Migrate VG250 to datasets #283
- Allow configuring the airflow port #281
- Migrate mastr, mv_grid_districts and re_potential_areas to datasets #297
- Migrate industrial sites to datasets #237
- Rename etrago tables from e.g. egon_pf_hv_bus to egon_etrago bus etc. #334
- Move functions used by multiple datasets #323
- Migrate scenario tables to datasets #309
- Migrate weather data and power plants to datasets #314
- Create and fill table for CTS electricity demand per bus #326
- Migrate osmTGmod to datasets #305
- Filter osm landuse areas, rename industrial sites tables and update load curve function #378
- Remove version columns from eTraGo tables and related code #384
- Remove country column from scenario capacities table #391
- Update version of zenodo download #397
- Rename columns gid to id #169
- Remove upper version limit of pandas #383

- Use random seed from CLI parameters for CHP and society prognosis functions #351
- Changed demand.egon_schmidt_industrial_sites - table and merged table (industrial_sites) #423
- Replace 'gas' carrier with 'CH4' and 'H2' carriers #436
- Adjust file path for industrial sites import #397
- Rename columns subst_id to bus_id #335
- Apply black and isort for all python scripts #463
- Update deposit id for zenodo download #397
- Add to etrago.setug.py the busmap table #484
- Migrate dlr script to datasets #508
- Migrate loadarea scripts to datasets #525
- Migrate plot.py to dataset of district heating areas #527
- Migrate substation scripts to datasets #304
- Update deposit_id for zenodo download #540
- Add household demand profiles to etrago table #381
- Migrate zensus scripts to datasets #422
- Add information on plz, city and federal state to data on mastr without chp #425
- Assign residential heat demands to osm buildings #557
- Add foreign gas buses and adjust cross bording pipelines #545
- Integrate fuel and CO2 costs for eGon2035 to scenario parameters #549
- Aggregate generators and stores for CH4

  #629
- Fill missing household data for populated cells #431
- Fix RE potential areas outside of Germany by updating the dataset. Import files from data bundle. #592 #595
- Add DC lines from Germany to Sweden and Denmark #611
- H2 demand is met from the H2_grid buses. In Addtion, it can be met from the H2_saltcavern buses if a proximity criterion is fulfilled #620
- Create H2 pipeline infrastructure for eGon100RE #638
- Change refinement method for households types #651
- H2 feed in links are changed to non extendable #653
- Remove the '_fixed' suffix #628
- Fill table demand.egon_demandregio_zensus_electricity after profile allocation #620
- Change method of building assignment #663
- Create new OSM residential building table #587
- Move python-operators out of pipeline #644
- Add annualized investment costs to eTraGo tables #672
- Improve modelling of NG and biomethane production #678

- Unify carrier names for both scenarios #575
- Add automatic filtering of gas data: Pipelines of length zero and gas buses isolated of the grid are deleted. #590
- Add gas data in neighbouring countries #727
- Aggregate DSM components per substation #661
- Aggregate NUTS3 industrial loads for CH4 and H2 #452
- Update OSM dataset from 2021-02-02 to 2022-01-01 #486
- Update deposit id to access v0.6 of the zenodo repository #627
- Include electricity storages for eGon100RE scenario #581
- Update deposit id to access v0.7 of the zenodo repository #736
- Update hh electricity profiles

  #735
- Improve CH4 stores and productions aggregation by removing dedicated task #PR775
- Add CH4 stores in Germany for eGon100RE #779
- Assigment of H2 and CH4 capacitites for pipelines in eGon100RE #686
- Update deposit id to access v0.8 of the zenodo repository #760
- Add primary key to table openstreetmap.osm_ways_with_segments #787

8.1.3 Bug Fixes

- Some dependencies have their upper versions restricted now. This is mostly due to us not yet supporting Airflow 2.0 which means that it will no longer work with certain packages, but we also won't get and upper version limit for those from Airflow because version 1.X is unlikely to to get an update. So we had to make some implicit dependencies explicit in order to give them them upper version limits. Done via PR #692 in order to fix issues #343, #556, #641 and #669.
- Heat demand data import #157
- Substation sequence #171
- Adjust names of demandregios nuts3 regions according to nuts version 2016 #201
- Delete zensus buildings, apartments and households in unpopulated cells #202
- Fix input table of electrical-demands-zensus #217
- Import heat demand raster files successively to fix import for dataset==Everything #204
- Replace wrong table name in SQL function used in substation extraction #236
- Fix osmtgmod for osm data from 2021 by updating substation in Garenfeld and set srid #241 #258
- Adjust format of voltage levels in hvmv substation #248
- Change order of osmtgmod tasks #253
- Fix missing municipalities #279
- Fix import of hydro power plants #270
- Fix path to osm-file for osmtgmod_osm_import #258
- Fix conflicting docker containers by setting a project name #289

- Update task insert-nep-data for pandas version 1.3.0 #322
- Fix versioning conflict with mv_grid_districts #340
- Set current working directory as java's temp dir when executing osmosis #344
- Fix border gas voronoi polygons which had no bus_id #362
- Add dependency from WeatherData to Vg250 #387
- Fix unnecessary columns in normal mode for inserting the gas production #387
- Add xlrd and openpyxl to installation setup #400
- Store files of OSM, zensus and VG250 in working dir #341
- Remove hard-coded slashes in file paths to ensure Windows compatibility #398
- Add missing dependency in pipeline.py #412
- Add prefix egon to MV grid district tables #349
- Bump MV grid district version no #432
- Add curl to prerequisites in the docs #440
- Replace NAN by 0 to avoid empty p_set column in DB #414
- Exchange bus 0 and bus 1 in Power-to-H2 links #458
- Fix missing cts demands for eGon2035 #511
- Add *data_bundle* to *industrial_sites* task dependencies #468
- Lift *geopandas* minimum requirement to *0.10.0* #504
- Use inbuilt *datetime* package instead of *pandas.datetime* #516
- Add missing 'sign' for CH4 and H2 loads #516
- Delete only AC loads for eTraGo in electricity_demand_etrago #535
- Filter target values by scenario name #570
- Reduce number of timesteps of hh electricity demand profiles to 8760 #593
- Fix assignemnt of heat demand profiles at German borders #585
- Change source for H2 steel tank storage to Danish Energy Agency #605
- Change carrier name from 'pv' to 'solar' in eTraGo_generators #617
- Assign "carrier" to transmission lines with no value in grid.egon_etrago_line #625
- Fix deleting from eTraGo tables #613
- Fix positions of the foreign gas buses #618
- Create and fill transfer_busses table in substation-dataset #610
- H2 steel tanks are removed again from saltcavern storage #621
- Timeseries not deleted from grid.etrago_generator_timeseries #645
- Fix function to get scaled hh profiles #674
- Change order of pypsa-eur-sec and scenario-capacities #589
- Fix gas storages capacities #676
- Distribute rural heat supply to residetntial and service demands #679

- Fix time series creation for pv rooftop #688
- Fix extraction of buildings without amenities #693
- Assign DLR capacities to every transmission line #683
- Fix solar ground mounted total installed capacity #695
- Fix twisted number error residential demand #704
- Clean up *"pipeline.py"* #562
- Assign timeseries data to crossborder generators ego2035 #724
- Add missing dataset dependencies in "pipeline.py" #725
- Fix assignemnt of impedances (x) to etrago tables #710
- Fix country_code attribution of two gas buses #744
- Fix voronoi assignemnt for enclaves #734
- Set lengths of non-pipeline links to 0 #741
- Change table name from `boundaries.saltstructures_inspee` to `boundaries.inspee_saltstructures` #746
- Add missing marginal costs for conventional generators in Germany #722
- Fix carrier name for solar ground mounted in scenario parameters #752
- Create rural_heat buses only for mv grid districts with heat load #708
- Solve problem while creating generators series data egon2035 #758
- Correct wrong carrier name when assigning marginal costs #766
- Use db.next_etrago_id in dsm and pv_rooftop dataset #748
- Add missing dependency to heat_etrago #771
- Fix distribution of resistive heaters in district heating grids #783
- Fix missing reservoir and run_of_river power plants in eTraGo tables, Modify fill_etrago_gen to also group generators from eGon100RE, Use db.next_etrago_id in fill_etrago_gen #798 #776

CHAPTER 9

Reference

9.1 egon.data

9.2 egon.data.airflow package

9.2.1 Subpackages

egon.data.airflow.dags namespace

Submodules

egon.data.airflow.dags.pipeline module

9.2.2 Submodules

9.2.3 egon.data.airflow.tasks module

9.3 egon.data.airflow.dags.pipeline module

9.4 egon.data.airflow.tasks module

9.5 egon.data.cli module

Module that contains the command line app.

Why does this file exist, and why not put this in __main__?

You might be tempted to import things from __main__ later, but that will cause problems: the code will get executed twice:

- When you run *python -megon.data* python will execute `__main__.py` as a script. That means there won't be any `egon.data.__main__` in `sys.modules`.
- When you import __main__ it will get executed again (as a module) because there's no `egon.data.__main__` in `sys.modules`.

Also see (1) from http://click.pocoo.org/5/setuptools/#setuptools-integration

`egon.data.cli.`**`main`**`()`

9.6 egon.data.importing.openstreetmap package

9.6.1 Module contents

9.7 egon.data.processing namespace

9.7.1 Submodules

9.7.2 egon.data.processing.openstreetmap module

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

Index

E

M