

SOFTWARE METAPAPER

esy-osmfilter – A Python Library to Efficiently Extract OpenStreetMap Data

Adam Pluta and Ontje Lünsdorf

DLR Institute of Networked Energy Systems – Energy Systems Analysis, DE

Corresponding author: Adam Pluta (adam.pluta@dlr.de)

OpenStreetMap is the largest freely accessible geographic database of the world. The necessary processing steps to extract information from this database, namely reading, converting and filtering, can be very consuming in terms of computational time and disk space.

esy-osmfilter is a Python library designed to read and filter OpenStreetMap data under optimization of disc space and computational time. It uses parallelized prefiltering for the OSM pbf-files data in order to quickly reduce the original data size. It can store the prefiltered data to the hard drive. In the main filtering process, these prefiltered data can be reused repeatedly to identify different items with the help of more specialized main filters. At the end, the output can be exported to the GeoJSON format.

Keywords: OSM; OpenStreetMap; Python; GeoJSON; PBF; Protocol buffers; GeoJSON; geo

Funding statement: This work was funded as part of DLR Institute for Networked Energy Systems project *SciGRID_gas* by the German Federal Ministry for Economic Affairs and Energy (BMWi) within the funding of the 6. Energieforschungsprogramm der Bundesregierung. Funding Code: 03ET4063.

(1) Overview

Introduction

OpenStreetMap (OSM) is a powerful and freely accessible database of geo-referenced objects with continuously increasing data coverage and data quality. The open access policy of OSM contributes to many research areas. In the field of energy system modelling, for example, it has been used successfully in the creation of power grid models [1] or for the optimisation of flexibility options in urban areas [2, 3]. In general, within OSM data, items can be identified by relevant key-value pairs, called *tags*. However, the correct identification of relevant tags can be an iterative process. Often, it will require the user to repeatedly filter the data and adapt the used filters. It can be very inefficient to filter the whole OSM data files repeatedly as they can be quite large (pbf-file Europe 2019: 20.6 GB). For this reason, it is of advantage to optimize these processes, especially if they are applied to big data. *esy-osmfilter* is a direct outcome of the *SciGRID_gas* project.¹ The library can read OSM **pbf-files**, which are downloadable from *geofabrik*.²

Within the *SciGRID_gas* project, *esy-osmfilter* has been used to extract European gas transport pipelines from OSM and further to identify other relevant components (e.g. gas compressor stations, pipeline marker, gas storages) of the European gas transport network.

Outside of Python, such tasks could also be realised with popular tool as IMPOSM³ or OSMOSIS.⁴ The first is

a PostgreSQL/PostGIS tool, the second is a JAVA open source command-line software. The authors of this work have only personal experience with the tool OSMOSIS. From their point of view, the OSMOSIS syntax is not very user friendly if used for sophisticated filter operations.

OSM element types

The performance of filtering OSM data scales not only with the size of the underlying pbf-file and the complexity of the defined filter rules but also with the count and type of each identified OSM element. The three standard OSM element types, are explained in details on the OpenStreetMap webpage.⁵ However, we give a brief summary. Each OSM item contains a unique ID and optionally meta information in the form of a list of key-value pairs.

- **OSM Nodes** are geo-referenced points on the surface of earth. They are used to represent smaller standalone features, for instance traffics signals or benches. They are also referenced in OSM Ways, to define the shape of a way.
- **OSM Ways** can represent linear features as rivers and roads or boundaries of areas such as buildings or forests.
- **OSM Relations** represents relationships between nodes, ways and/or other relations. They can represent a bus route or complex areas with holes.

Obviously, the identification of ways requires a second filter loop for the referenced nodes. In the case of relations, this even requires recursive looping over the referenced nodes, relation members in order to find sub-sequential OSM elements.

Performance

The performance of *esy-osmfilter* has been compared to the performance of OSMOSIS on an Intel(R) Xeon(R) CPU E5-2630 v2 2.60GHz machine, which has 24 CPUs. For the comparison we have chosen to filter all ways and referenced nodes of the tags “*railway:tram*” and “*railway:tram_stop*” from three different sized pbf-files [2.2 MB, 59 MB, 707 MB] on a linux machine.

The performance of *esy-osmfilter* [0.6s, 8.9s, 98.45s] has been consistently about four times as fast as OSMOSIS [2.8s, 36.0s, 416.5s]. Both performances are depicted in **Figure 1**.

However, we do not state that our software outperforms established software in every case, as this statement would require more testing. Nevertheless, we expect that the real time advantage for the user can be much larger if he uses the *esy-osmfilter* structure appropriately. The user could use a very permeable prefilter once and than reuse the stored prefiltered data in the data dictionary to customize his black and whitefilters. This will significantly reduce the computational cost for each reuse.

Implementation

This Python library has been tested on Unix and Windows. On some older windows machines we noticed problems with the python multiprocessing library. As workaround, the user can switch of the multiprocessing, as described in the documentation.

Architecture

OSM objects are stored in OSM pbf-files, which serve as a input to *esy-osmfilter*. The second input consists of the three customizable filters: a) **prefilter**, b) **whitefilter**, and c) **blackfilter**. They are described in more detail in the documentation.⁶ In **Figure 2** we demonstrate the workflow of the *esy-osmfilter*, which consist of a **read phase**, a **prefilter phase** and a **mainfilter phase**.

In the **read phase**, the internal blocks in the pbf-file are read with the help of the *esy-osm-pbf* library.

In the **prefilter phase**, the *esy-osmfilter* takes advantage of the pbf-file block structure. It reduces the computational time for the prefiltering by parallelizing this process. This is done with the help of the standard python *multiprocessing* module. In this phase, the user can define a customizable prefilter with complex filter rules for all OSM element types, namely *nodes*, *ways*, and *relations*. The prefilter searches for the all OSM items which fulfill the filter rules. Additionally, it also searches for the references and relation members of these items, which are equally OSM items by themselves, and stores all items in the *Data* dictionary. Here, we give a brief overview of the stored items:

- **nodes**
- **ways** + way nodes
- **relations** + relation nodes
 - + member ways
 - + way nodes

During the **mainfilter phase** our library applies the user defined *whitefilters* and *blackfilters* to select specific items from the *Data* dictionary. These items are stored in the *Elements* dictionary, which can subsequently be written to a *pickle*-file for quick reuse or to a human-readable *JSON*-file.

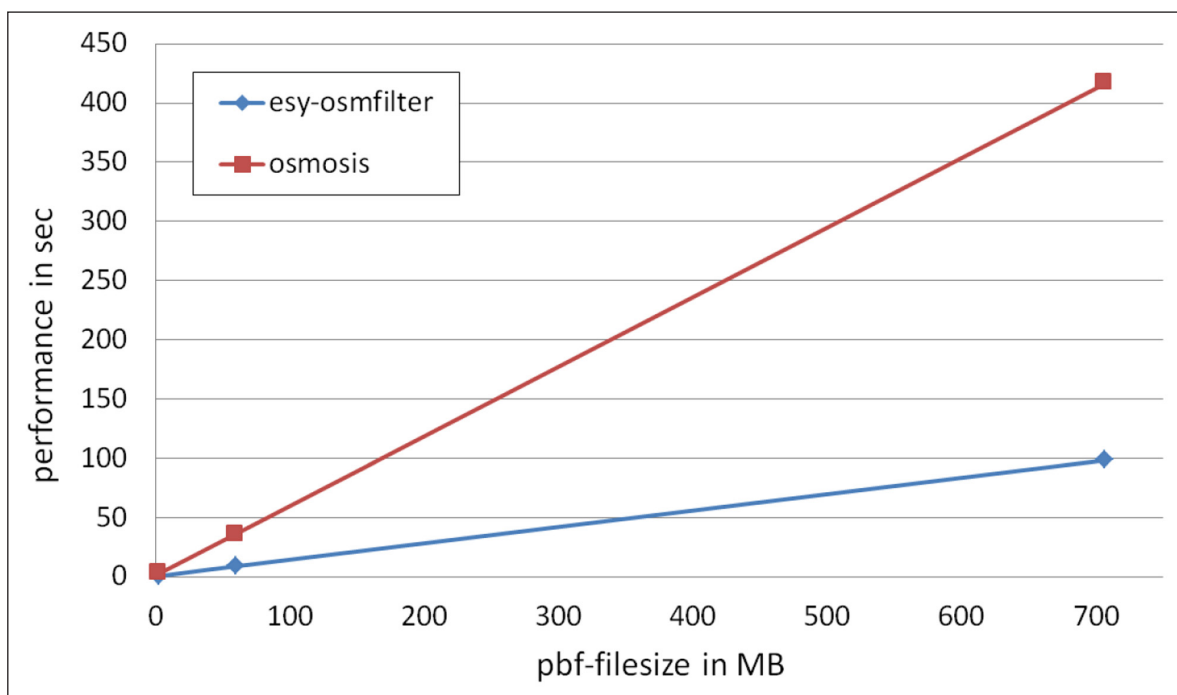


Figure 1: Performance comparison of *esy-osmfilter* and OSMOSIS for the same filter operation in 3 different-sized pbf-files.

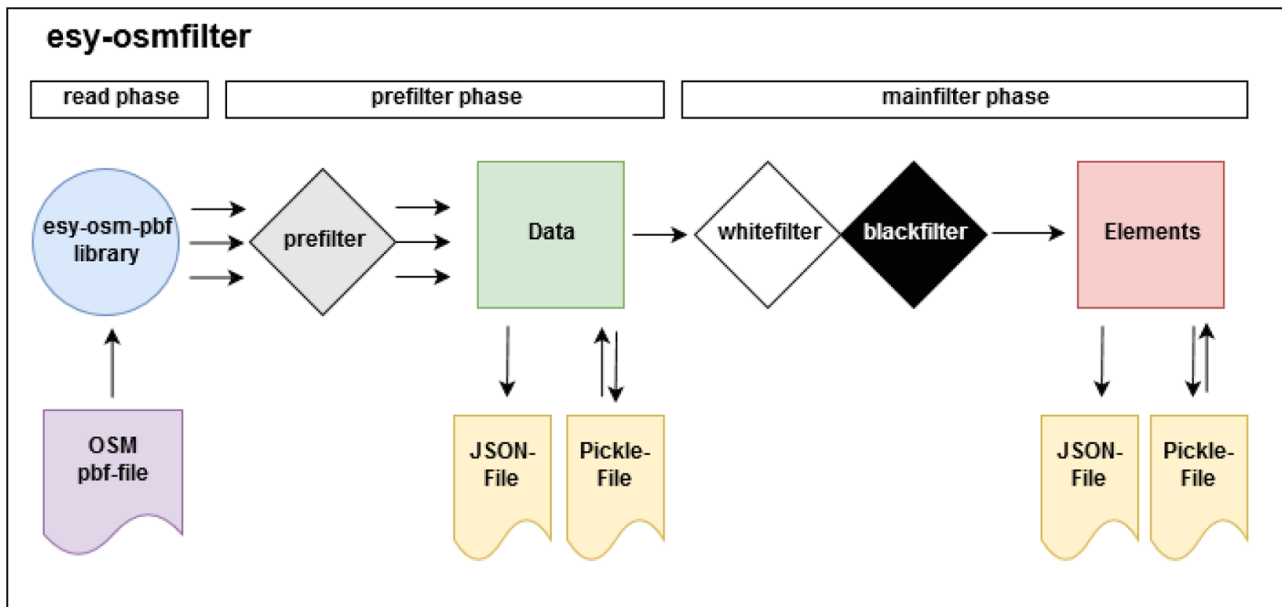


Figure 2: Diagram of the esy-osmfilter architecture.

It should be emphasised again that the *Elements* dictionary contains only those items from the *Data* dictionary, which directly fulfil the main filter rules. However, all referenced nodes and relation members can still be accessed by their IDs from the *Data* dictionary.

Export to GeoJSON

GeoJSON is an open format for geographic data. It is compatible with geographic information system (GIS) applications or the very popular python shapely library.⁷ Further it is also easily convertible to other popular data formats (e.g. shapefiles). *esy-osmfilter* provides the function `export_geojson` which takes both, the *Data* and the *Elements* dictionary, as input. Therefrom, it constructs GeoJSON **Line** or **Point** objects, which are finally stored in a GeoJSON file. This procedure is demonstrated in the already mentioned `sample.py` file. It has to be noted, that the conversion with `export_geojson` to other GeoJSON object types as **Polygons**, **MultiPoints**, **MultiLineStrings** and **MultiPolygons** is currently not implemented. However, this might change with future updates.

Visualisation

The visualisation of the final results is beyond the scope of *esy-osmfilter*. However, the user can drag and drop the resulting GeoJSON files on the map at <https://geojson.io> to visualise the results in no time.

Installation

To install on Linux run 'sudo python setup.py install'.

Usage

The usage of this tool is well documented in the README.md file provided in the GitLab repository mentioned below. The tool is also accompanied by an executable sample file `sample.py`, which guides the user through the download of pbf-files, the usage of the different filters and the conversion of the filter results towards the GeoJSON

format. We strongly recommend new users to download this file from the repository and simply customizing it to their own needs. Please find further information on this topic in the *esy-osmfilter* online documentation.⁸

Quality control

The filter results of *esy-osmfilter* for European gas pipelines in June 2020 are displayed in **Figure 3**. We compare them visually to the results produced by the IMOSM extraction tool, which are displayed in **Figure 4** and taken from `openinframap`.⁹ Obviously `inframap` has intentionally removed short OSM ways from their map for a better visibility. However, this might even result in the loss of some longer pipelines, as some are internally constructed from very short OSM ways. Besides that, both gas pipeline networks appear very similar.

To make further comparisons available, we have also used *esy-osmfilter* together with historical European pbf-files from 2014 to 2019 to create a video of the annual gas pipeline data within the OSM database.¹⁰

In order to confirm that our application delivers the same filter results as established tools, we have also used OSMOSIS to reproduce the results from `sample.py`, described in the usage section of the documentation.¹¹ This comparison is based on finding all pipelines within the accompanied pbf-file (`liechtenstein-191101.osm.pbf`). In both cases we have only identified the same two drain pipelines named "Wäschgräble" and "Wäschgräbli".

Developer Tests

Developer-tests have been implemented under `esy-osmfilter/test`, which can confirm the integrity of *esy-osmfilter*. They can be executed manually with the execution of `pytest` module from the main program folder. In addition, comparable tests have been implemented in the README.md file which can be executed with python module `doctest`. They are automatically executed with each push to GitLab.

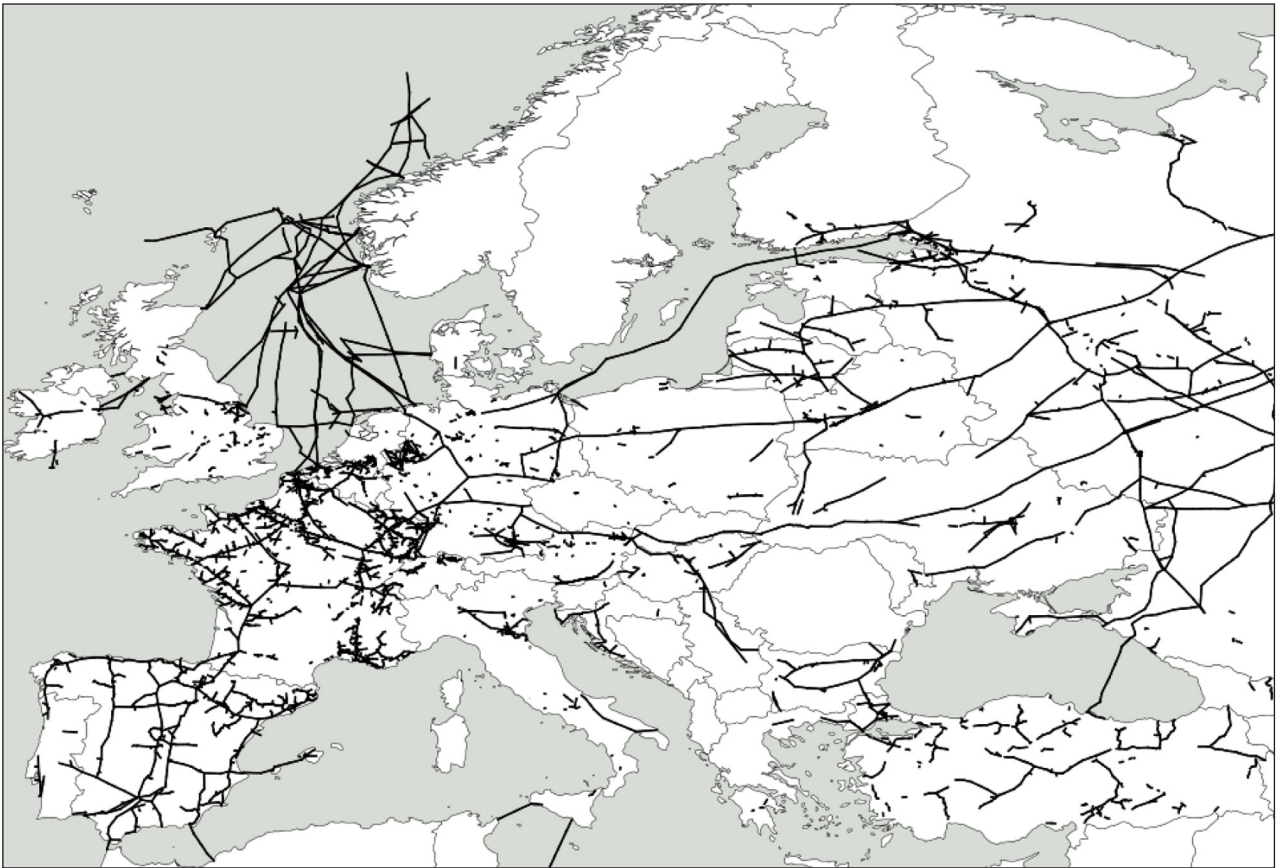


Figure 3: European gas transport pipelines extracted from OSM with *esy-osmfilter* in June 2020.

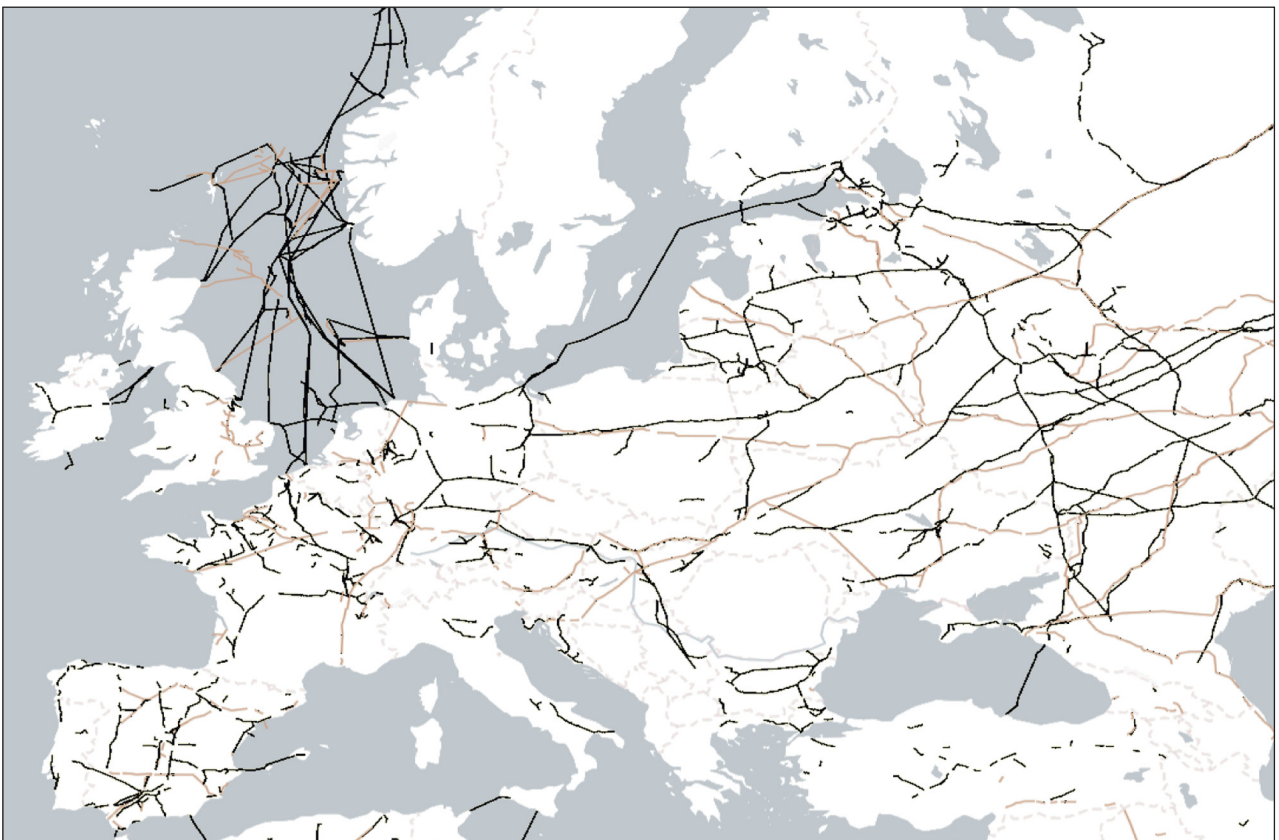


Figure 4: European gas (black) and oil (orange) transport pipelines from openinframap extracted from OSM with IMPOSM in June 2020. Image colors have been enhanced.

(2) Availability

Operating system

OS and Windows

Programming language

Python > 3.6

Dependencies

esy-osm-pbf
protobuf

Software location

Name: Zenodo

Persistent identifier: <https://doi.org/10.5281/zenodo.3874597>

Licence: GNU GPL v3.0

Date published: 06/03/2020

Version: 1.0.7

Code repository

Name: GitLab

Identifier: <https://gitlab.com/dlr-ve-esy/esy-osmfilter>

Licence: GNU GPL v3.0

Date published: 02/11/2020

Version: 1.0.7

Language

English

(3) Reuse potential

This software can be used for most purposes, which involve the extraction of geographic infrastructure from the OSM database. This can be realized by the adaption of the customizable filters to the relevant OSM tags. In the reference section, we give some examples for the potential reuse potential of our application. Also, an introduction to OpenStreetMap in geographic information science can be found in the book of Arsanjani et al. [4].

Limitations

We have noticed the current two limitations:

- **Bounding boxes/bounding polygon files**

esy-osmfilter does not allow for regional filtering via bounding boxes or bounding polygon files. However, we do not regard this as a real limitation, as pbf-files are available on different regional levels. Due to the fast performance of *esy-osmfilter*, users can filter the results for an upper level pbf-file and afterwards filter the GeoJSON files for the desired specific region, which is a very easy task.

Alternatively, one can use OSMOSIS to prefilter pbf-files to a desired sub-region and write the filter results back to a pbf-file, which is a one line command.

- **Export of Relations**

The current version of the *export_gejson* function can not export entire relations to complex GeoJSON

objects, as mentioned earlier. This functionality has not been implemented as it was needed within the scope of the SciGRID_gas project. We believe that such a functionality in the *export_gejson* could be very useful. Therefore, we are considering the implementation of such a feature with a future update.

Support

Support is currently provided via GitLab issues. You can also contact the developers via email.

Notes

¹ <https://www.gas.scigrid.de/>.

² <https://download.geofabrik.de/>.

³ <https://imposm.org/>.

⁴ <https://learnosm.org/en/osm-data/osmosis/>.

⁵ <https://wiki.openstreetmap.org/wiki/Elements>.

⁶ <https://dlr-ve-esy.gitlab.io/esy-osmfilter/>.

⁷ <https://pypi.org/project/Shapely/>.

⁸ <https://gitlab.com/dlr-ve-esy/esy-osmfilter>.

⁹ <https://openinframap.org/>.

¹⁰ https://www.gas.scigrid.de/pdfs/GasPipelines_history_new.mp4.

¹¹ <https://dlr-ve-esy.gitlab.io/esy-osmfilter/usage.html>.

Acknowledgements

These authors like to acknowledge the contribution of Jan Diettrich and Jan Dasenbrock to the overall project.

Competing Interests

The authors have no competing interests to declare.


References

1. **Medjroubi, W, Philipp Müller, U, Scharf, M, Matke, C and Kleinhans, D** 2017 Open data in power grid modelling: New approaches towards transparent grid models. *Energy Reports*, 3: 14–21. DOI: <https://doi.org/10.1016/j.egy.2016.12.001>
2. **Alhamwi, A, Medjroubi, W, Vogt, T and Agert, C** Apr 2017 GIS-based urban energy systems models and tools: Introducing a model for the optimization of flexibilisation technologies in urban areas. *Applied Energy*, 191: 1–9. DOI: <https://doi.org/10.1016/j.apenergy.2017.01.048>
3. **Alhamwi, A, Medjroubi, W, Vogt, T and Agert, C** 2017 Openstreetmap data in modelling the urban energy infrastructure: a first assessment and analysis. In *Proceedings of the 9th International Conference on Applied Energy*, 142: 1968–1976. Elsevier. DOI: <https://doi.org/10.1016/j.egypro.2017.12.397>
4. **Arsanjani, J J, Zipf, A, Mooney, P and Helbich, M** 2015 An introduction to openstreetmap in geographic information science: Experiences, research, and applications. In *OpenStreetMap in GIScience*, 1–15. Springer. DOI: https://doi.org/10.1007/978-3-319-14280-7_1

How to cite this article: Pluta, A and Lünsdorf, O 2020 esy-osmfilter – A Python Library to Efficiently Extract OpenStreetMap Data. *Journal of Open Research Software*, 8: 19. DOI: <https://doi.org/10.5334/jors.317>

Submitted: 27 January 2020 **Accepted:** 22 June 2020 **Published:** 01 September 2020

Copyright: © 2020 The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License (CC-BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. See <http://creativecommons.org/licenses/by/4.0/>.

 *Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press.

OPEN ACCESS 