Creating Energy System Data Sets from OpenStreetMap

Adam Pluta, Ontje Lünsdorf

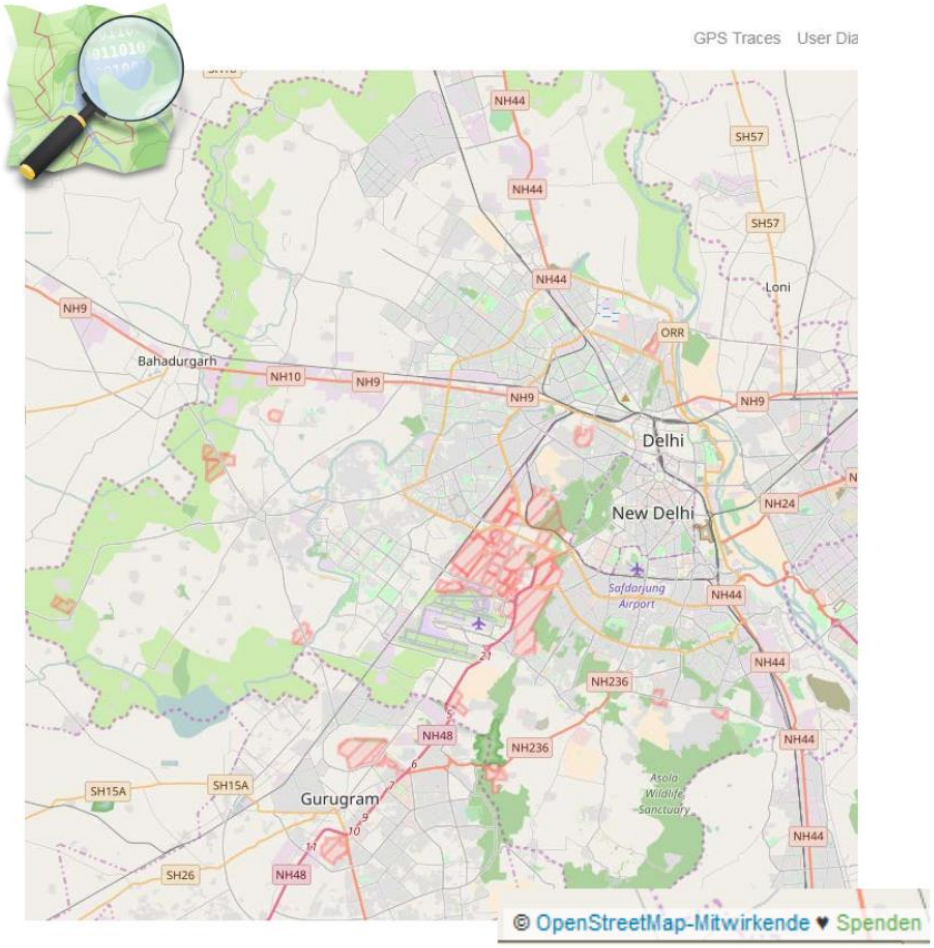
15-17.01.2020

Openmod Berlin2020

DLR-Institut für Vernetzte Energiesysteme

What is OpenStreetMap?

- OpenStreetMap is the largest geo-reference open source database of the world
- The open database license (ODbL) allows free usage, changements and distribution under the same license.
- Open source tools for download and processing (Osmium, Osmosis, imposm, osm2pgsql, mkgmap, overpass-turbo,etc.)



Geofabrik

http://download.geofabrik.de/

OpenStreetMap Data Extracts

The OpenStreetMap data files provided on this server do **not** contain the user names, user IDs and changeset IDs of the OSM objects. These metadata fields contain personal information about the OpenStreetMap contributors and are subject to data protection regulations in the European Union. Please note that these regulations apply even to processing that happens outside the European Union because some OpenStreetMap contributors live in the European Union.
Extracts with full metadata are available to OpenStreetMap contributors only.

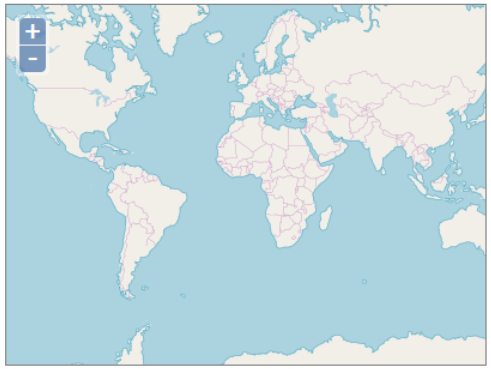
Welcome to Geofabrik's free download server. This server has data extracts from the OpenStreetMap project which are normally updated every day. Select your continent and then your country of interest from the list below. (If you have been directed to this page from elsewhere and are not familiar with OpenStreetMap, we highly recommend that you read up on OSM before you use the data.) This open data download service is offered free of charge by Geofabrik GmbH.

Willkommen auf dem Geofabrik-Downloadserver. Hier gibt es Daten-Auszüge aus dem OpenStreetMap-Projekt, die normalerweise täglich aktualisiert werden. Wählen Sie aus dem Verzeichnis unten den Kontient und ggf. das Land, für die Sie Daten benötigen. (Wenn Sie von anderswo auf dieser Seite gelandet sind und von OpenStreetMap nichts wissen, dann ist es empfehlenswert, sich mit dem Projekt vertraut zu machen, bevor Sie mit den Daten arbeiten.) Diese Downloads werden von der Geofabrik GmbH kostenlos angeboten.

Click on the region name to see the overview page for that region, or select one of the file extension links for quick access.

Sub Region	Quick Links			
	.osm.pbf		.shp.zip	.osm.bz2
Africa	[.osm.pbf]	(2.9 GB)	✖	[.osm.bz2]
Antarctica	[.osm.pbf]	(29.0 MB)	[.shp.zip]	[.osm.bz2]
Asia	[.osm.pbf]	(6.9 GB)	✖	[.osm.bz2]
Australia and Oceania	[.osm.pbf]	(645 MB)	✖	[.osm.bz2]
Central America	[.osm.pbf]	(346 MB)	✖	[.osm.bz2]
Europe	[.osm.pbf]	(19.6 GB)	✖	[.osm.bz2]
North America	[.osm.pbf]	(8.4 GB)	✖	[.osm.bz2]
South America	[.osm.pbf]	(1.5 GB)	✖	[.osm.bz2]

Technical details about this download service.



Not what you were looking for? Geofabrik is a consulting and software development firm based in Karlsruhe, Germany specializing in OpenStreetMap services. We're happy to help you with data preparation, processing, server setup and the like. Check out our web site and contact us if we can be of service.

Nicht das Richtige dabei? Die Geofabrik ist ein auf OpenStreetMap spezialisiertes Beratungs- und Softwareentwicklungsunternehmen in Karlsruhe. Gern helfen wir Ihnen bei der Datenaufbereitung, Datenkonvertierung, Serverinstallation und ähnlichen Aufgaben. Besuchen Sie unsere Webseite und sprechen Sie mit uns, wenn wir Ihnen helfen können.

Data in OSM

OSM data:

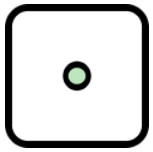
- 20 GB for Europa
- Accurate georeferencing
- Nodes, Ways, Relations
- Metainformation in key-value-pairs (Tags)

Example: gas data:

- „man_made = pipeline“
- „substance = gas“
- „pipeline = marker“
- „man_made = storage_tank“
- „man_made = gasometer“

OSM object types:

Nodes:



Ways:

Relations:

Objects in OSM


```
{ "id":553081211,
        "tags":{
                "barrier "      : "fence",
                "industrial "  : "gas",
                "landuse "    : "industrial",
                "name "        : "Gas Pressure Reducing
                Station",              "operator "   :
                "National Grid plc",
               "ref"            : "Tilbury Thames North
       2731
                },
         "refs":{
                    1871614593,
                    1871614595,
                     …
                    },
```


Identification of Relevant Tags

Openstreetmap Wiki

Tag:man_made=pipeline – Weitere Sprachen

čeština · Deutsch · English · polski · português · русский · 日本語 ·

Andere Sprachen · Übersetzen

Auszeichnung von **Rohrleitungen** für Wasser, Öl, Gas, Abwasser, Fernwärme usw. Die Rohrleitung kann sowohl über der Erde, unter der Erde, als auch unter Wasser verlegt sein. Zum größten Teil werden Pipelines zum Transport von Erdöl, Erdgas, Wasser und Abwasser (Kanal) verwendet.

Inhaltsverzeichnis [Verbergen]

1 Wie mappen?
2 Tags, die in diesem Zusammenhang verwendet werden
3 Rohrleitungen auf dem offenen Meer
4 Sehr lange Rohrleitungen
5 Andere Attribute die im Zusammenhang gebraucht werden
6 Beispiele
7 Karte
8 Siehe auch

Wie mappen?

Zeichne einen Linie und füge `man_made=pipeline` hinzu.

Tags, die in diesem Zusammenhang verwendet werden

Die folgenden Tags sind auch in Kombination mit `bridge=yes` für eine Rohrbrücke gebräuchlich:

Tag	Beschreibung
`name=*`	Offizielle Name der Rohrleitung, z.B. 'Westdeutsche Anbindungsleitung'
`ref=*`	Verwende diesen Tag, wenn es eine kurze, eindeutige Kennung für diese Rohrleitung gibt z.B. 'WEDAL'. Dieser Wert wird verwendet, um Pipelines logisch miteinander zu verknüpfen. In der Regel enthalten bereits die Markierungslaschen an den Rohrleitungen eine eindeutige Kennung.
`location=*`	Dieser Tag gibt die Lage der Leitung an. Mögliche Werte: • `location=underground` - für unterirdische, von der Oberfläche nicht sichtbare Rohrleitungen. • `location=overground` - für Rohrleitungen an oder in der Nähe der Oberfläche. Einer Person oder einem Fahrzeug ist es nicht möglich, unter der Leitung hindurch zu kommen. • `location=overhead` - für Rohrleitungen an der Oberfläche. Sie befindet sich in einer Höhe, in der es Personen oder Fahrzeuge möglich ist, unter der Leitung hindurch zu kommen. Falls zutreffend sollten andere Tags wie `maxheight=*` kombiniert werden. Wo die Überkopf-Rohrleitung die Handelsschifffahrt (Kreuzung der Kanal- oder Flussschifffahrt) beeinflussen könnte, sollte die Verwendung von `seamark:type=pipeline_overhead` geprüft werden. • `location=underwater` - für Rohrleitungen unter der Oberfläche auf dem Grund von Gewässern wie Seen oder Flüssen. Für Rohrleitungen/Pipelines auf dem Meer sollte das Seamark-Schema (en) verwendet werden.

man_made = pipeline

Beschreibung
Rohrleitungen für Wasser, Öl, Gas, Abwasser, Fernwärme usw.

Gruppe: Bauwerk

Für diese Elemente

Sinnvolle Kombinationen
- `location=*`
- `substance=*`
- `operator=*`

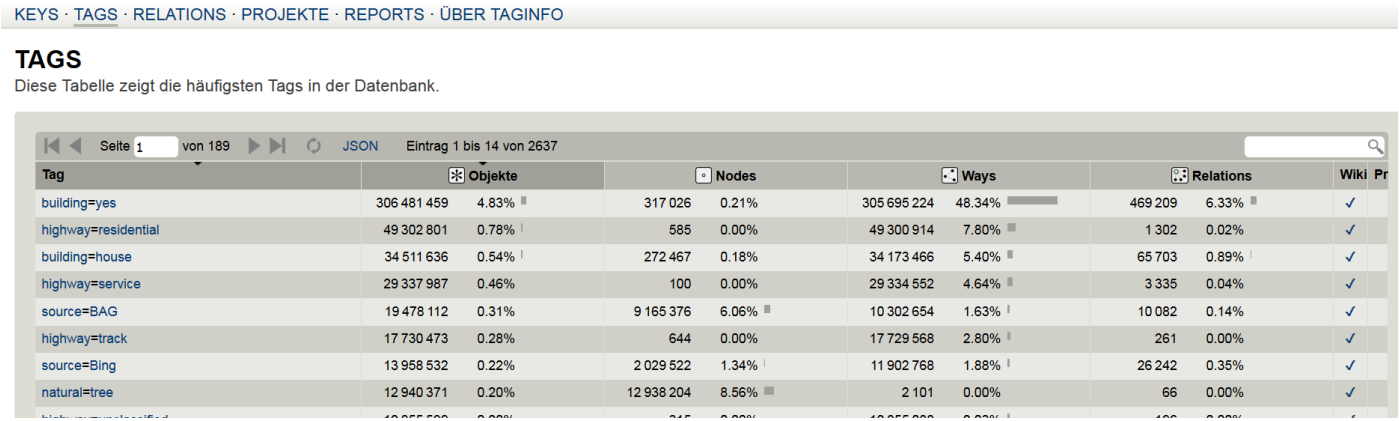
Wikidata
Q16885408

Status: akzeptiert

taginfo [More...]
695 0.05 %
130 403 7.22 %
70 0.84 %

Werkzeuge für dieses Attribut
- taginfo · CH · CZ · HU · PL
- overpass-turbo

https://taginfo.openstreetmap.org/



KEYS · TAGS · RELATIONS · PROJEKTE · REPORTS · ÜBER TAGINFO

TAGS

Diese Tabelle zeigt die häufigsten Tags in der Datenbank.

Seite 1 von 189 JSON Eintrag 1 bis 14 von 2637

Tag	Objekte		Nodes		Ways		Relations		Wiki	Pr
building=yes	306 481 459	4.83%	317 026	0.21%	305 695 224	48.34%	469 209	6.33%	✓	
highway=residential	49 302 801	0.78%	585	0.00%	49 300 914	7.80%	1 302	0.02%	✓	
building=house	34 511 636	0.54%	272 467	0.18%	34 173 466	5.40%	65 703	0.89%	✓	
highway=service	29 337 987	0.46%	100	0.00%	29 334 552	4.64%	3 335	0.04%	✓	
source=BAG	19 478 112	0.31%	9 165 376	6.06%	10 302 654	1.63%	10 082	0.14%	✓	
highway=track	17 730 473	0.28%	644	0.00%	17 729 568	2.80%	261	0.00%	✓	
source=Bing	13 958 532	0.22%	2 029 522	1.34%	11 902 768	1.88%	26 242	0.35%	✓	
natural=tree	12 940 371	0.20%	12 938 204	8.56%	2 101	0.00%	66	0.00%	✓	

1.1 esy-osmfilter

esy-osmfilter is a Python library to read and filter OpenStreetMap data files in the Protocol Buffers (PBF) format and export them to a Python dictionary and/or JSON file.

The OpenStreetMap PBF format defines three primary data types:

- `Node`: An annotated point on Earth
- `Way`: An list of `Node` items forming a path or polygon
- `Relation`: A set of related entries

This library relies on the esy.osm.pbf library in order to read the pbf-files.

gitlab repository: https://gitlab.com/dlr-ve-esy

documentation: https://dlr-ve-esy.gitlab.io/esy-osmfilter/

1.1.1 Features

What it provides:

- A pythonic way to work with `.pbf` data file entries
- An efficent way to filter OpenStreetMap data with Python
- customisable filters
- improved preformance by - parallization of the filtering process - storage and reuse of (intermediate) results with PICKLE and JSON

What it **doesn't** provide:

- A mechanism to spatially query OpenStreetMap entries.
- Visualization of OpenStreetMap data.

Workingflow

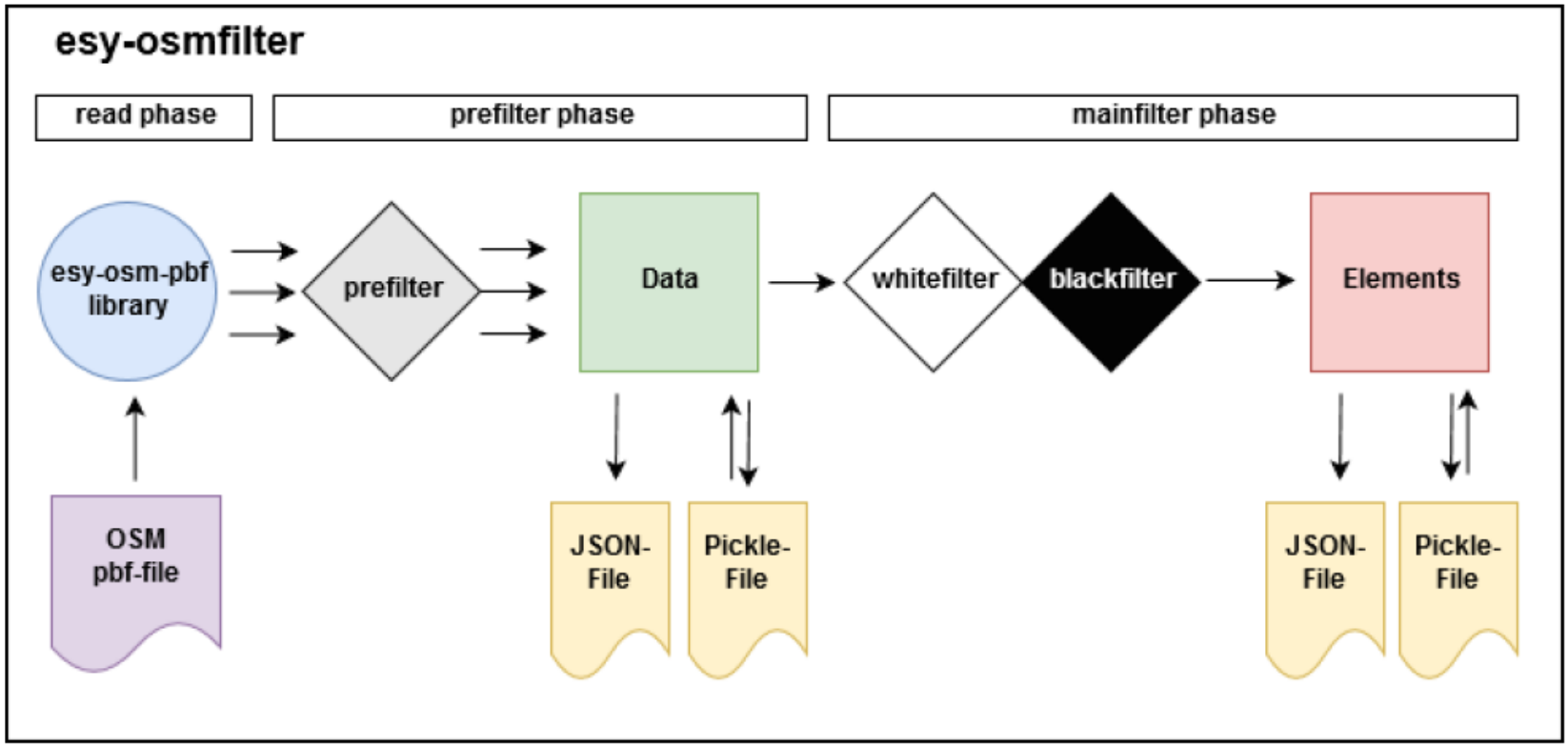


Fig.2: Working-flow of the esy-osmfilter library.

NOTE: Please upgrade pip first
python -m pip install -upgrade pip

1.1.2 Installation

`esy-osmfilter` depends on a Python version of 3.5 or above as well as on the Google Protocol Buffers. Use `pip` to install `esy-osmfilter`:

```
$ pip install esy-osmfilter
```

1.1.3 License

`esy-osmfilter` is licensed under the GNU General Public License version 3.0.

1.1.4 The Team

`esy-osmfilter` is developed at the DLR Institute of Networked Energy Systems in the departement for Energy Systems Analysis (ESY).

Initialize esy-osmfilter

1.2 Usage

In the following example the prefilter of esy-osmfilter is used to extract all pipelines from Lichtenstein.

First, all necessary libraries and functions are imported.

```
>>> import configparser, contextlib
>>> import os, sys
>>> from esy.osmfilter import osm_colors as CC
>>> from esy.osmfilter import run_filter
>>> from esy.osmfilter import Node, Way, Relation
```

```
>>> import urllib.request
>>> if not os.path.exists('lichtenstein-latest.osm.pbf'):
...     filename, headers = urllib.request.urlretrieve(
...         'https://download.geofabrik.de/europe/liechtenstein-191101.osm.pbf',
...         filename='lichtenstein-191101.osm.pbf'
...     )
...     PBF_inputfile = filename
```

```
>>> JSON_outputfile = os.path.join(os.getcwd(),
...                                'tests/output/LI/liechtenstein-191101.json')
```

Prefilter Phase

In the next step, a prefilter for all pipeline objects is defined. With the prefilter, we accept all way-items that have "man_made" as key and "pipeline" as value in their taglist. The white and black filter are left empty for the moment.

```
>>> prefilter   = {Node: {}, Way: {"man_made":["pipeline",],}, Relation: {}}
>>> whitefilter = []
>>> blackfilter = []
```

PLEASE NOTICE: You can also set *"man_made":True* to accept items independently of a key value.

```
>>> [Data,_]=run_filter('noname',
...                     PBF_inputfile,
...                     JSON_outputfile,
...                     prefilter,
...                     whitefilter,
...                     blackfilter,
...                     NewPreFilterData=True,
...                     CreateElements=False,
...                     LoadElements=False,
...                     verbose=True)
```

Workaround:
Disable multiprocessing for some Windows machines:
multiprocess=False

The prefilter returns the filter results to the `Data` dictionary. This means all OSM way-items with the tag "man_made"="pipeline" are stored there. But not enough, additionally, all referenced node items of these pipelines are stored there too.

```
>>> len(Data['Node'])
13
>>> len(Data['Relation'])
0
>>> len(Data['Way'])
2
```

In this example, we have only found two pipelines and their correspondent 13 nodes.

Mainfilter Phase

In the next step we use run_filter to load the `Data` dictionary and specify the main filtering results. In this example, we use the blackfilter to exclude possible pipelines substations from our prefiltering results.

```
>>> blackfilter = [("pipeline","substation"),]
```

We further only accept the drain pipelines that have the really great name “Wäschgräbli”.

```
>>> whitefilter =[(("waterway","drain"),("name","Wäschgräbli")), ]
```

We initiate the mainfilter phase by setting CreateElements=True.

```
>>> [_,Elements]=run_filter('funny-waterway-pipelines',
...                          PBF_inputfile,
...                          JSON_outputfile,
...                          prefilter,
...                          whitefilter,
...                          blackfilter,
...                          NewPreFilterData=False,
...                          CreateElements=True,
...                          LoadElements=False,
...                          verbose=True)
```

Accessing References and Relation Members

We see, that there is only one way-item left in the `Elements` dictionary, the other has been filtered out. There are no referenced nodes (or relation members) of the remaining way-item passed to the `Elements` dictionary.

```
>>> len(Elements['funny-waterway-pipelines']['Node'])
0
>>> len(Elements['funny-waterway-pipelines']['Relation'])
0
>>> len(Elements['funny-waterway-pipelines']['Way'])
1
```

Exporting to GeoJSON

However, esy-osmfilter comes with an export function for GeoJSON files (not implemented for relations yet) which will make thinks a lot easier:

```
>>> from esy.osmfilter import export_geojson
>>> export_geojson(Elements['funny-waterway-pipelines']['Way'],Data,
... filename='test.geojson',jsontype='Line')
{'location': 'underground', 'man_made': 'pipeline', 'name': 'Wäschgräble', 'waterway': 'drain'}
```

To visualize the output-file just open http://geojson.io and drag it on the screen.

Example: Prefilter for Gas Transport Data (Ways)

```
pre_filter = {
Node: {
Way: {
      "substance"     : ["gas","cng"],
      "man_made"      : ["gasometer","pipeline","petroleum_well","pumping_station",
                         "pipeline_marker","pipeline_station","storage_tank",
                         "gas_cavern"],
      "industrial"    : ["gas","terminal","cng"],
      "gas"           : ["station"],
      "content"       : ["gas","cng"],
      "pipeline"      : ["substation","marker","valve","pressure_control_station",
                         True],
      "storage"       : ["gas","cng"],
      "seamark:type"  : ["pipeline_submarine"],
      "land_use"      : ["industrial:gas"]
      },
 Relation: {},
 }
```

<u>Results: Europe 2019</u>

Ways: 207393

Example: Whitefilter for Gas Transport Data

The whitefilter is a **list of list of (key,value) tuples**. If any **list of (key,value) tuples** is present, the item will pass (if it is not blocked by the blacklist at the same time).

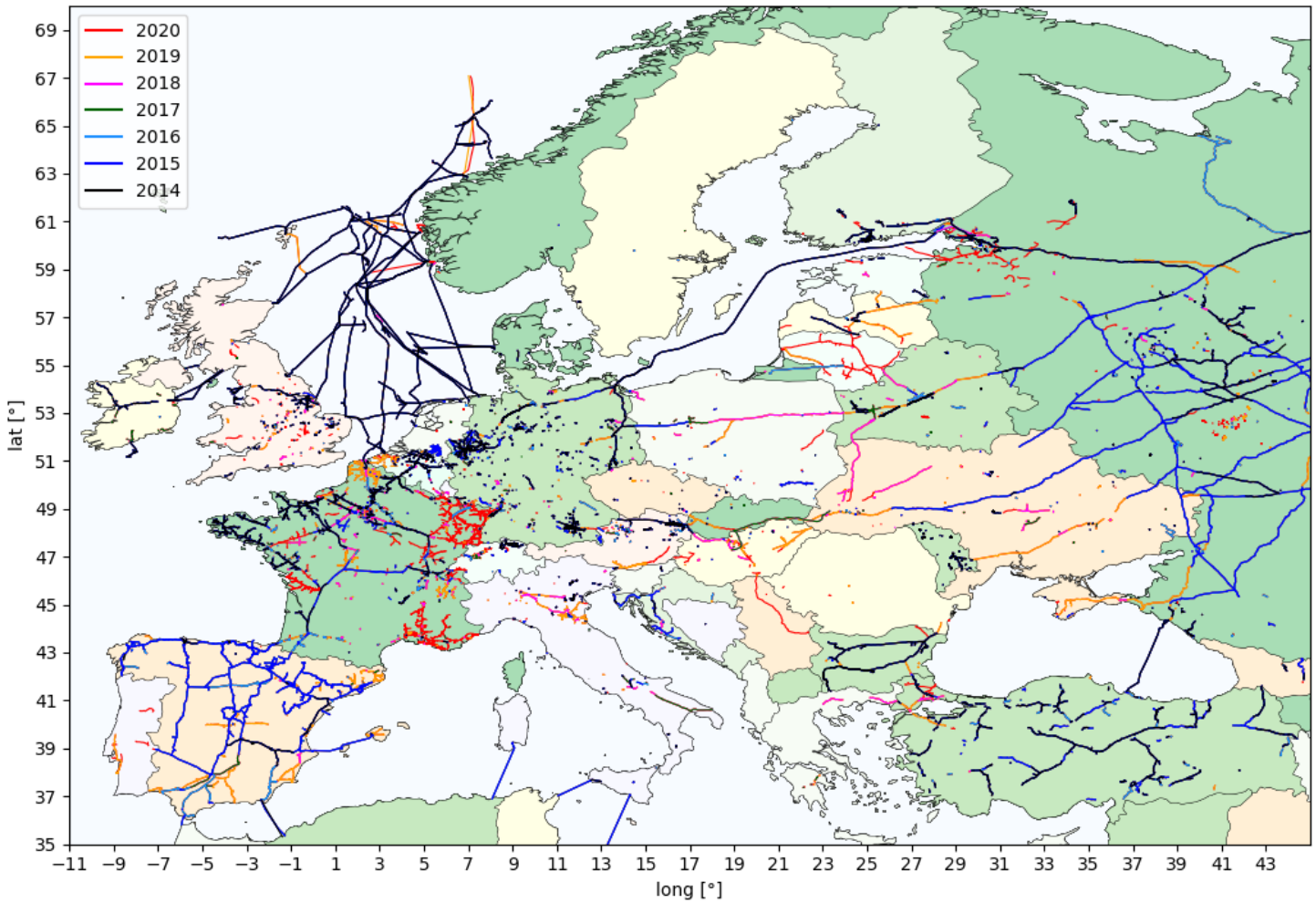
```
whitefilter=[
             (("man_made","pipeline"),("substance","gas")),
             (("man_made","pipeline"),("substance","cng")),
             (("man_made","pipeline"),("substance","natural_gas")),
             (("man_made","pipeline"),("pipeline:type","natural_gas")),
             (("man_made","pipeline"),("type","gas")),
             (("man_made","pipeline"),("type","natural_gas")),
             (("man_made","pipeline"),("type","cng")),
             (("man_made","pipeline"),("industrial","gas")),
             (("man_made","pipeline"),("industrial","cng")),
             (("man_made","pipeline"),("industrial","natural_gas"))
            ]
```

Example: Blackfilter for Gas Transport Data

The blackfilter is a **list of (key,value) tuples**, which will not let items pass if one of the **(key,value) tuples** is present. It is prioritized above the whitelist.

```
blackfilter=[
            ("pipeline","substation"),
            ("substation","distribution"),
            ('usage', 'distribution'),
            ('pipeline:type','water'),
            ('pipeline:type', 'sewer'),
            ("pumping_station","water"),
            ("pumping_station","sewage"),
            ('pumping_station','wastewater'),
            ("type","wastewater"),
            ("type","fuel"),
            ("type","sewage"),
            ("type","oil"),
            ("type","water"),
            ("substance","sewage"),
            ("substance","water"),
            ("substance","hot_water"),
            ("substance","fuel"),
            ("substance","wastewater"),
            ("substance","rainwater"),
            ("substance", "drain"),
            ("substance", "heat"),
            ("substance", "gas,heat"),
            ("substance", "heat,gas"),
            ("substance", "ammonia"),
            ("substance", "ethylen"),
            ("substance","oil")
            ]
```

Example: European Gas Transport Pipelines



European gas transport pipelines in OSM colored after entering year.

Presentation and References:

This document will be uploaded to the SciGRID gas homepage:

https://www.gas.scigrid.de/archives.html

How to reference esy-osmfilter:

Upcoming article in 2020:

Journal of Open Research Software:

esy-osmfilter - A Python Library to Efficiently Extract OpenStreetMap data

Adam Pluta, Ontje Lünsdorf

Some other common tags:

Seite 1 von 165 JSON Eintrag 1 bis 16 von 2638

Tag	Objekte		Nodes		Ways		Relations	
building=yes	306 831 065	4.83%	319 214	0.21%	306 042 429	48.34%	469 422	6.32%
highway=residential	49 353 773	0.78%	585	0.00%	49 351 887	7.80%	1 301	0.02%
building=house	34 552 017	0.54%	275 498	0.18%	34 210 695	5.40%	65 824	0.89%
highway=service	29 400 082	0.46%	105	0.00%	29 396 642	4.64%	3 335	0.04%
source=BAG	19 479 197	0.31%	9 165 838	6.06%	10 303 276	1.63%	10 083	0.14%
highway=track	17 748 352	0.28%	645	0.00%	17 747 445	2.80%	262	0.00%
source=Bing	13 962 136	0.22%	2 029 544	1.34%	11 906 341	1.88%	26 251	0.35%
natural=tree	12 951 926	0.20%	12 949 757	8.56%	2 103	0.00%	66	0.00%
highway=unclassified	12 867 360	0.20%	319	0.00%	12 866 847	2.03%	194	0.00%
surface=asphalt	12 148 862	0.19%	37 503	0.02%	12 100 443	1.91%	10 916	0.15%
waterway=stream	12 140 039	0.19%	6 160	0.00%	12 112 784	1.91%	21 095	0.28%
power=tower	12 030 063	0.19%	12 029 743	7.95%	320	0.00%	0	0.00%
wall=no	12 000 381	0.19%	315	0.00%	11 993 345	1.89%	6 721	0.09%
source=cadastre-dgi-fr_source_:_Direction_Généra	11 777 664	0.19%	289 371	0.19%	11 485 608	1.81%	2 685	0.04%
oneway=yes	11 671 876	0.18%	600	0.00%	11 669 857	1.84%	1 419	0.02%
highway=footway	11 358 066	0.18%	358	0.00%	11 350 891	1.79%	6 817	0.09%

→ **OpenStreetMap** · Data © OSM contributors (ODbL) Datenquellen · Do

DLR