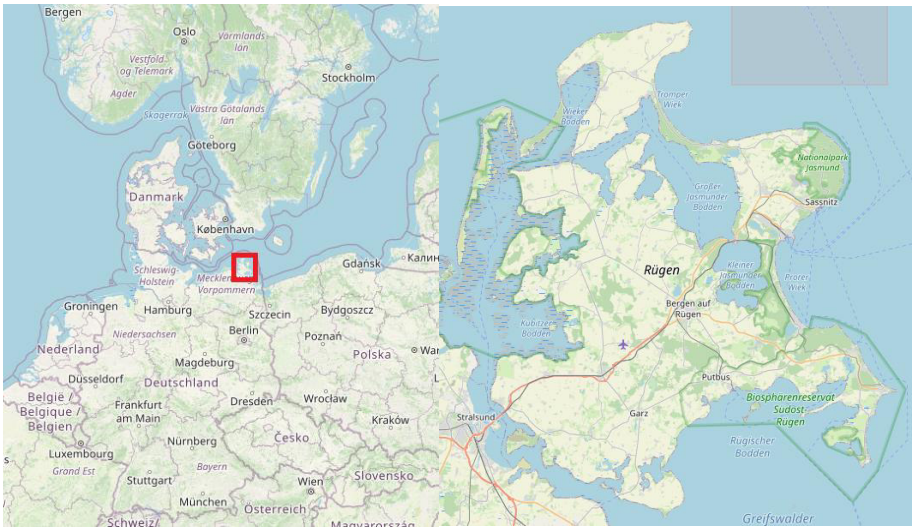
Project: Wrangling OpenStreetMap Data

Oliver Koch, 17 April 2020

1 Map Area

- Rügen, Meckenburg-Vorpommern, Germany
- OpenStreetMap relation: https://www.openstreetmap.org/relation/1434381
- Wikipedia: https://de.wikipedia.org/wiki/Rügen

I chose this region because it is one of my favorite places for holidays and extended weekends.



2 Example query and problems encountered in the data

As an initial query, I counted the occurance of city (village) names, originating from `addr:street` tags.

This query highlighted some problems in the data:

2.1 City names in nodes_tags and ways_tags

City names occur both in nodes tags and ways tags. To overcome this, I created a union of the `nodes_tags` and `ways_tags` tables as a view, giving it the name `tags`. The figure below compares the results for the query on table `nodes_tags` and for the query on the combined `tags` view.

```
CREATE VIEW tags AS
    SELECT * FROM nodes_tags
    UNION ALL
    SELECT * FROM ways_tags;
```

```
Q("""
SELECT ci.value AS city, COUNT(*) AS num
FROM nodes_tags ci, nodes_tags na
WHERE ci.type = 'addr'
  AND ci.key = 'city'
  AND na.type = 'regular'
  AND na.key = 'name'
  AND ci.id = na.id
GROUP by city
ORDER BY num DESC
LIMIT 10
""")
```

	city	num
0	Binz	151
1	Sassnitz	91
2	Bergen auf Rügen	60
3	Stralsund	52
4	Sellin	49
5	Hiddensee	25
6	Baabe	23
7	Mönchgut	21
8	Putbus	21
9	Breege	17

```
Q("""
SELECT ci.value AS city, COUNT(*) AS num
FROM tags ci, tags na
WHERE ci.type = 'addr'
  AND ci.key = 'city'
  AND na.type = 'regular'
  AND na.key = 'name'
  AND ci.id = na.id
GROUP by city
ORDER BY num DESC
LIMIT 10
""")
```

	city	num
0	Binz	379
1	Sassnitz	156
2	Sellin	130
3	Bergen auf Rügen	116
4	Stralsund	101
5	Hiddensee	99
6	Mönchgut	88
7	Baabe	60
8	Glowe	57
9	Breege	49

2.2 Different names are used for the same city

```
<tag k="addr:city" v="Ostseebad Sellin" />
<tag k="addr:city" v="Sellin" />
<tag k="addr:city" v="Garz/Rügen" />
<tag k="addr:city" v="Insel Hiddensee" />
<tag k="addr:city" v="Hiddensee" />
```

The table below shows some examples of odd city names encounted.

```
Garz      -- Garz/Rügen
Hiddensee -- Insel Hiddensee
Baabe     -- Ostseebad Baabe,
Putbus    -- Putbus / OT Vilmnitz
```

In the frist case, both Garz and Garz/Rügen are used. The latter form is usually used to distinguish this particular village of Garz on Rügen from other villages with the same name. Because we are interested in viallages on the island of Rügen only, the suffix can be dropped.

In the second case, both „Hiddensee“ and „Insel Hiddensee“ are used. Hiddensee the name is a small island belonging to Rügen, and also the name of the village there. Talking about `addr:city` tags here, I assume that the authors mean the village of Hiddensee and not the island. The prefix „Insel“ is hence dropped.

The third case is an example of a name prefix. „Ostseebad“ could be translated as „seaside resort“. The prefix is used in some entries, in others not. A By dropping the „Ostseebad“ prefix, a clear representation of the place is made.

In the fouth case, „OT Vilmnitz“ specifies with part of the town of Putbus is meant. By dropping this suffix, the city can be identified as Putbus.

The highlighted issues are resolved during data cleaning (see file `addr_city.py`) using regular expressions.

RE for detecting „strange“ city names on `addr:city` tags:

```
RE_CITY_WITH_ADDON = re.compile("^Ostseebad|^Insel|[^ a-zäöüßA-ZÄÖÜ-]")
```

RE implementing the substitutions explained above:

```
re.sub("^Ostseebad *|^Insel *| */.*$", "", cityname)
```

2.3 European and Asian alphabets

The OSM data contains names for places in various languages and alphabets. The OSM xml file format clearly states UTF-8 as the used encoding.

```
<tag k="name" v="Rügen"/>
<tag k="name:ar" v="روغن"/>
<tag k="name:cs" v="Rujána"/>
<tag k="name:he" v="רוגן"/>
<tag k="name:is" v="Rügen"/>
<tag k="name:it" v="Rügen"/>
<tag k="name:ja" v="リューゲン島"/>
```

The provided example code was not able to handle this content without some modifications. I ported the code to Python 3, thus eliminating the need for UNICODE data type and use of the codings library. Care was taken to explicitely specify the UTF-8 encoding when writing and reading csv files.

```
with open(os.path.join(dirname, NODES_FILE), 'w',
          encoding="utf-8", newline='') as nodes_file:
```

```
with open(csv_path, encoding='utf-8') as fp:
    reader = csv.DictReader(fp)
```

3 Overview of the data

Data was downloaded on the 16th of April 2020 the this URL:

https://overpass-api.de/api/map?bbox=13.0943,54.2123,13.7769,54.6945

The uncompressed osm file is 124 MB, it contains 519104 nodes, 83902 ways and 49892 members (analysed with count_elements.py).

Only nodes and ways were loaded into the database. The figure below provides overview statistics on the number of elements. The number of nodes and ways is the same as in the osm file, meaning that no elements were deleted because of bad quality.

```
Q("""
SELECT 'nodes' as element, count(*) AS num FROM nodes
UNION
SELECT 'nodes_tags' as element, count(*) AS num FROM nodes_tags
UNION
SELECT 'ways' as element, count(*) AS num FROM ways
UNION
SELECT 'ways_tags' as element, count(*) AS num FROM ways_tags
UNION
SELECT 'ways_nodes' as element, count(*) AS num FROM ways_nodes
""")
```

	element	num
0	nodes	519104
1	nodes_tags	89588
2	ways	83902
3	ways_nodes	784533
4	ways_tags	224127

The dataset was edited by 1277 unique users, 263 of them are one-time editors. The top user made over 150000 contributions.

Top 10 users

```
Q("""
SELECT user, count(*) AS edits
  FROM (SELECT user FROM nodes
        UNION ALL
        SELECT user FROM ways)
GROUP BY user
ORDER BY edits DESC
LIMIT 10;
""")
```

	user	edits
0	da-sch	150202
1	Geofreund1	81407
2	Schwedenhagen	61250
3	jacobbraeutigam	46374
4	!i!	17791
5	Kiekin	12646
6	nbuettler	10390
7	north	9084
8	changchun_1	8515
9	SunCobalt	7035

4 Further investigations performed on the data

A further query investigates restaurants, cafés, and fast food places. I wanted to find out their name, place, and style of cuisine. The query below shows the most common styles.

```
Q("""
select cuisine, count(*) as num
from resto
group by cuisine
order by num desc
limit 15
""")
```

	cuisine	num
0	None	354
1	regional	79
2	fish	39
3	german	28
4	italian	22
5	kebab	15
6	ice_cream	14
7	german;regional	9
8	greek	7
9	pizza	5
10	regional;german	4
11	asian	3
12	burger	3

Problems and challenges encountered, in brief:

- Both nodes and ways are tagged as `amenity:restaurant`, `amenity:cafe` etc. I again used the `tags` union defined above.
- Only very few places provide all information. The query below creates a view of all restaurant-like places, and the additional properties if present, otherwise NULL. This stackoverflow post was helpful in formulating the query: https://stackoverflow.com/questions/1237068/how-to-pivot-in-sqlite-or-i-e-select-in-wide-format-a-table-stored-in-long-form
- Looking at the results (figure on top), we can see that there is not a clear-cut distinction between cuisine styles (if defined). A popular category is `regional`, but also `german;regional` and `regional;german` occur quite often. A more detailed investigation would do auditing of the ciusine types and come up with an idea for cleaning and combining the categories.

```
DROP VIEW IF EXISTS resto_tags;
CREATE VIEW resto_tags AS
    select id, key, value
    from tags
    where id in (select id
                 from tags
                 where key = 'amenity'
                 and value in ('restaurant', 'cafe', 'fast_food', 'ice_cream')
                 )
      and key in ('amenity', 'name', 'cuisine', 'city');

DROP VIEW IF EXISTS resto;
CREATE VIEW resto AS
    select distinct
        am.id as id,
        na.value as name,
        ci.value as city,
        cu.value as cuisine
    from
        resto_tags am
        left join resto_tags na on
            am.id = na.id
            and na.key = 'name'
        left join resto_tags ci on
            am.id = ci.id
            and ci.key = 'city'
        left join resto_tags cu on
            am.id = cu.id
            and cu.key = 'cuisine';
```

Appendix: Code and file overview

```
# script for creating a smaller sample of the full dataset
sampling.py

# scripts for auditing and cleaning various aspects of the data
count_elments.py
tags.py
street_abbrev.py
street_housenum.py
street_type.py
postcode.py
addr_city.py

# scripts for extracting, cleaning and transforming into csv
# use of some of the cleaning functions above
prepare_csv.py
schema.py

# files for creating and filling the database
load_database.py
osm.sql

# jupyter notebook for queries (printed version in appendix)
queries.ipynb
```

Appendix: Documentation of queries

In [1]:
```
import pandas as pd
import sqlite3
```

Simple Query functions

In [2]:
```
DBPATH = "full/osm_ruegen.db"

def Q(query):
    """return query result as pd.DataFrame"""
    with sqlite3.connect(DBPATH) as conn:
        df = pd.read_sql_query(query, conn)
    return df

def H(query):
    """Display query result as html string, without index column"""
    print(Q(query).to_html(index=False))
```

Create views

- Union of way and node tags.
- restaurant tags
- a join of restaurant attributes

In [3]:
```
view_query = """
DROP VIEW IF EXISTS tags;
CREATE VIEW tags AS
   SELECT * FROM nodes_tags
   UNION ALL
   SELECT * FROM ways_tags;

DROP VIEW IF EXISTS resto_tags;
CREATE VIEW resto_tags AS
    select id, key, value
    from tags
    where id in (select id
                 from tags
                 where key = 'amenity'
                 and value in ('restaurant', 'cafe', 'fast_food', 'ice_cream')
                 )
      and key in ('amenity', 'name', 'cuisine', 'city');

DROP VIEW IF EXISTS resto;
CREATE VIEW resto AS
    select distinct
        am.id as id,
        na.value as name,
        ci.value as city,
        cu.value as cuisine
    from
        resto_tags am
        left join resto_tags na on
            am.id = na.id
            and na.key = 'name'
        left join resto_tags ci on
            am.id = ci.id
            and ci.key = 'city'
        left join resto_tags cu on
            am.id = cu.id
            and cu.key = 'cuisine';

"""

with sqlite3.connect(DBPATH) as conn:
    cur = conn.cursor()
    cur.executescript(view_query)
    conn.commit()
```

Number of Nodes, Ways, tags

In [4]:
```
Q("""
SELECT 'nodes' as element, count(*) AS num FROM nodes
UNION
SELECT 'nodes_tags' as element, count(*) AS num FROM nodes_tags
UNION
SELECT 'ways' as element, count(*) AS num FROM ways
UNION
SELECT 'ways_tags' as element, count(*) AS num FROM ways_tags
UNION
SELECT 'ways_nodes' as element, count(*) AS num FROM ways_nodes
""")
```

Out[4]:

	element	num
0	nodes	519104
1	nodes_tags	89588
2	ways	83902
3	ways_nodes	784533
4	ways_tags	224127

Users

unique users

In [5]:
```
Q("""
SELECT count(user) as num_users
  FROM (SELECT DISTINCT user FROM nodes
        UNION
        SELECT DISTINCT user FROM ways)
""")
```

Out[5]:

	num_users
0	1277

Number of users having only one edit

In [6]:
```
Q("""
SELECT count(*) as num_once
FROM (SELECT user, count(*) AS edits
        FROM (SELECT user FROM nodes
              UNION ALL
              SELECT user FROM ways)
      GROUP BY user
      HAVING edits = 1
);
""")
```

Out[6]:

	num_once
0	263

Top 10 users

In [7]:
```
Q("""
SELECT user, count(*) AS edits
  FROM (SELECT user FROM nodes
        UNION ALL
        SELECT user FROM ways)
GROUP BY user
ORDER BY edits DESC
LIMIT 10;
""")
```

Out[7]:

	user	edits
0	da-sch	150202
1	Geofreund1	81407
2	Schwedenhagen	61250
3	jacobbraeutigam	46374
4	!i!	17791
5	Kiekin	12646
6	nbuettler	10390
7	north	9084
8	changchun_1	8515
9	SunCobalt	7035

Query example: Cities, sorted by frequency of apprearance

In [8]:
```
Q("""
SELECT ci.value AS city, COUNT(*) AS num
FROM nodes_tags ci, nodes_tags na
WHERE ci.type = 'addr'
  AND ci.key = 'city'
  AND na.type = 'regular'
  AND na.key = 'name'
  AND ci.id = na.id
GROUP by city
ORDER BY num DESC
LIMIT 10
""")
```

Out[8]:

	city	num
0	Binz	151
1	Sassnitz	91
2	Bergen auf Rügen	60
3	Stralsund	52
4	Sellin	49
5	Hiddensee	25
6	Baabe	23
7	Mönchgut	21
8	Putbus	21
9	Breege	17

In [9]:
```
Q("""
SELECT ci.value AS city, COUNT(*) AS num
FROM tags ci, tags na
WHERE ci.type = 'addr'
  AND ci.key = 'city'
  AND na.type = 'regular'
  AND na.key = 'name'
  AND ci.id = na.id
GROUP by city
ORDER BY num DESC
LIMIT 10
""")
```

Out[9]:

	city	num
0	Binz	379
1	Sassnitz	156
2	Sellin	130
3	Bergen auf Rügen	116
4	Stralsund	101
5	Hiddensee	99
6	Mönchgut	88
7	Baabe	60
8	Glowe	57
9	Breege	49

Restaurants, Cafés etc.

- problem 1: both on nodes and ways
- solution: combied table / view
- problem 2: i want name, place, and cuisine style.
- for most resaurants, not all info is present
- problem 3: several cuisines, separated by semikolon

some statistics

In [10]:
```
Q("""
select count(*) num_in_nodes
from nodes_tags
where key = 'amenity'
and value in ('restaurant', 'cafe', 'fast_food', 'ice_cream')
""")
```

Out[10]:

	num_in_nodes
0	524

In [11]:
```
Q("""
select count(*) num_in_ways
from ways_tags
where key = 'amenity'
and value in ('restaurant', 'cafe', 'fast_food', 'ice_cream')
""")
```

Out[11]:

	num_in_ways
0	130

In [12]:
```
Q("""
select count(*)
from tags
where key = 'amenity'
and value in ('restaurant', 'cafe', 'fast_food', 'ice_cream')
""")
```

Out[12]:

	count(*)
0	654

In [13]:
```
Q("""select count(distinct id) as total from resto_tags""")
```

Out[13]:

	total
0	654

In [14]:
```
Q("""
select 'amenity' as key, count(*) as n from resto_tags where key='amenity'
union
select 'city' as key, count(*) as n from resto_tags where key='city'
union
select 'name' as key, count(*) as n from resto_tags where key='name'
union
select 'cuisine' as key, count(*) as n from resto_tags where key='cuisine'
""")
```

Out[14]:

	key	n
0	amenity	654
1	city	275
2	cuisine	300
3	name	612

joined table

In [15]:
```
Q("""
select name, city, cuisine
from resto
order by city
""")
```

Out[15]:

	name	city	cuisine
0	Venezia	None	italian
1	Schwalbennest	None	None
2	Bernstein	None	None
3	"Zur Sandbank"	None	regional
4	Hitthim	None	None
...	...	...	...
649	Café Ummanz	Ummanz	coffee_shop;cake
650	Hiddenseer Fischerklause	Vitte	None
651	Zur Wittower Fähre	Wiek	None
652	Blumencafe	Wiek	coffee_shop
653	Jacobs - Die Fischgaststätte	Wiek	None

654 rows × 3 columns

Where are most restaurants located?

In [16]:
```
Q("""
select city, count(*) as num
from resto
group by city
order by num desc
limit 10
""")
```

Out[16]:

	city	num
0	None	379
1	Binz	50
2	Sassnitz	29
3	Sellin	22
4	Stralsund	20
5	Baabe	16
6	Hiddensee	16
7	Mönchgut	15
8	Putbus	13
9	Glowe	12

Most common cuisine?

In [17]:
```
Q("""
select cuisine, count(*) as num
from resto
group by cuisine
order by num desc
limit 8
""")
```

Out[17]:

	cuisine	num
0	None	354
1	regional	79
2	fish	39
3	german	28
4	italian	22
5	kebab	15
6	ice_cream	14
7	german;regional	9

Population

- actual population: Wikipedia has two figures: 62900 or 77000. Not clear which value is correct.
- https://de.wikipedia.org/wiki/Rügen (https://de.wikipedia.org/wiki/Rügen)
- Database result is of the same order of magnitude an in between the wikipedia data -> plausible.

In [18]:
```
Q("""
select sum(po.value) as total_population
from nodes_tags po
     join nodes_tags na using(id)
     join nodes_tags pl using(id)
where po.key = 'population'
  and po.type = 'regular' --> important because there are also OpenGeoDB entries!!!
  and na.key = 'name'
  and na.type = 'regular'
  and pl.key = 'place'
  and pl.type = 'regular'
""")
```

Out[18]:

	total_population
0	69620

In [19]:
```
Q("""
select na.value as city, 1*po.value as population
from nodes_tags po
     join nodes_tags na using(id)
     join nodes_tags pl using(id)
where po.key = 'population'
  and po.type = 'regular' --> important because there are also OpenGeoDB entries!!!
  and na.key = 'name'
  and na.type = 'regular'
  and pl.key = 'place'
  and pl.type = 'regular'
order by population desc
limit 10
""")
```

Out[19]:

	city	population
0	Bergen auf Rügen	14328
1	Sassnitz	9481
2	Ostseebad Binz	5595
3	Putbus	4330
4	Sellin	2540
5	Sagard	2512
6	Garz	2220
7	Samtens	2209
8	Dranske	1677
9	Brandshagen	1308