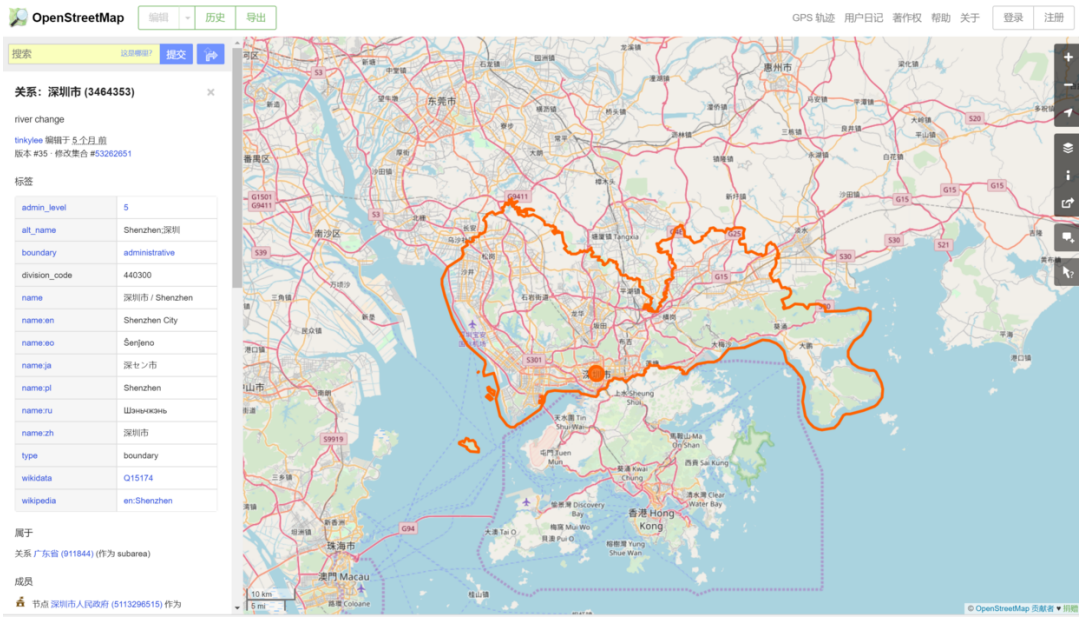
OpenStreetMap Case Study

1. 选定区域

中国深圳
- https://www.openstreetmap.org/relation/3464353

选取的区域是深圳，`shenzhen.osm` 大小 145.328M。以下为选定区域示意图：



2. 选择区域的原因

选择深圳市地图的主要原因是： 我在深圳工作生活，比较熟悉，比较容易找到那些数据是不合适。

3. 地图数据中的问题

我将下载后的 `shenzhen.osm` 通过 `make_a_sample.py` 代码，转换为一个小型的 `sample.osm` 文件，对数据进行审查,必要时对原数据进行审查。
1.街道名称不统一
通过 `street_names.py` 查看 `shenzhen.osm` 中的街道地址时，发现街道地址描述标准不一，有中文，如南山大道、公园路等，也有英文San Hong

Street、San Fung Avenue等，以及中文+英文。由于结果显示了较多的地址名字描述不一致的情形，因此暂时不处理地址描述问题。

2.审查标签类型时，出现异常标签
使用`tags.py`中的正则表达式查看`sample.osm`数据中下的'k'属性标签类型时，没有发现问题 key（problemchars 类型），但是审查原数据`shenzhen.osm`时发现有一个问题key(problemchars 类型)，发现有异常标签。

将这个问题标签打印出来，结果是`name:MT Bike`,这是个有特殊字符的标签，这个特殊字符就是空格，可以将空格去掉，使用下划线将空格替换，同时只截取“：”后部分。在`tags.py`使用函数`process_problematic_keys(filename)`进行处理，得到`MT_Bike`.

3.审查时发现部分邮政编码有问题
利用`audit_post_code.py`审查`shenzhen.osm`中的邮政编码时，发现部分邮政编码不是6位数字，而中华人民共和国邮政编码应该为6位
`shenzhen.osm`中出现了5位和8位的邮编，均为异常邮编，在写入数据的时候，我们选取问题邮编中同时含有字符"DD"和空格的邮编进行处理。
问题邮编数据清洗见`clean_post_code.py`.

4. 将数据写入CSV文件

按照“案例研究"：OpenStreetMap数据[SQL]”中准备数据集的方法，将深圳市地图相关数据读入csv文件中，请查看`data.py`中的代码。

5. 将深圳市地图写入数据库

5.1 创建数据库

创建`shenzhen.db`数据库

5.2 将CSV文件导入sql table

用python代码将`nodes.csv`，`nodes_tags.csv`，`ways.csv`，`way_nodes.csv`，`ways_tags.csv`这5个csv文件分别写进数据库`shenzhen.db`中，对应表格分别为 nodes, nodes_tags, ways, ways_nodes, ways_tags。

python代码见 `import_nodes_csv.py` `import_nodes_tags_csv.py` `import_ways_csv.py` `import_way_nodes_csv.py` `import_ways_tags.py` 文件，以下以 nodes.csv 导入 nodes 数据表为例，代码如下：

```
# coding: utf-8

import csv
import sqlite3

db = sqlite3.connect("shenzhen.db")
db.text_factory = str
c = db.cursor()


c.execute('drop table if exists nodes')

nodes = '''
create table nodes
(
id Integer NOT NULL PRIMARY KEY,
lat float,
lon float,
user Text,
uid Integer,
version Integer,
changeset Integer,
timestamp Text
);
'''

c.execute(nodes)

with open('nodes.csv','rb') as nodes_f:
    dict_reader = csv.DictReader(nodes_f)
    for row in dict_reader:

        id_value = int(row['id'])
        lat_value = float(row['lat'])
        lon_value = float(row['lon'])
        user_value = str(row['user'])
        uid_value = int(row['uid'])
        version_value = int(row['version'])
        changeset_value = int(row['changeset'])
        timestamp_value = str(row['timestamp'])

        c.execute('INSERT INTO nodes VALUES (?,?,?,?,?,?,?,?)',(
```

```
db.commit()
db.close()
```

6. 用sql查询数据

1.查询数据库的表格
sqlite> .tables
nodes nodes_tags ways ways_nodes ways_tags

2.查询表node和ways的数量

sqlite> select count() from nodes; 721154
sqlite> select count() from ways； 78078

3.查询唯一用户的数量

```
# coding=utf-8
import csv
import sqlite3
db = sqlite3.connect("shenzhen.db")
c = db.cursor()

#提取 nodes 数据表中的独立用户数
query1 = "select uid from nodes group by uid order by uid"
c.execute(query1)
nodes_uid = c.fetchall()
print 'Unique uid in table nodes is:'
print len(nodes_uid)    #748

#提取 ways 数据表中的独立用户数
query2 = "select uid from ways group by uid order by uid"
c.execute(query2)
ways_uid = c.fetchall()
print 'Unique uid in table ways is:'
print len(ways_uid)   #534

#计算整个数据库中的独立用户数，需要将nodes数据表和ways数据表中的独立用户数相加
for i in range(len(ways_uid)):
    if ways_uid[i] not in nodes_uid:
        nodes_uid.append(ways_uid[i])
unique_uid = len(nodes_uid)
print 'total unique users is:'
print unique_uid    #832
```

7.关于数据集的其他想法

对于非英语国家，在地址描述上会显得比较杂乱，可以设置约束，某些字段要求用英文来描述，再增加本国语言描述的字段，方便使用。
这样做的好处在于：有统一的约束条件之后，可以让不同用户编辑的时候，有统一的标准，避免每个用户都按照自己的标准来编辑，导致最终清理时，出现不统一的情况。
预期的问题：统一地图上的元素命名规则后，需要注意街道名称尽量采用全称，减少缩写和不规范的情况，各元素信息应当遵循规范化的表达。
在本次数据整理和思考的过程中，无疑可以看出 OpenStreetMap 的数据确实有很多不规范的地方，这也是开源项目不可避免会出现的问题。如果真的有需要使用这些数据进行数据分析工作，一定要进行很多的数据整理，并且这些工作还可能会随着分析的深入需要反复进行。第一次窥探到现实世界数据分析工作的冰山一角，很有挑战性！