Finding ideal home location with OpenStreetMap (https://www.openstreetmap.org/)

by ***Paul F. Seke***

1. Dataset information

1.1. Background

It is not easy to choose the area to establish to when moving to a new city. The equation becomes even more complex when a family has young kids.. In this situation, most people would want to find a permanent accomodation in an area with a school, parkings, and one or more restaurants. In addition, emergency services, including hospitals, firefighters and police stations, can come handy in a number of unexpected but quite common situations. The OpenStreetMap (https://www.openstreetmap.org) project offers a brilliant way for mapping cities in a "Wikipedia" fashion. This project may constitute a good source of information for locating the right area for a home in a given city.
However, the strength of this project, i.e. the possibility for numerous users from different background to contribute data or verify their accuracy, also constitutes it's weakness. Notably, such project is expected to include a number of human errors in large databases. Furthermore, no data is available for cities not "added" to databases by users.

1.2. Study objectives

In the present study, we will use data wrangling techniques (using python and its csv, pandas, and sqlite3 modules mainly) on data from 9 cities' databases to detect and clean errors. Then, we will plot the locations of schools (and kindergarten), hospitals, parkings, restaurants, as well as police and firefighter stations using latitudes and longitudes from cleaned city databases. Of the 9 cities, Verona (Italy) was chosen because it is a city I have a good knowledge of (5 years spent there) and whose data were available. I also chose some cities I visit seasonally and that I really like such as Nairobi (Kenya), Turin (Italy) and Lausanne (Switzerland). In additions, I have added some cities I have heard a lot about from friends and that I would like to visit, and why not, establish to: Saint Louis Archipelago (Senegal), Durban (South Africa), and the Canadian cities of Fredericton, Quebec City, and Halifax. About the cities:

- African cities: Durban (https://en.wikipedia.org/wiki/Durban) 2,292 km^2, population 3.5 million; Nairobi (https://en.wikipedia.org/wiki/Nairobi) 696 km^2, population 47.2 million; Saint Louis (http://www.preventionweb.net/files/section/230_SaintLouisisgettingready.pdf) 46 km^2, population 176,000
- Canadian cities (http://www.statcan.gc.ca/tables-tableaux/sum-som/l01/cst01/demo05a-eng.htm): Quebec city 484.1 km^2, population 806,400; Halifax 5,490 km^2, population 417,800; Fredericton 130.7 km^2, population 56,224
- European cities: Verona (https://en.wikipedia.org/wiki/Verona): 206.6 km^2, population 259,069; Turin (https://en.wikipedia.org/wiki/Turin): 130.2 km^2, population 892,649; Lausanne (https://en.wikipedia.org/wiki/Lausanne): 41.37 km^2, population 146,372

2. OSM data processing

THe OpenStreetMap data are stored in a proprietary xml format termed as osm format. They can be downloaded from mapzen.com (https://mapzen.com/data/metro-extracts/).

2.1. From osm to csv

2.1.1. Filtering, validation and conversions

We downloaded data of all the aforementioned cities on August 2016. Typically, elements of osm file have the following tags: node, way, relation, tag, and nd (for way nodes). Node attributes include: 'id', 'lat', 'lon', 'user', 'uid' (user ID), 'version', 'changeset', and 'timestamp'. Ways have the same attributes except for latitude and longitude. Tags include 'k' (key) and 'v' (value), and way node only a 'ref' (reference) value.
Considering our focus (on nodes, ways and their tags mainly), osm files were processed as follows:

- only valid node, way and relation elements with valid ID (unique identifier) were included. ID validity was assessed by checking the convertibility into integer (as they were expected to be). Data failing to meet this criterion were filtered out (see **osm2csv_step0.py**).
- the other attributes were also converted to the format in which they were supposed to be. Data not meeting the standard (not valid) were dealt with as follows (also in **osm2csv_step0.py**):
  - node latitude and longitude were converted from string to float; missing or not float were filtered out without discarding the entire element. Node and way changeset and time stamp missing or erroneous were treated similarly. Way nodes with references not convertible to integer were discarded as well
  - tag keys with problematic characters were filtered out using regular expressions detecting the characters =+/&<>;\'"\?%#
  - mixed tag keys and types where also detected using regular expressions and separated programmatically (e.g. key = 'school:stAnna' to key = 'school', type = 'stAnna')
  - node, way and relation IDs were added to tags and used to group the tags according to the node, way or relation they belong to (this ID was used later as FOREIGN KEY in the tag tables)
  - the position in the way they belong to was also added to each way tags*
- then, with the exception of relations that were not included given our focus, following a specific schema, the following element and attributes were released by:
  - the sequence **osm2csv_step1.py -> osm2csv_step2.py**: *nodes* (including 'id', 'lat', 'lon', 'user', 'uid', 'version', 'changeset', and 'timestamp') and *node tags* (including node 'id', 'key', 'value', 'type')
  - **osm2csv_step1.py -> osm2csv_step2b.py**:*ways* (including 'id', 'user', 'uid', 'version', 'changeset', and 'timestamp'), *way nodes* (way 'id', 'ref', 'position'), and *way node tags* (including way 'id', 'key', 'value', 'type')
  - the conformity of the code output to a specific schema was tested on a sample (**fredericton_sample.osm**) using python modules *cerberus* and *schema* (see **schema_validator.py**)
- Illustration of the schema (elements from chicago.osm):
  - node element with its tags: {'node': {'id': 757860928, 'user': 'uboot', 'uid': 26299, 'version': '2', 'lat': 41.9747374, 'lon': -87.6920102, 'timestamp': '2010-07-22T16:16:51Z', 'changeset': 5288876} | 'node_tags': [{'id': 757860928, 'key': 'amenity', 'value': 'fast_food', 'type': 'regular'}, {'id': 757860928, 'key': 'cuisine', 'value': 'sausage', 'type': 'regular'}]}
  - way element with its tags and nodes: {'way': {'id': 209809850, 'user': 'chicago-buildings', 'uid': 674454, 'version': '1', 'timestamp': '2013-03-13T15:58:04Z', 'changeset': 15353317} 'way_nodes': [{'id': 209809850, 'node_id': 2199822281, 'position': 0},b{'id': 209809850, 'node_id': 2199822390, 'position': 1}] | 'way_tags': [{'id': 209809850, 'key': 'housenumber', 'type': 'addr', 'value': '1412'}, {'id': 209809850, 'key': 'street', 'type': 'addr', 'value': 'West Lexington St.'}, {'id': 209809850, 'key': 'street:name', 'type': 'addr', 'value': 'Lexington'}]}

- each type of element (node, way), tag (node tag, way tag), and way tag was written into a csv file (each type had its own file, later converted in its own table). Thus, for each city osm file the final code produced 5 files corresponding to nodes, ways, node tags, way tags, and way nodes.

2.1.2. Assessment of data quality

- City by city, csv data were converted in pandas DataFrames (**durban.py... verona.py**)
  - for instance, durban.py took durban_nodes.csv, durban_nodes_tags.csv, durban_ways.csv, durban_ways_nodes.csv, and durban_ways_tags.csv and returned nodes_df, node_tags_df, node_tags_df, ways_df, ways_tags_df, and way_nodes_df
  - df were checked for the right types in each column looking at DataFrame head (see code in **data_verifier.py**)

2.1.3. Data quality report

As expected, all data appeared to be in the right format and to meet the specifications set during csv creation.

2.2. From csv to SQL database

- while adding df to the database the following error occurred (see below):
  - encoding error: the encoding of data was changed first to 'cp1252', then finally, to 'latin1' as the error still occured in other files
  - integrity error: null values from NON NULL columns were removed (indicating that there were missing values in some NON NULL columns) (code in **null_values_remover.py**)
- then csv files were added to a database named 'the9cities.db':
  - creating first the database, then the 5 tables from each city (see code in **table_creator.py**)
  - then, adding DataFrame data in the right tables (code in **table_csv_appender.py**)
- Errata of multiple entry of the same data were prevented by dropping tables with same name on creation of new tables (**table_creator.py**)
- latitudes and longitudes of interest were extracted and plotted from the database built

2.2.1. Encoding errors

Warning processing Halifax data

~/anaconda2/lib/python2.7/site-packages/IPython/core/interactiveshell.py:2723: DtypeWarning: Columns (4) have mixed types. Specify dtype option on import or set low_memory=False. interactivity=interactivity, compiler=compiler, result=result)

UnicodeDecodeError processing Quebec City data

UnicodeDecodeError: 'charmap' codec can't decode byte 0x81 in position 7: character maps to

UnicodeDecodeError processing Turin data

UnicodeDecodeError: 'charmap' codec can't decode byte 0x8d in position 6: character maps to

Solution:

All solved by setting encoding to 'latin1'

2.2.2. Problems met during data addition to database

First problem met:

failure to get unique identifiers in node/way tags and way nodes, even by combining all parameters.

Tables without PRIMARY KEY :

Introduction of unique indices.

Integrity error processing Nairobi data:

IntegrityError: (sqlite3.IntegrityError) NOT NULL constraint failed

Solution:

1. Detection for missing values in NON NULL fields added to the header_type_verifier function (**null_values_remover.py**)
2. Lines with missing values in NON NULL columns were dropped in any further processings using the null_values_remover custom function (**null_values_remover.py**)

3. Operations using the db

In [1]:
```
# opening database connection
import sqlite3

db = 'the9cities.db'
conn = sqlite3.connect(db)
cursor = conn.cursor()
print "Database {db} opened successfully".format(**vars())
```

```
Database the9cities.db opened successfully
```

For the functions used along the text see **functions_load_first.py**

3.1. Verona - Italy

3.1.1. Amenities

- Amenity (http://wiki.openstreetmap.org/wiki/Key:amenity): Used to map facilities used by visitors and residents. For example: toilets, telephones, banks, pharmacies, cafes, parking and schools.

Statistical overview of the dataset

```
In [3]: city = 'Verona'
        city_node_tag, city_node, city_way_tag, city_way = city_files_informatio
        n(city)
```

```
Verona files' size in MB:
city.osm: 184.6    nodes.csv: 66.1    node_tag.csv: 7.8
ways.csv: 5.2    way_nodes.csv 17.2    way_tags.csv 6.4

The number of nodes in verona_node is: 806207

The number of ways in verona_way is: 91252

The number of unique users in Verona tables is: 534
```

Amenities of interest

```
In [4]: # Amenities of interest and number in the node table
        print amenities_of_interest_viewer(city_node_tag, needed_values)
```

```
amenities of interest in verona_node_tag
       key         value  count
0  amenity        clinic      1
1  amenity       doctors      3
2  amenity  fire_station      1
3  amenity      hospital      3
4  amenity  kindergarten     10
5  amenity       parking     62
6  amenity        police      8
7  amenity    restaurant    194
8  amenity        school     27
```

Observations and comments

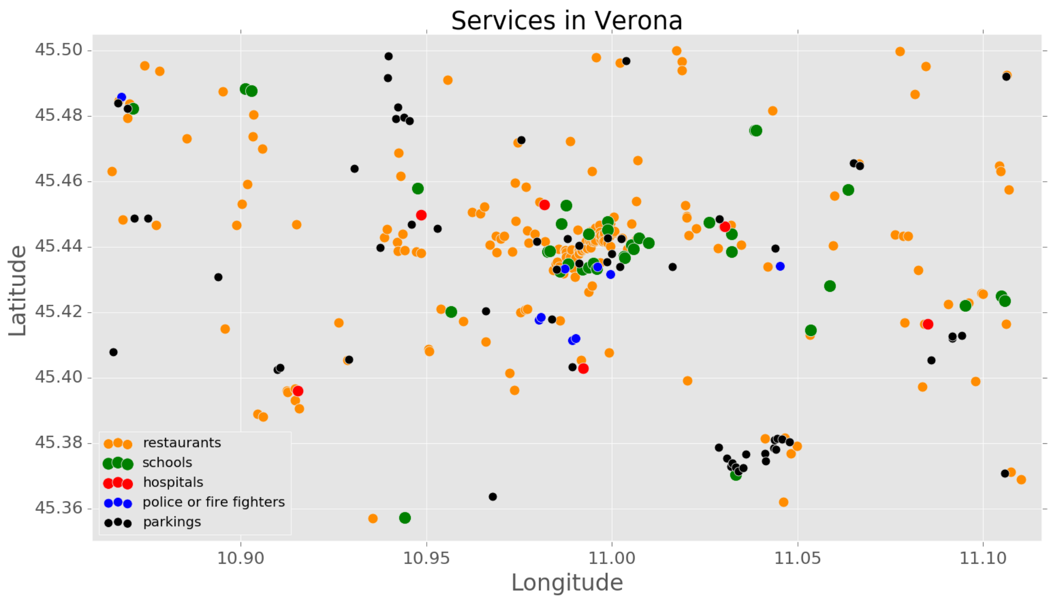
The number and location of major hospitals is accurate. In Italy, hospitals are generally few, but very big and offer mainly specialized and advanced interventions. However, the number of doctors and clinic is highly underestimated as the system uses family doctors (a lot of them) generally grouped according to specialties.

- 62 parkings: this is a lot for a city of this size.. As it was very difficult to find a parking in Verona around 2007 close to mayor elections, people voted a candidate offering to create a lot of parkings all over the city.. And he did!
- A lot of restaurants (194): this is Italy and people like colloquial family-managed restaurants, so there are so many of them (usual average capacity: 20-30 tables)

Location of amenities of interest

In [5]:
```
#plotting the amenities of interest
city_fig_number = 1
legend_position = 'lower left'
x_lim = [10.86, 11.116]
y_lim = [45.35, 45.505]
plotter(city, city_fig_number, legend_position, x_lim, y_lim)
```

```
verona_node_tag and verona_node proceeded

FIGURE 1
```



Observations and comments

The city center is crowded with restaurant.. Normal, considering that it contains a lot of tourist attractions (historic center including a lot of old churches and 6 museums), commercial centers, public library and the largest campus of the University of Verona (medical and computer sciences are in another campus as the same location as the red point under the center/towards South). The center also includes the famous arena of Verona, and almost all public offices.. So a lot of people work there or come to enjoy themselves with their friends or family. Therefore, it is a great place for a commercial activity.
The South-East of the city, close to any of the hospitals, appears as a great location for a home, as there are schools and restaurants around, and less congestion than in the center, where the traffic at rush hours can be infernal... I would also advice the South-East of the city because there are a lot of parks. As they do not have location (latitude, longitude) information (except building complex polygons not of much use now given our focus), let's count the number of parks and related grounds reported in the city of Verona.

3.1.2. Way tags of interest

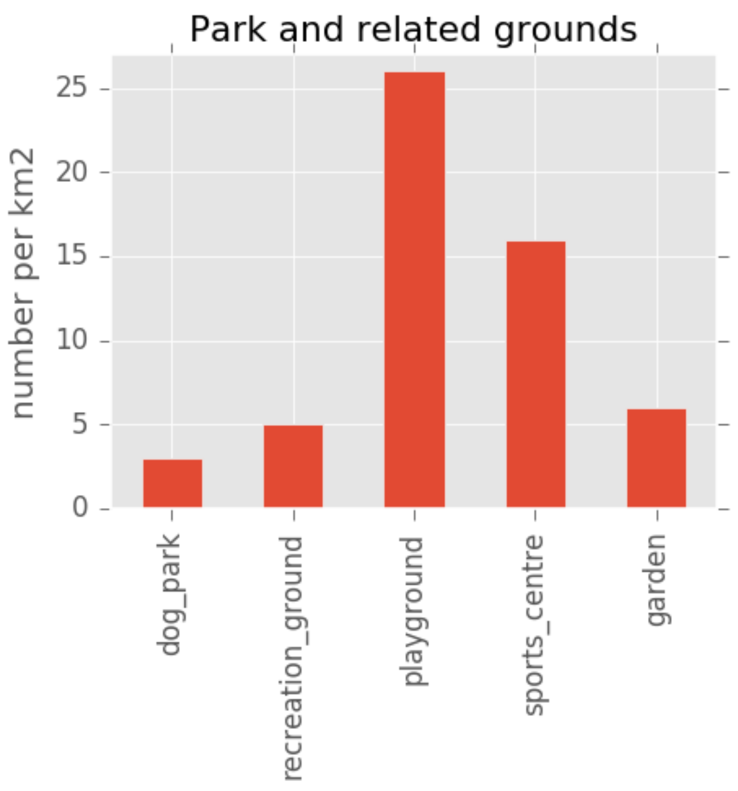
This was a good idea.. The way tag has information on parks, playground, and other recreational places ideal for families!

Notes:

- Landuse (http://wiki.openstreetmap.org/wiki/Landuse)
  - This is used to describe the purpose for which an area of land is being used.
  - e.g. recreation_ground: An open green space for general recreation, which may include pitches, nets and so on, usually municipal but possibly also private to colleges or companies
- Leisure (http://wiki.openstreetmap.org/wiki/Key:leisure)
  - This is used to tag leisure and sports facilities. e.g.:
  - garden: Place where flowers and other plants are grown in a decorative and structured manner or for scientific purposes
  - dog_park: Designated area, with or without a fenced boundary, where dog-owners are permitted to exercise their pets unrestrained

In [6]:
```
# parks and related array
needed_places = ('dog_park', 'recreation_ground', 'playground', 'sports_
centre', 'garden')
# creating the data pd series
pd_series = lands_pd_maker(city, city_way_tag, city_way, needed_places)
# plotting them
city_fig_number = city_fig_number + 1
title = 'Park and related grounds'
ylabel = 'number per km2'
y_lim = [0, 27]
fig_size = [4, 3]
pandas_plotter(pd_series, city_fig_number, title, ylabel, y_lim, fig_siz
e)
```

```
FIGURE 2
```



Observations and comments

This is the city I know! There are a lot of parks, recreation grounds and other great places for families... The verona database really reflects the realities of the city! Let's have a quick look at other cities' (I don't no them much, so I will go straight to the point and just make general observations). Let's start by other Italian and European cities I like: Turin and Lausanne.

3.2. Turin - Italy

Statistical overview of the dataset

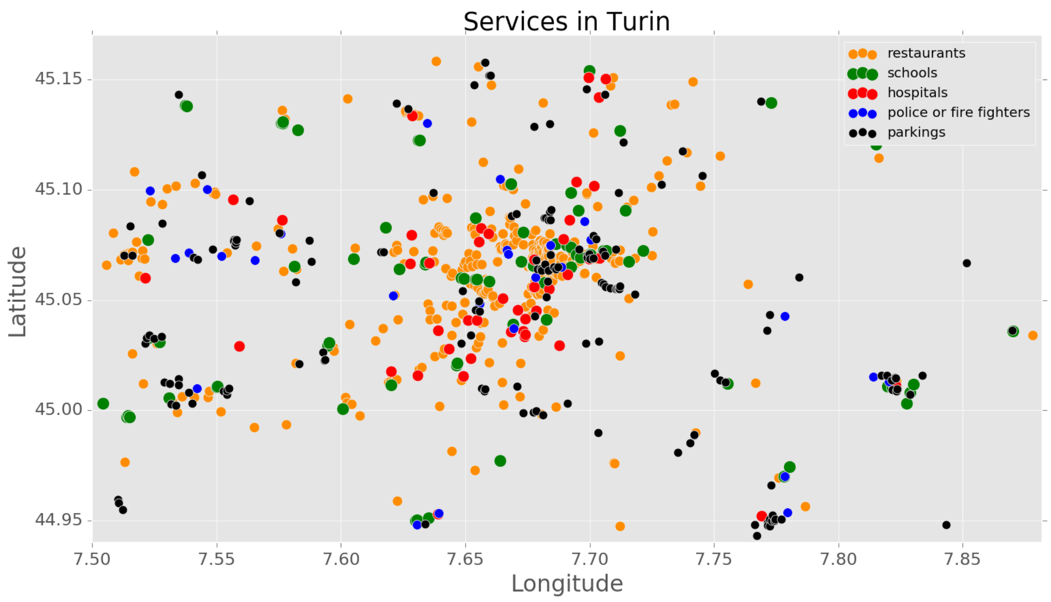
In [7]:
```
city = 'Turin'
city_node_tag, city_node, city_way_tag, city_way = city_files_informatio
n(city)
```

```
Turin files' size in MB:
city.osm: 130    nodes.csv: 47.1    node_tag.csv: 3.7
ways.csv: 5.4    way_nodes.csv 17.2    way_tags.csv 6.5

The number of nodes in turin_node is: 575125

The number of ways in turin_way is: 89119

The number of unique users in Turin tables is: 870
```

Amenities of interest

In [8]:
```
#plotting the amenities of interest
city_fig_number = city_fig_number + 1
legend_position = 'upper right'
x_lim = [7.5, 7.882]
y_lim = [44.94, 45.17]
plotter(city, city_fig_number, legend_position, x_lim, y_lim)
```

```
turn_node_tag and turin_node proceeded

FIGURE 3
```



Observations and comments

There are a lot of great locations for a home close to the center of the city. If someone wants to avoid living close to the center, according to our amenities of interest, there are also great locations close to hospitals in either the North-Eastern or the South-Eastern parts of the city.

3.3. Lausanne - Switzerland

Statistical overview of the dataset

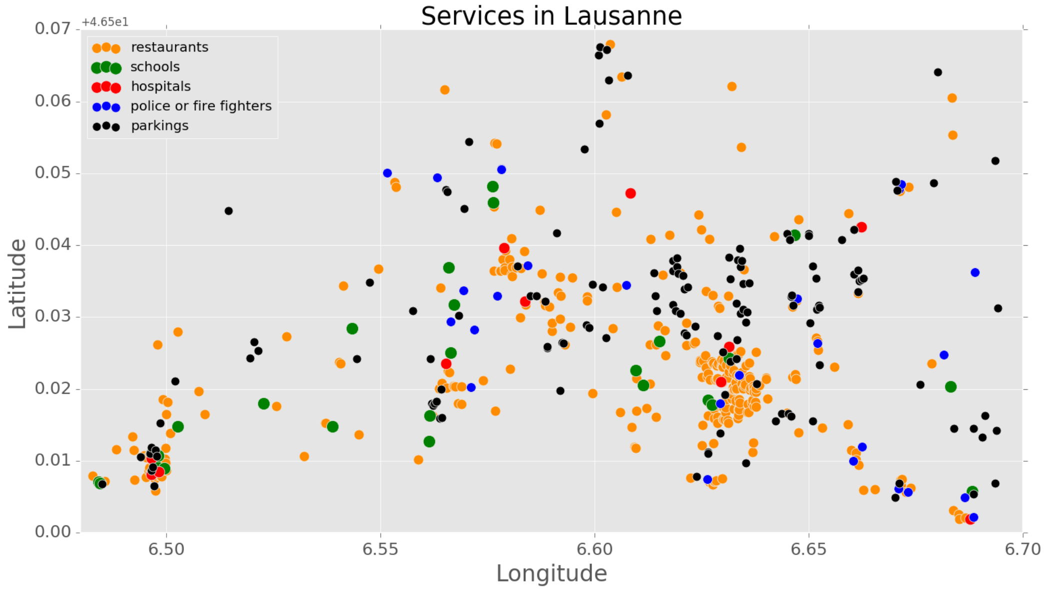
In [9]:
```
city = 'Lausanne'
city_node_tag, city_node, city_way_tag, city_way = city_files_informatio
n(city)
```

```
Lausanne files' size in MB:
city.osm: 114.1    nodes.csv: 40.3    node_tag.csv: 2.9
ways.csv: 4.9    way_nodes.csv 13.9    way_tags.csv 9.1

The number of nodes in lausanne_node is: 481200

The number of ways in lausanne_way is: 80120

The number of unique users in Lausanne tables is: 686
```

Amenities of interest

```
In [10]: #plotting the amenities of interest
         city_fig_number = city_fig_number + 1
         legend_position =  'upper left'
         x_lim = [6.48, 6.70]
         y_lim = [46.50, 46.57]
         plotter(city, city_fig_number, legend_position, x_lim, y_lim)
```

```
lausanne_node_tag and lausanne_node proceeded

FIGURE 4
```



Observations and comments

The center seems crowded... Two great areas for a home are probably close to the hospitals in the North-West and in the South-West (however, there may not be a police station near by)... They are probably in front of the famous lake Leman, which is a huge touristic attraction with stupendous public transportation and a lot of actitivies all along the year. Let's go towards South-East: to Africa.

3.4. Nairobi - Kenya

Statistical overview of the dataset

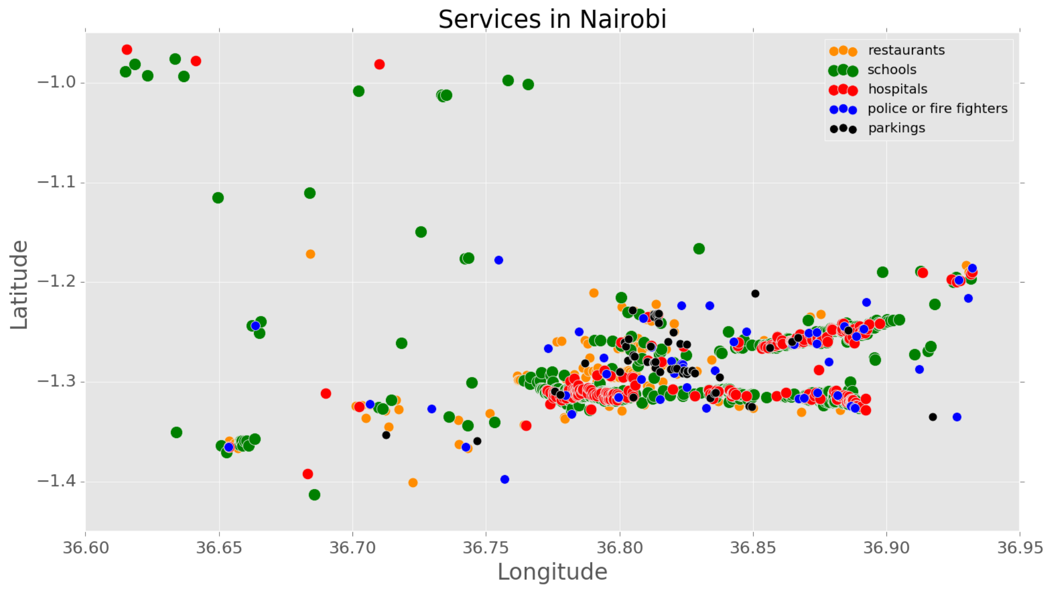
```
In [11]: city = 'Nairobi'
         city_node_tag, city_node, city_way_tag, city_way = city_files_informatio
         n(city)
```

```
Nairobi files' size in MB:
city.osm: 75.6    nodes.csv: 29.0    node_tag.csv: 2.7
ways.csv: 3.0    way_nodes.csv 9.7    way_tags.csv 2.3

The number of nodes in nairobi_node is: 349396

The number of ways in nairobi_way is: 49136

The number of unique users in Nairobi tables is: 824
```

Amenities of interest

```
In [12]: #plotting the amenities of interest
         city_fig_number = city_fig_number + 1
         legend_position =  'upper right'
         x_lim = [36.6, 36.95]
         y_lim = [-1.45, -0.95]
         plotter(city, city_fig_number, legend_position, x_lim, y_lim)
```

```
nairobi_node_tag and nairobi_node proceeded

FIGURE 5
```



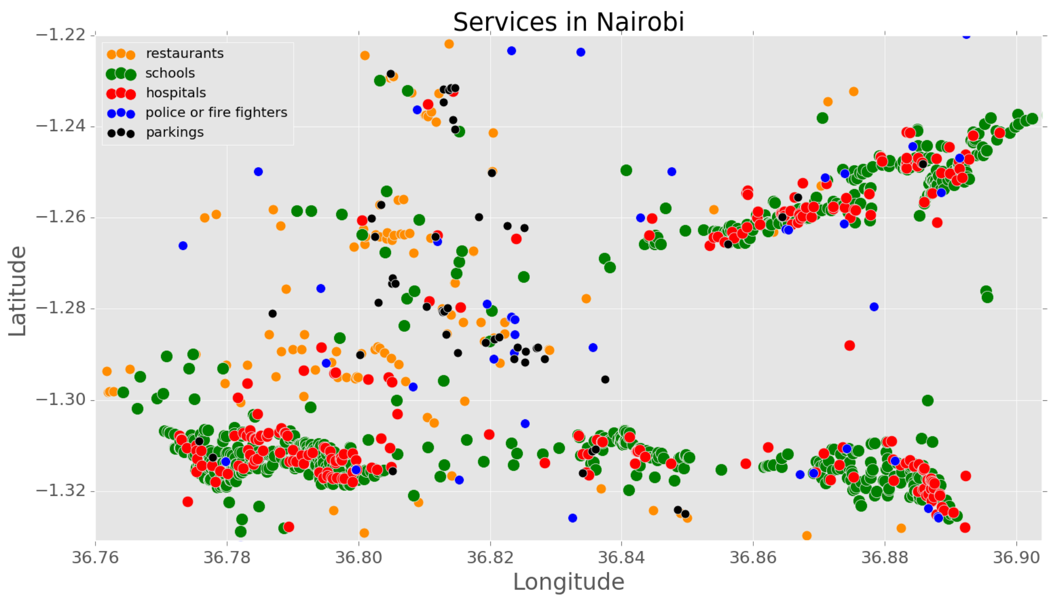
Observations and comments

The city is extended... Almost all the amenities of interest seem concentrated in the South-Eastern part of the city, where almost all companies and offices are located (except the airport that is out of the city, 1h by car). Let's zoom the crowded area.

Amenities of interest in the center of Nairobi

In [13]:
```
#plotting the amenities of interest
city_fig_number = city_fig_number + 1
legend_position = 'upper left'
x_lim = [36.76, 36.904]
y_lim = [-1.331, -1.22]
plotter(city, city_fig_number, legend_position, x_lim, y_lim)
```

```
nairobi_node_tag and nairobi_node proceeded

FIGURE 6
```



Observations and comments

Zooming the central area, it appears that there are 3 major poles, with very few amenities of interest in the central area, where instead, most parkings are located. This is because people go to the city center only for work or shopping.. Driving through Nairobi center at rush hours is one of my worst memories.
The 3 major poles appear as great places for a home, with a lot of schools and hospitals... They are also close to the center.

3.5. Durban - South Africa

Statistical overview of the dataset

In [14]:
```
city = 'Durban'
city_node_tag, city_node, city_way_tag, city_way = city_files_informatio
n(city)
```

```
Durban files' size in MB:
city.osm: 73.2    nodes.csv: 28.5    node_tag.csv: 1.4
ways.csv: 2.0    way_nodes.csv 9.2    way_tags.csv 3.6

The number of nodes in durban_node is: 351216

The number of ways in durban_way is: 34163

The number of unique users in Durban tables is: 220
```

Observations and comments

Considering the city size and population (2,292 km², 3.5 million), the nodes are under reported (only 220 users), so not enough to discuss home location.

3.6. Saint Louis - Senegal

Statistical overview of the dataset

In [15]:
```
city = 'Saint Louis'
city_node_tag, city_node, city_way_tag, city_way = city_files_informatio
n(city)
```

```
Saint Louis files' size in MB:
city.osm: 32.9    nodes.csv: 12.5    node_tag.csv: 0.139
ways.csv: 1.5    way_nodes.csv 4.5    way_tags.csv 1.6

The number of nodes in st_louis_node is: 150999

The number of ways in st_louis_way is: 24604

The number of unique users in Saint Louis tables is: 167
```

Observations and comments

Only 167 users and 150999 nodes.. Saint Louis nodes are also under-reported. Let's go towards North-West: to Canada.

3.7. Fredericton - Canada

Statistical overview of the dataset

In [16]:
```
city = 'Fredericton'
city_node_tag, city_node, city_way_tag, city_way = city_files_informatio
n(city)
```

```
Fredericton files' size in MB:
city.osm: 32.5    nodes.csv: 11.9    node_tag.csv: 0.254
ways.csv: 1.2    way_nodes.csv 4.3    way_tags.csv 2.5

The number of nodes in fredericton_node is: 144890

The number of ways in fredericton_way is: 21424

The number of unique users in Fredericton tables is: 129
```

Observations and comments

Unfortunately, in Fredericton too amenities the number of users is low and reported nodes do not suffice to discuss home location.

3.8. Quebec City - Canada

Statistical overview of the dataset

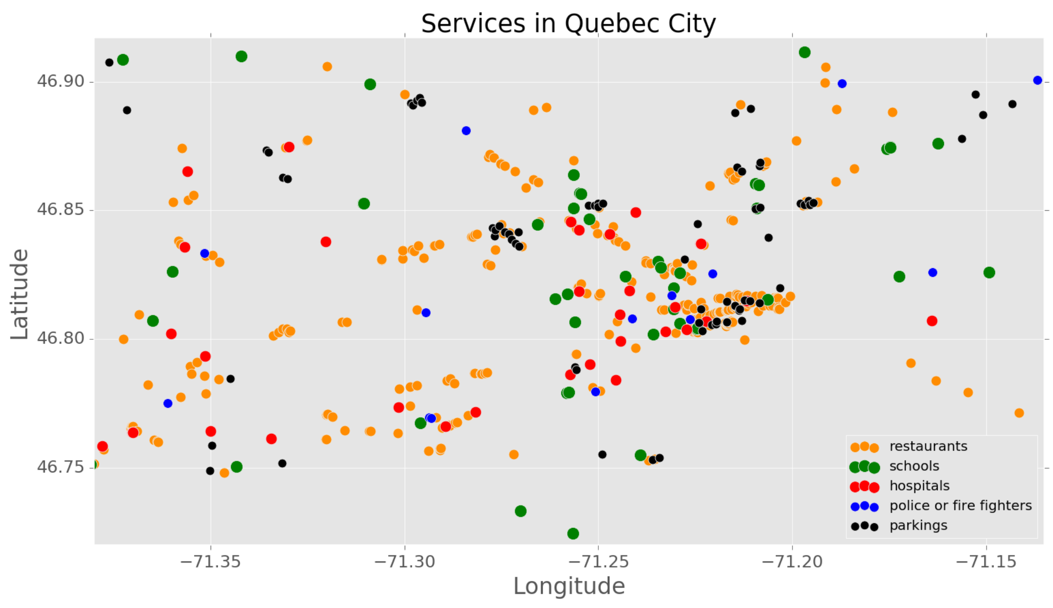
In [17]:
```
city = 'Quebec City'
city_node_tag, city_node, city_way_tag, city_way = city_files_informatio
n(city)
```

```
Quebec City files' size in MB:
city.osm: 115.3    nodes.csv: 39.5    node_tag.csv: 11.2
ways.csv: 4.7    way_nodes.csv 12.9    way_tags.csv 6.3

The number of nodes in quebec_node is: 466636

The number of ways in quebec_way is: 74394

The number of unique users in Quebec City tables is: 501
```

Amenities of interest

In [18]:
```
#plotting the amenities of interest
city_fig_number = city_fig_number + 1
legend_position =  'lower right'
x_lim = [-71.38, -71.135]
y_lim = [46.72, 46.917]
plotter(city, city_fig_number, legend_position, x_lim, y_lim)
```

```
quebec_node_tag and quebec_node proceeded

FIGURE 7
```



Observations and comments

Based on this plot, there are numerous good places ideal for establishing a home, particularly close to hospitals in the South-West and around the center of the city (going towards the North).

3.9. Halifax - Canada

Statistical overview of the dataset

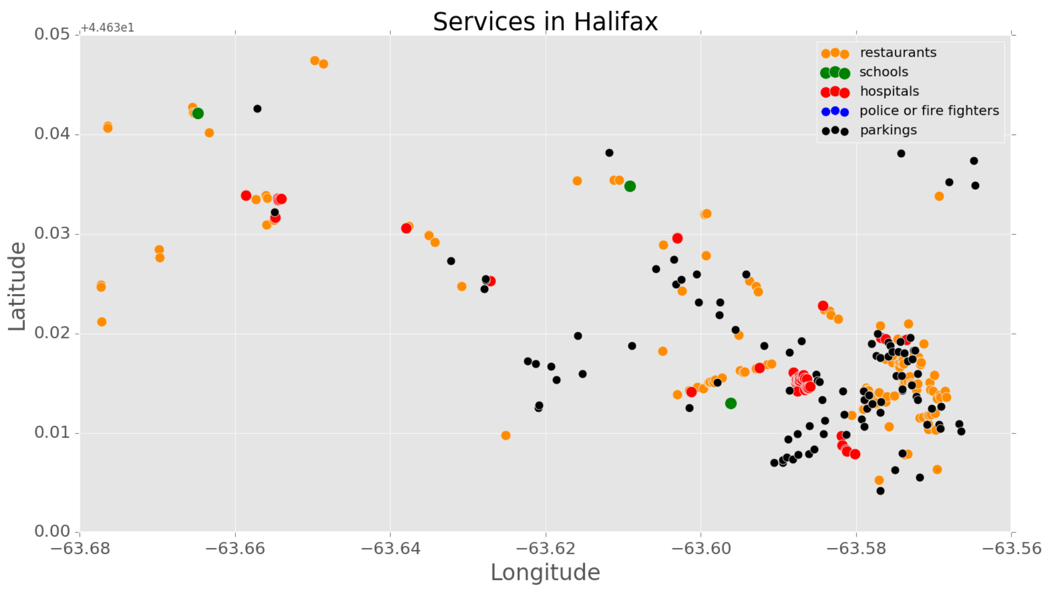
```
In [19]: city = 'Halifax'
         city_node_tag, city_node, city_way_tag, city_way = city_files_informatio
         n(city)
```

```
Halifax files' size in MB:
city.osm: 83.1    nodes.csv: 32.3    node_tag.csv: 3.7
ways.csv: 2.1    way_nodes.csv 10.3    way_tags.csv 3.0

The number of nodes in halifax_node is: 391802

The number of ways in halifax_way is: 35864

The number of unique users in Halifax tables is: 367
```

Amenities of interest

```
In [20]: #plotting the amenities of interest
         city_fig_number = city_fig_number + 1
         legend_position =  'upper right'
         x_lim = [-63.68, -63.56]
         y_lim = [44.63, 44.68]
         plotter(city, city_fig_number, legend_position, x_lim, y_lim)
```

```
halifax_node_tag and halifax_node proceeded

FIGURE 8
```



Observations and comments

Here too, as in Europe, there are a lot of restaurant (193) and parkings (152). Note the clusters of hospital in the South-East. although schools are probably under-reported, areas around the hospitals could be good locations for a home.

4. Summary of cities with enough data

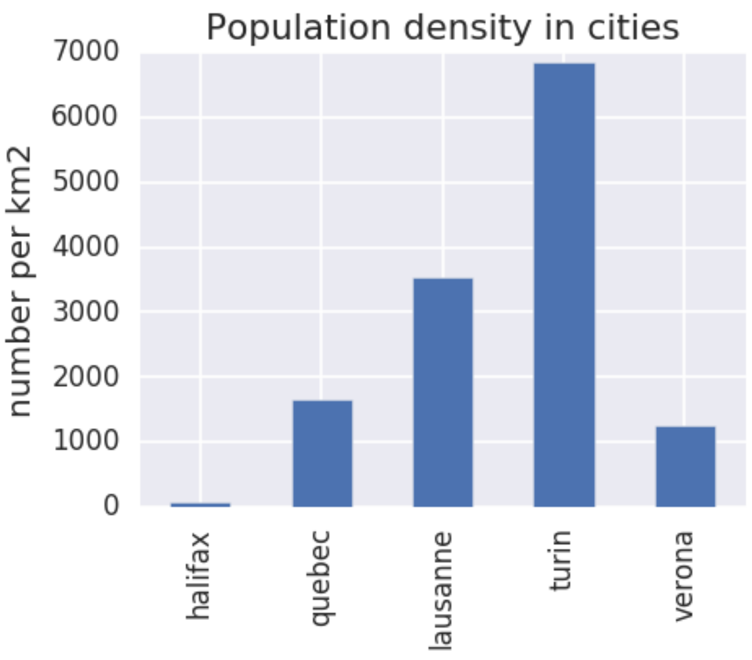
For code creating the pd DataFrame plotted see **getting_city_summary.py**

4.1. Population density

As shown above (section 1.2) the complementary information on population and surface areas was retrieved from the internet.

```
In [22]: import seaborn as sns
         # Plotting the Dataframe of population density
         city_fig_number = city_fig_number + 1
         title = 'Population density in cities'
         ylabel = 'number per km2'
         y_lim = [0, 7000]
         fig_size = [4, 3]
         pandas_plotter(population_pd_series, city_fig_number, title, ylabel, y_l
         im, fig_size)
```

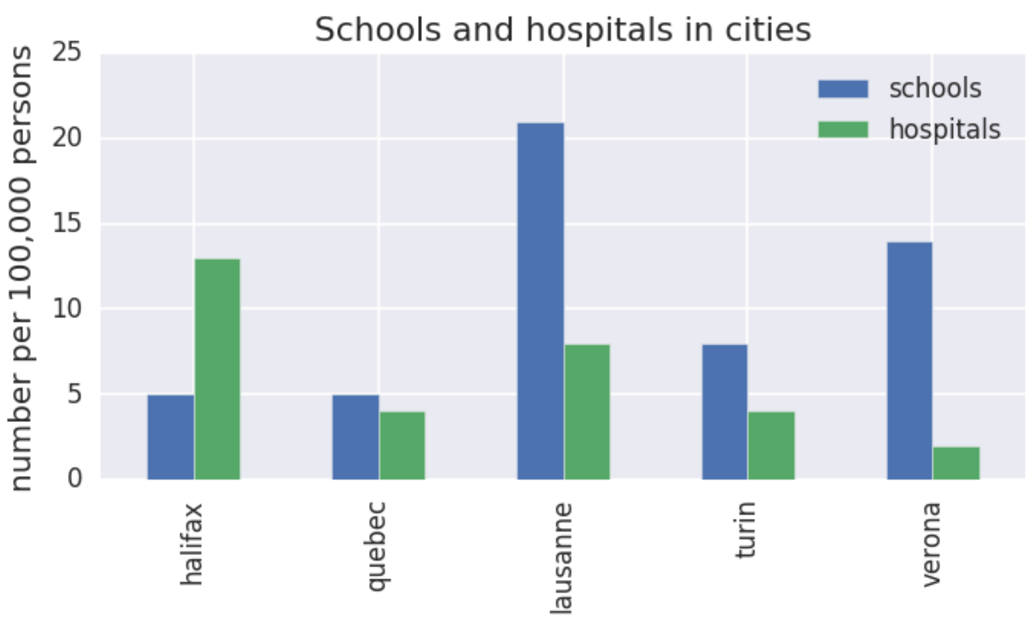
```
FIGURE 9
```



4.2. Number of schools and hospitals

In [23]:
```
# Plotting the relative number of schools and hospitals per city
city_fig_number = city_fig_number + 1
title = 'Schools and hospitals in cities'
ylabel = 'number per 100,000 persons'
y_lim = [0, 25]
fig_size = [6.5, 3]
pandas_plotter(all_cities_df, city_fig_number, title, ylabel, y_lim, fig
_size)
```

```
FIGURE 10
```



Observations and comments

Turin would have almost two fold the density of Lausanne, and the remaining city would be almost two fold smaller than Lausanne, with the exception of Halifax that would be less than 1/10th of the density of Verona. However, Lausanne may have the higher number of shools and halifax the higher number of hospitals, per inhabitants. Verona would have the second highest number of schools per inhabitants' number.

In [24]:
```
# closing the database
conn.close()
print "{db} is closed".format(**vars())
```

```
the9cities.db is closed
```

5. Concluding remarks: benefits, problems and suggestions

The OpenStreetMap project provides access to a lot of informations about a city, and can really have a lot of interesting and helpful applications for people planning to visit or move to a given city. I downloaded osm files containing data of many cities of our choice, easily and without any constraint. Then, I wrangled the data, obtaining neat and clean datasets that were converted to the csv format, then stored in an SQL database.

- A major challenge was the collection of elements attributes from the osm files and their conversion into csv format. The osm format (and xml in general) is great for compressing large data. Thus opening osm files like milan.osm, 694 MB (not shown), I found myself dealing with hundred millions of values stored transiently in the computer memory (in Milan case the memory use was higher than 6 GB). After a test speed for each step of the code, I improved it mainly processing data line by line (and not the whole bulk in the memory as before) and by using less iterations and more regular expressions. It also helped to validate only a sample of the data. Furthermore, dumping temporary results into csv files and re-using them until the production of the final csv files, among other divide-to-conquer and speed improving approaches in python partly conceived after reading this (https://www.airpair.com/python/posts/top-mistakes-python-big-data-analytics#3-mistake-2-not-tuning-for-performance), helped as well. All together, these changes allowed an efficient use of multiprocessing (it was possible then to process more IDE sessions and files at the same time, using up to 12 threads and 8 GB of memory). The conversions useless for 3 days when using a unique file and data validation on the whole thing were now completed in minutes or hours according to the size of the osm file, and it was possible to convert up to 8 files concomitantly.

The OpenStreetMap is a great project and user-provided information seems to allow accurate mapping of cities (at least for Verona and other cities I have visited which were processed in this study). However, here are some small small issues that to my opinion may hamper the project:

- In many cases, osm files of many cities do not have enough information, and nodes, ways and relations are under-reported. The beneficial potential of the OpenStreetMap project could be further improved with more people getting involved. Thus, advertising the project more and simplifying the mapping procedure (http://wiki.openstreetmap.org/wiki/Tags) may attract considerably more people and improve the mapping where needed. For instance, adding a piece of code recognizing automatically the key of a node from its value (e.g. for a value = 'hospital', automatically add key = 'amenity') while adding it. This will save the users from overbrowsing through the potential keys (http://wiki.openstreetmap.org/wiki/TagFinder) to use for a given value, considering also that these keys are subject to updates (https://taginfo.openstreetmap.org/). A possible drawback of a massive arrival of people into the project could be an increase in human errors in the city files. This may be partly solved by appointing "city supervisors" among the project participants. With administrator privileges, these supervisors would be in charge to ensure that data in the file of their city is accurate.
- The operational definition of concepts, which can be really confusing. Notably, the difference between terms like "doctors" for many doctors sharing a building for their private practices, "clinic" (for health institutions offering outpatient procedures), and 'hospital' for health institutions offering inpatient services) can really be confusing, because in French culture for instance 'clinic' ('clinique') usually indicates private health institutions. From country to country health system is different. So in Italy, for instance, it can really be puzzling to determine where to classify a building shared by family doctors. Deep culture-related understanding of concepts may also explain the comparably lower number of restaurants in the African cities' osm files proceeded (e.g. 173 only in a large city like Nairobi). In Sub-Saharan African culture, when it comes to have a meal out of their homes most people go to "gargotes" (small restaurant where food is good and relatively cheap). Although they offer up to 50 tables (in some cities even more), for most people they do not meet the standard to be called "restaurants", as the latter term is usually used only for VIP places where food is expensive.

The time spent wrangling these data, the couple of days spent to come out with a code fast and efficient enough to proceed the data in a relatively short time, and the week spent preparing this project were worth it because I really improved my python programming skills and I really enjoyed the virtual tour of these cities.

References

See **references.txt**