Problems Encountered in the Data

1. **Street names**

   There were some differences within the street names across the data provided in the Open Street Map data. Out of the 1122 street names provided in the sample file, 987 of the street names are one word long. When examining street names that were longer than one word, the two most common words by far that was at the end of the name were **väg** and **gata**. In Swedish, these words mean road and street respectively.

   In the data, there were streets that ended with both **Väg** and **väg**. After some research, I found the convention is for these words not to be capitalized, so the street names were updated appropriately prior to being uploaded into MongoDB.

2. **Buildings**

   In the dataset, the most common tag is building. In the sample dataset, there were 2879 buildings, and for 1799 of them, the value is "yes". Unlike other values for the building tag like "house" or "industrial", a value of "yes" provides no insight into the building. Before uploading the data into MongoDB, this value was changed to "building".

3. **Shaping the data**

   There were some tags that had two words separated by a colon. Usually, the second word was a description of the first word. For example, "building:floors" referred to the number of floors in the building. Often times, an element had multiple of these fields with the same first word. Therefore, to more efficiently group these values, I grouped all fields with a colon and the same first word into a dictionary.

   For example, for an element with the following tags:

```
<tag k="building" v="house" />
<tag k="roof:shape" v="gabled" />
<tag k="building:colour" v="red" />
<tag k="building:levels" v="1" />
<tag k="building:material" v="wood" />
```

   the relevant part of the document would look like:

```
"building" : {
        "building" : "house",
        "colour" : "red",
        "material" : "wood",
        "levels" : "1"
}
```

   These updates were made prior to uploading the data into MongoDB.

4. **City Names**

After uploading the data into MongoDB, I inspected the city names. I ran the following query:

```
> db.Stockholm.aggregate(
  [
    {"$match":{"address.city":{"$exists":1}}},
    {"$group":{"_id":"$address.city", "count":{"$sum":1}}}, {"$sort":{"count":-1}}
  ]
)
```

This returns documents like:

```
{ "_id" : "31", "count" : 1 }
{ "_id" : "BROMMA", "count" : 1 }
{ "_id" : "Väddö", "count" : 1 }
```

I found the document in the database that had a city of 31:

```
> db.Stockholm.find({"address.city":"31"}, {"address":1}).pretty()
{
        "_id" : ObjectId("55b51f1d272b081aa8bbc24c"),
        "address" : {
                "city" : "31",
                "street" : "Överbyvägen",
                "housenumber" : "31",
                "country" : "SE"
        }
}
```

The house number was mistakenly uploaded as the city name as well. After a little research in the database, I found which city this was located:

```
>db.Stockholm.find({"address.street":'\u00D6verbyv\u00E4gen'},
{"address.city":1}).pretty()
```

Every document on this street was in the city of Vaxholm, so I set this document to be in Vaxholm as well:

```
> db.Stockholm.update({"address.city" : "31"},
                              {"$set" : {"address.city": 'Vaxholm'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Also, in the example above, there were some cities that had unusual formats. For example, **BROMMA** should be **Bromma**. I created a mapping dictionary of all the incorrectly named cities and updated them in a similar fashion programmatically.

After the data was first uploaded into MongoDB, there were 163 distinct cities. After cleaning, there were 140.

Data Overview

File Sizes

Filename	Size (GB)
stockholm_sweden.osm	1.06
Stockholm_sweden.osm.json	1.23

Number of Documents

```
> db.Stockholm.find().count()
5733662
```

Number of Nodes

```
>  db.Stockholm.find({"type" : "node"}).count()
5155852
```

Number of Ways

```
> db.Stockholm.find({"type" : "way"}).count()
577128
```

Number of Distinct Users

```
> db.Stockholm.distinct("created.user").count()
2099
```

Top 10 Contributing Users

```
> db.Stockholm.aggregate(
        [
         {"$group":{"_id":"$created.user", "count":{"$sum":1}}},
         {"$sort":{"count":-1}},
         {"$limit": 10}
        ])
{ "_id" : "MichaelCollinson", "count" : 590698 }
{ "_id" : "Fringillus", "count" : 463743 }
{ "_id" : "emj", "count" : 352431 }
{ "_id" : "jordgubbe", "count" : 276728 }
{ "_id" : "huven", "count" : 233542 }
{ "_id" : "Tooga", "count" : 195118 }
{ "_id" : "TheOddOne2", "count" : 179111 }
{ "_id" : "Zorac", "count" : 170270 }
{ "_id" : "Snusmumriken", "count" : 142770 }
{ "_id" : "StellanL", "count" : 130821 }
```

These users contribute 47.7% of the data

Additional Exploration

Most Popular Amenities

```
> db.Stockholm.aggregate(
        [
         {"$match":{"amenity" : {"$exists":1}}},
         {"$group":{"_id":"$amenity", "count":{"$sum":1}}},
         {"$sort":{"count":-1}},
         {"$limit": 10}
        ])
{ "_id" : "parking", "count" : 10527 }
{ "_id" : "restaurant", "count" : 2052 }
{ "_id" : "bench", "count" : 1249 }
{ "_id" : "school", "count" : 1214 }
{ "_id" : "cafe", "count" : 1098 }
{ "_id" : "fast_food", "count" : 1059 }
{ "_id" : "kindergarten", "count" : 747 }
{ "_id" : "recycling", "count" : 738 }
{ "_id" : "post_box", "count" : 737 }
{ "_id" : "shelter", "count" : 658 }
```

Location Data

I tried to see if any of the documents were outside of the bounds listed in the XML file.

```
> db.Stockholm.find(
        {
                '$or': [{"pos.1": {'$gt': 19.526}}, {"pos.1": {'$lt': 16.858}}]
        }).count()
0

> db.Stockholm.find(
        {
                '$or': [{"pos.0": {'$gt': 60.249}}, {"pos.0": {'$lt': 58.718}}]
        }).count()
0
```

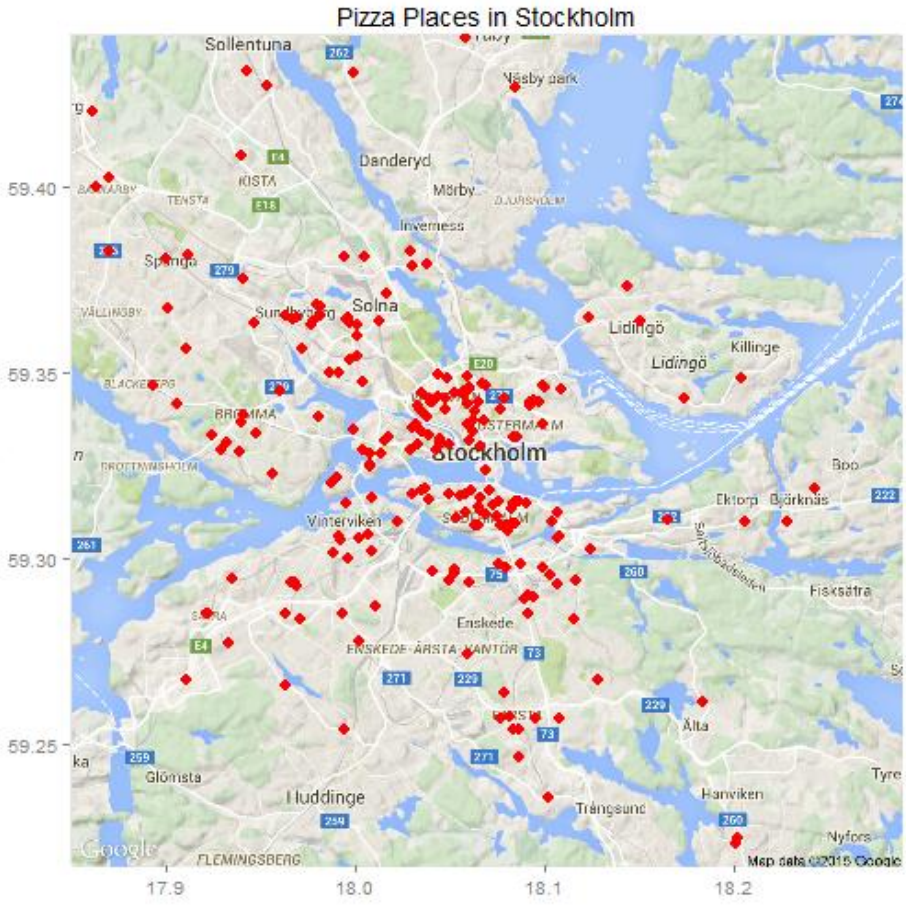
All the data is within the bounds. I also wanted to see how many documents had no positions listed.

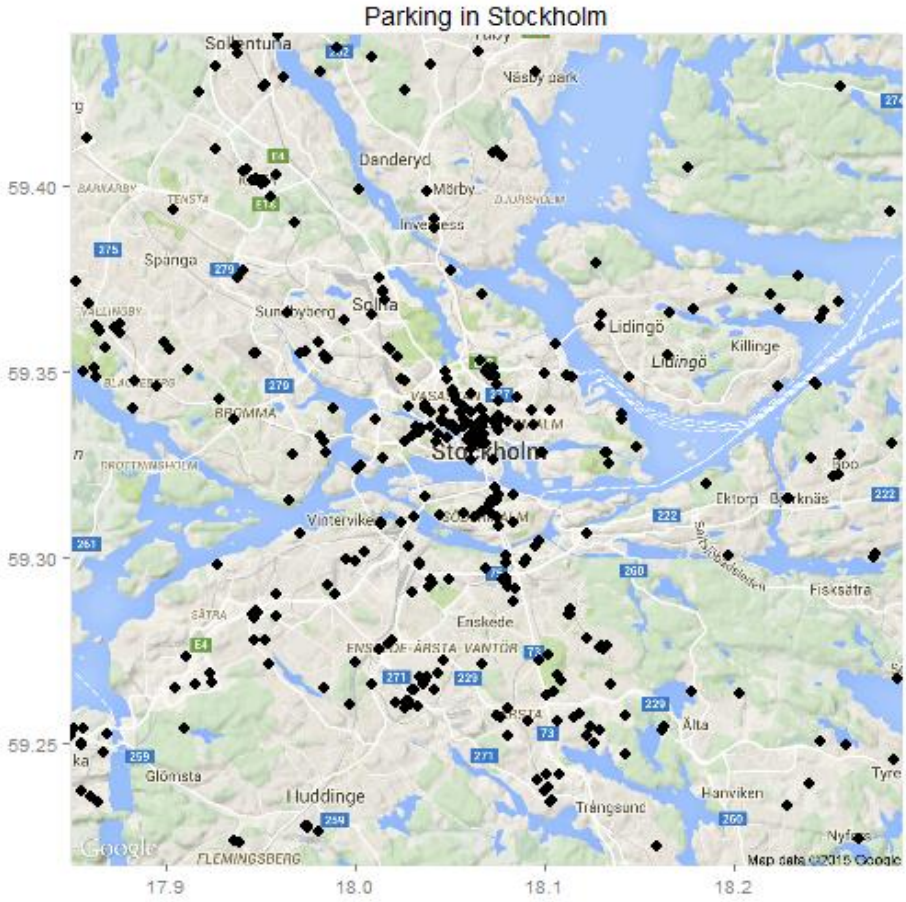
```
> db.Stockholm.find({"pos" : []}).count()
577517
```

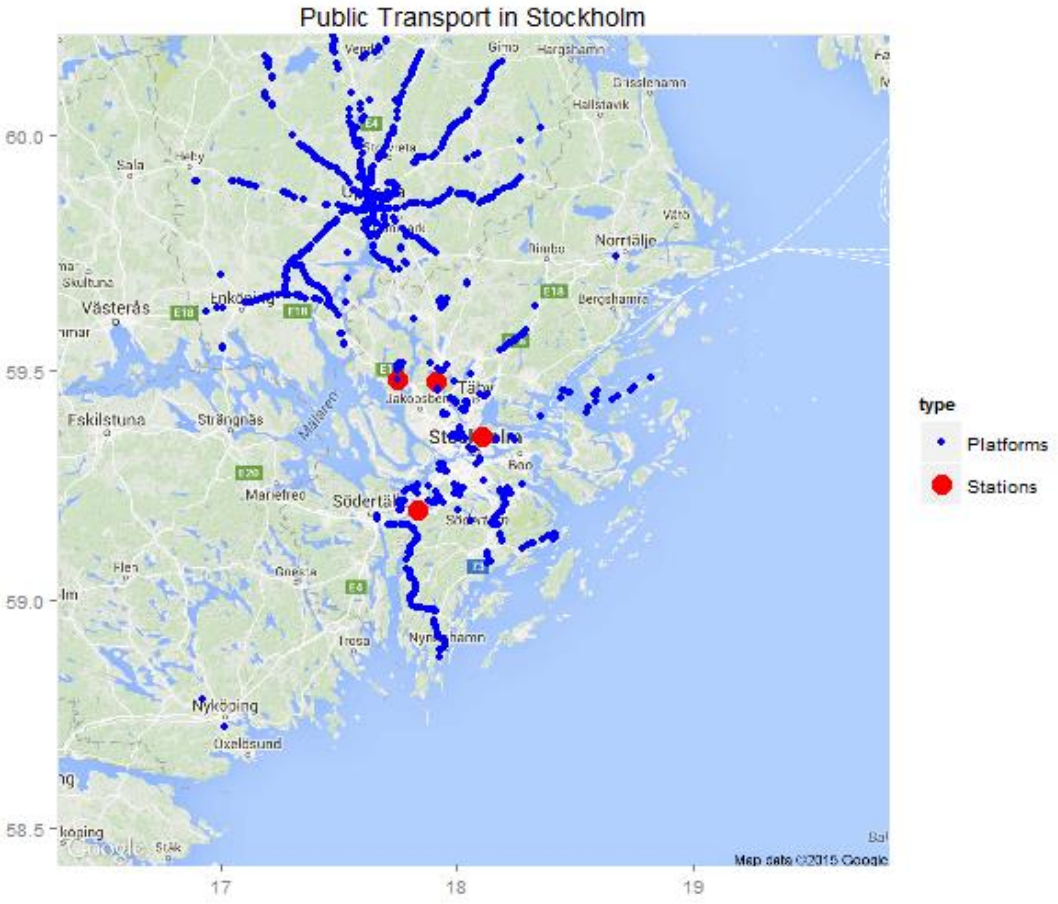
Around 10% of documents are missing position data.

Visualization

The following plots were plotted using the ggmap package in R. The positions of each point were retrieved from the database. The maps are zoomed in to different levels so the visuals would convey the most information. Additionally, some documents didn't contain any position data. These documents were omitted from the visuals.




Parking in Stockholm
59.40
59.35
59.30
59.25
17.9
18.0
18.1
18.2
Solentuna
Näsby park
Danderyd
Mörby
Spånga
Sollna
Lidingö
Killinge
Stockholm
Vinterviken
Enskede
Älta
Glömsta
Huddinge
Trångsund
Hanviken
Fisksätra
Google
Map data ©2015 Google



It looks like Uppsala has a really good public transportation system.

Other Ideas

There is a lot more information that can be gathered from this dataset. Here are some more ideas:

- We can verify that the city and zipcode is accurate for all the entries.
- We can find which users input the best quality of data, and which ones don't.
- We can structure the data a little better. There are still some subdocuments which have fields with colons. These can be further separated into subdocuments. A notable example are the seamark subdocuments which have 3 or 4 colons.

  Also, most of the data is in Unicode. For some fields, it might be useful to have the data in other formats like int or datetime. One great feature about MongoDB is that the schema can evolve over time. So, if someone was trying to do data exploration on a field where they want dates, they can update the documents fairly easily and continue with their exploration.
- Visualization provides a great view of the data. This can be used for city planning purposes, so people can easily see where different amenities are located in relation to

each other. For example, we can see if there are any cemeteries located close to schools.