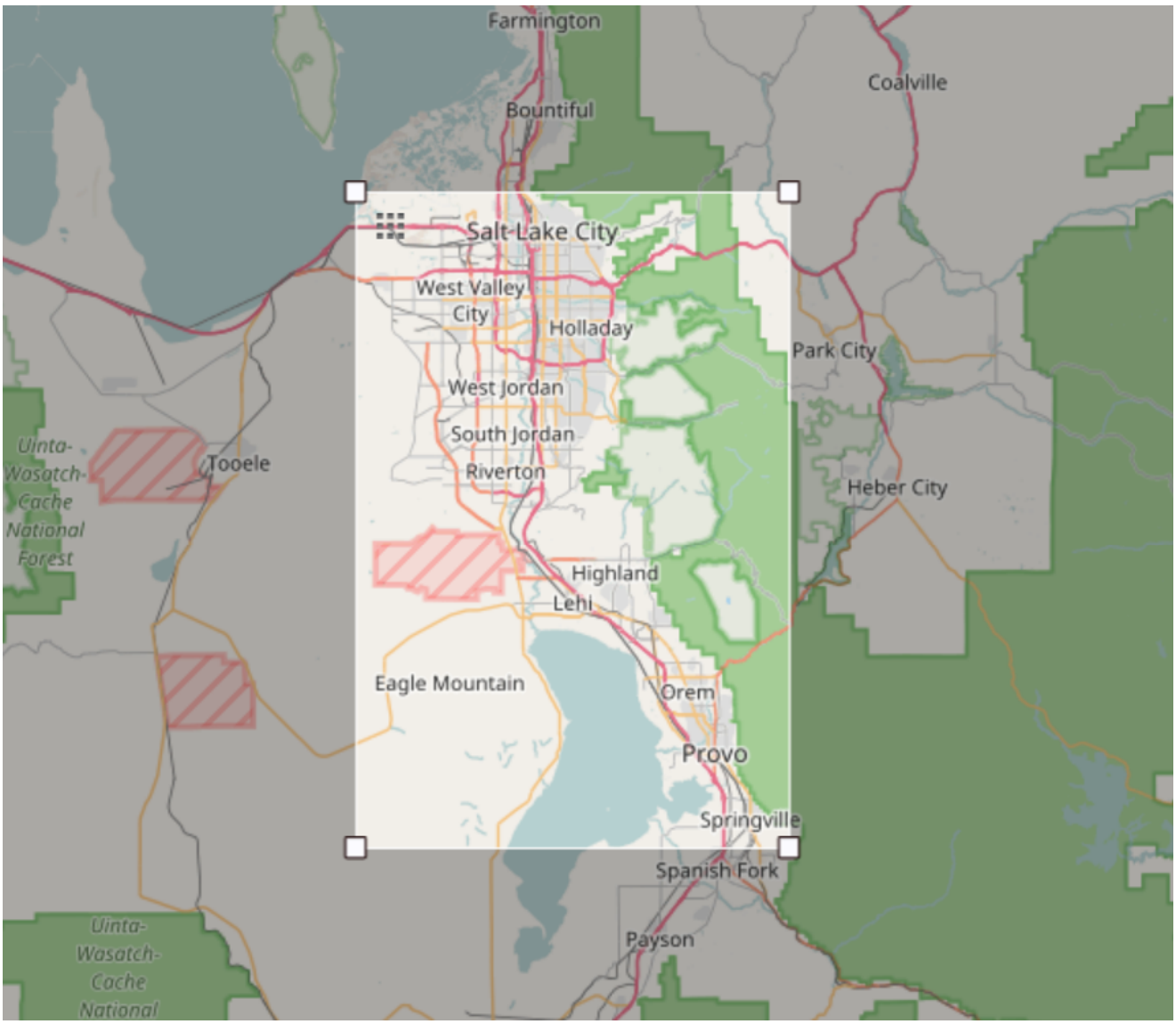
Wrangling Utah OpenStreetMap Data

Author: Daniel Pipkin

Map Area: Spans from North Salt Lake to Springville Utah

Salt Lake county and Utah county are the two largest counties in Utah, where I live. I thought it might be fun to check it out.



1. Problems Encountered in the Map

I followed a very similar process to the OpenStreetMap case study to get the data I wanted into MongoDB. After I inserted the collection, I used pipeline aggregation to clean the data further. These are a couple of things from the fields I was interested in.

County

Most county names show up as ", UT." But when I looked at all the unique values, I noticed some that looked like "Juab, UT:Millard, UT:, Utah, UT." I took each of those county fields and split them into arrays.

I also noticed that "Lincoln, LA" showed up in one "way." I looked it up on OpenStreeMap, found the actual county, and changed it. Here's the link for reference: http://www.openstreetmap.org/way/10147843

The county field was called "tiger:county" so I renamed it to "county" because I thought it looked better.

Building

Using the same trick as above, I looked at the distinct values for "building" and I found some had values of "no." I looked at pictures of these locations to try to see what happened. Here are two examples.

As you can see, those are definitely buildings. I changed all of the "building: no" entries to "building: yes."

Churches

Because I was interested in looking at different denominations, I wanted to make sure those were clean. There was one denomination that stuck in particular.

```
> db.with_churches.distinct('denomination')
[
        "mormon",
        "baptist",
        "methodist",
        "lutheran",
        "pentecostal",
        "foursquare",
        "catholic",
        "latter_day_saints,_mormon",
        "anglican",
        "jehovahs_witness",
        "presbyterian",
        "greek_orthodox",
        "seventh_day_adventist",
        "salvation_army",
        "scientist",
        "LDS",
        "roman_catholic",
        "latter_day_saints",
        "mormom",
        "united_methodist",
        "orthodox",
        "apostolic",
        "mormons;mormon",
        "lds"
]
```

`mormon`, `mormom`, `mormons;mormon`, `LDS`, `lds`, `latter_day_saints,_mormon`, and `latter_day_saints` all mean the same thing so I normalized the names using a MongoDB query:

```
> db.with_churches.updateMany({'denomination': {$in: ['mormon', 'mormom', 'mormons;mormon', 'LDS',
'lds', 'latter_day_saints,_mormon']}}, {$set: {'denomination': 'latter_day_saints'}})
```

2. Data Overview

File sizes

```
utah.osm ......... 208,716 KiB
utah.osm.json .... 208,717 KiB
```

I inserted the elements into MongoDB as I parsed and used `mongoexport` to create the JSON file, which explains the only 1 KiB difference. For a million documents, I don't think that's too bad.

Number of documents

```
> db.cleaned.find().count()
1032382
```

Number of nodes

```
> db.cleaned.find({'type': 'node'}).count()
918374
```

Number of ways

```
> db.cleaned.find({'type': 'way'}).count()
114008
```

Number of unique users

```
> db.cleaned.distinct('created.user').length
1022
```

Number of buildings

```
> db.cleaned.find({'building': {$exists: 1}}).count()
40000
```

Top building types

```
> db.cleaned.aggregate([
        {$match: {'building': {$exists: 1}}},
        {$group: {'_id': '$building', 'count': {'$sum': 1}}},
        {$sort: {count: -1}},
        {$limit: 5}
])
{ "_id" : "yes", "count" : 32850 }
{ "_id" : "house", "count" : 4317 }
{ "_id" : "apartments", "count" : 668 }
{ "_id" : "commercial", "count" : 385 }
{ "_id" : "church", "count" : 243 }
```

3. Other ideas about the dataset

Improvements

As seen in the query just above, 81% of buildings are just labeled with `yes` instead of the building type. This sample actually mirrors the overall percentage of all building tags; 81% are labeled `yes`. The OpenStreetMap wiki says to only use the `yes` value "where it is not possible to determine a more specific value."[1] I *highly* doubt that more than 80% of buildings can't be classified with a more specific value. Even getting that percentage down to 50% would be useful for looking at things like city zoning and densities of industrial or residential building types.

There isn't a clear programmatic way to fix these values, so I think crowdsourcing would be a great solution. When a user logs in to OSM during the campaign, a window would pop up showing how close the community is to the tagging goal and how much an average user contributes per day. Most people like to carry their weight and don't like being outdone, so showing that daily user average is key.

Potential Benefits

- More detailed data to work with.
- Increased community engagement that could spill over into future data cleaning campaigns.

Anticipated Issues

- Human error— If we just trusted one person to enter the right answer, there would be so many human errors. One way to get around that is to have multiple people work on the same buildings and compare their work. After about five people, there should be an apparent consensus.

More queries

These are just some things I found interesting.

Top counties

```
> db.cleaned.aggregate([
        {'$unwind': '$county'},
        {'$group': {'_id': '$county', 'count': {'$sum': 1}}},
        {'$sort': {'count': -1}}
])
{ "_id" : "Salt Lake, UT", "count" : 17638 }
{ "_id" : "Utah, UT", "count" : 10234 }
{ "_id" : "Summit, UT", "count" : 185 }
{ "_id" : "Wasatch, UT", "count" : 23 }
{ "_id" : "Morgan, UT", "count" : 4 }
{ "_id" : "Millard, UT", "count" : 2 }
{ "_id" : "Juab, UT", "count" : 2 }
{ "_id" : "Davis, UT", "count" : 2 }
```

Top amenities

```
> db.cleaned.aggregate([
        {$group: {'_id': '$amenity', count: {$sum: 1}}},
        {$sort: {count: -1}},
        {$limit: 5}
])
{ "_id" : null, "count" : 1024496 }
{ "_id" : "parking", "count" : 2491 }
{ "_id" : "place_of_worship", "count" : 1105 }
{ "_id" : "restaurant", "count" : 1033 }
{ "_id" : "school", "count" : 550 }
```

Top churches

```
> db.with_churches.aggregate([
        {$group: {'_id': '$denomination', count: {$sum: 1}}},
        {$sort: {count: -1}},
        {$limit: 10}
])
{ "_id" : "latter_day_saints", "count" : 855 }
{ "_id" : null, "count" : 139 }
{ "_id" : "baptist", "count" : 31 }
{ "_id" : "catholic", "count" : 17 }
{ "_id" : "lutheran", "count" : 12 }
{ "_id" : "jehovahs_witness", "count" : 10 }
{ "_id" : "presbyterian", "count" : 8 }
{ "_id" : "methodist", "count" : 8 }
{ "_id" : "anglican", "count" : 5 }
{ "_id" : "foursquare", "count" : 5 }
```

Conclusion

OpenStreetMap has a lot of data not only entered by users, but also defined by users in a lot of cases. Auditing one aspect of the dataset at a time, like done above, and fixing errors seems like the best approach. I think if OpenStreetMap developed auditing tools for the community, they could crowd-source the cleaning of their data.

Footnotes

1. http://wiki.openstreetmap.org/wiki/Map_Features#Other_Buildings