A preliminary analysis of the Bicincitta data

Vishal Sood

May 8, 2015

Contents

0.1 Problems in the Bicincitta data set from 2013

There are problems with the Bicincitta data that we need to address before loading the data into a reliable and proper data-base. We will point out these problems using examples, and measure their magnitude using systematic analysis, and then speculate about the cause behind these problems.

0.1.1 Data

We load the data from JSONs provided to us by Bicincitta at the end of April 2015.

The resulting data is in the form of lists of dictionaries.

```
a subnetwork is described by,
        id
        name

 a station is described by,
        latitude
        name
        id
        longitude
        subnetwork_id

 a user is described by
        subnetwork_id
        gender
        expires
        postal_code
        address
        id

 a transaction is described by,
        direction
        user_id
        event_time
        created_at
```

```
updated_at
station_id
id
```

The resulting dictionaries have ids that are UTF-8 strings. We change these to integers to make our work easier. The addersses in the user data are web-quotes, which we need to *unquote*. We will also unquote the station names, just to be safe. There are keys in a transaction that do not seem to correspond to the data, but refer to the time at which the data was loaded into the JSON provided to us.We will drop these variables, and change the *event_time* to a time object.

0.1.2 Who are the users?

The simplest question may be the fraction of females vs males,

```
Of all the users  60  percent are female  and  39  percent males.
```

It would be interesting if 60% of the users were in fact female. However, as we will see later there seems to be a problem of user duplicacy biased towards females.

0.1.3 Subnetworks for stations users, and transactions

We have a data table for subnetworks, which contains the subnetwork's id and name. Both users and stations have been assigned a *subnetwork_id* which should be an integer pointing to the *id* variable in the subnetwork table. We would expect all the subnetworks in this table to be conceptually equivalent. Thus the subnetworks *PubliBike* and *Campus* should refer to the same concept of a subnetwork. However a peek at the data hints against this assumption. **It seems that there are two distinct concepts of a subnetwork in the subnetwork table.** There are several lines of evidence leading us to this conclusion.

First of all, only 11 of the 18 subnetworks have a station assigned to them, and 7 have no stations (see table in the appendix).

While none of the stations have been assigned the subnetwork, most of the users are in subnetwork PubliBike,

```
Number of users from subnetwork PubliBike is 58927
```

The subnetworks thus appear to mean two different things:

1. a geographic subnetwork that has stations
2. an administrative subnetwork that is assigned to a user when she signs up.

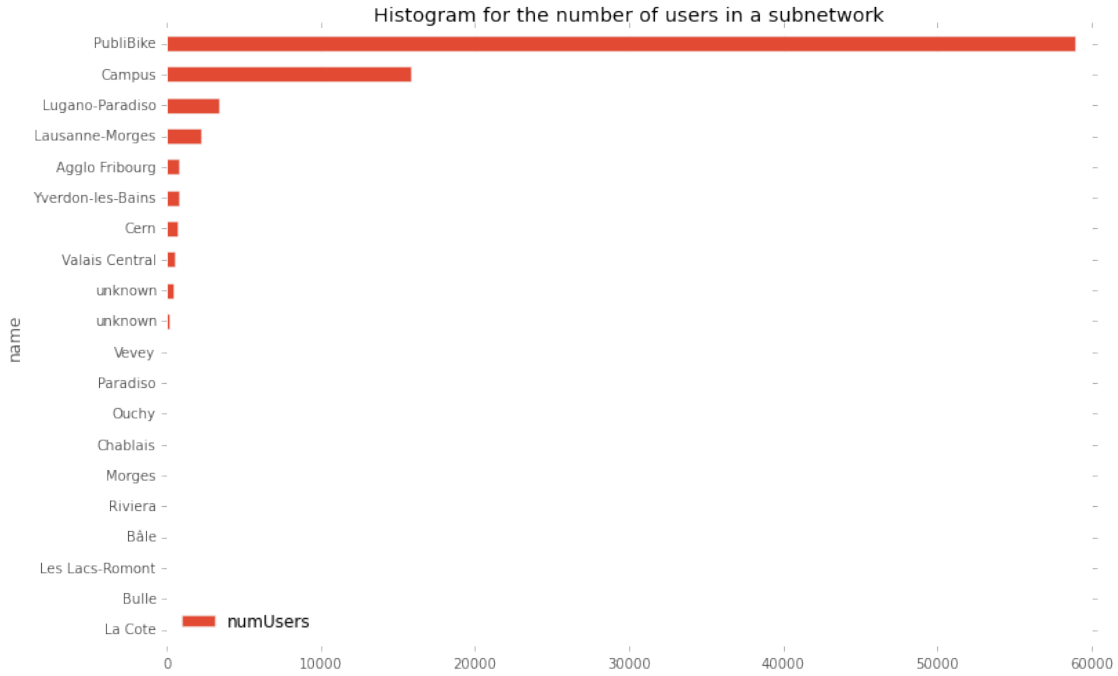
So we assign each transaction an administrative and a geographic subnetwork. The administrative subnetwork is the subnetwork that the user of the transaction has been assigned, and the geographic subnetwork is the subnetwork that the station of the transaction has been assigned. So administratively, all the transactions are in PubliBike, while geographically they are in 11 different subnetworks. In the appendix we present a table for subnetworks, showing the number of transactions that fall in a subnetwork, both administratively and geographically, along with the number of users.

Looking at the actual number of users who have registered a transaction makes us question the validity of the user data base.

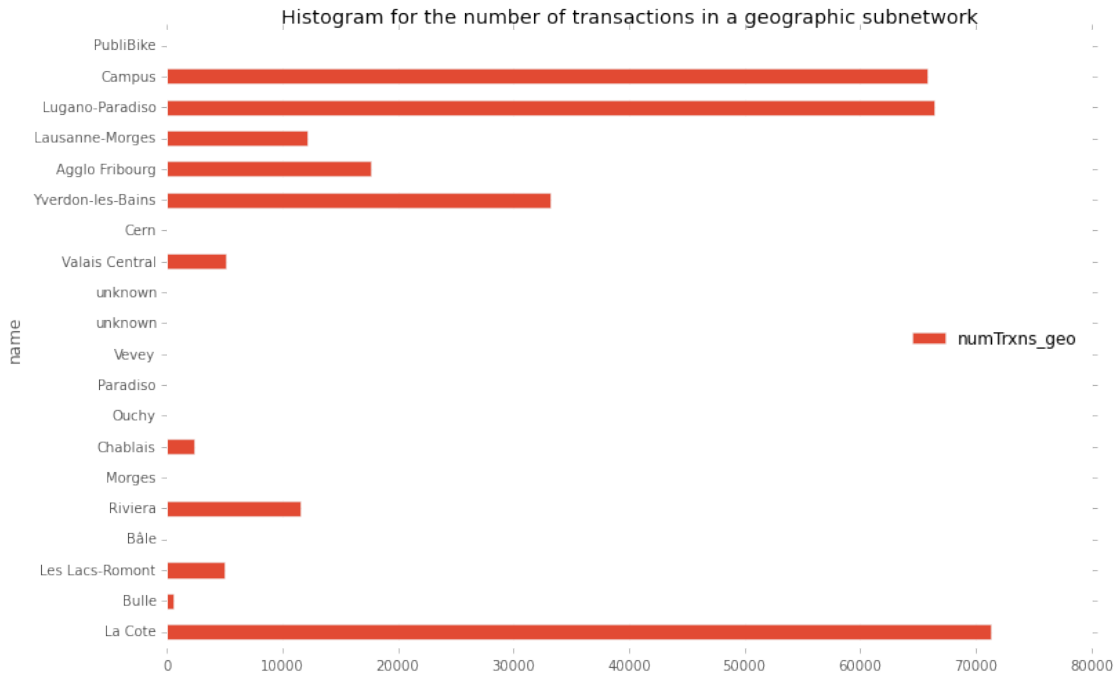
```
Fraction of users who have registered a transaction 0.10673152586
```

With only 10% users with registered transactions, where are the remaining 90% users from ? Are they left-overs from previous versions of the system? Or is there an error in the database? Additionally, all the users with the transactions have been assigned the subnetwork *PubliBike*.

```
Out[126]: <matplotlib.axes._subplots.AxesSubplot at 0x11a0ec790>
```


Histogram for the number of users in a subnetwork
PubliBike
Campus
Lugano-Paradiso
Lausanne-Morges
Agglo Fribourg
Yverdon-les-Bains
Cern
Valais Central
unknown
unknown
Vevey
Paradiso
Ouchy
Chablais
Morges
Riviera
Bâle
Les Lacs-Romont
Bulle
La Cote
name
numUsers
0
10000
20000
30000
40000
50000
60000


Histogram for the number of transactions in an administrative subnetwork
PubliBike
Campus
Lugano-Paradiso
Lausanne-Morges
Agglo Fribourg
Yverdon-les-Bains
Cern
Valais Central
unknown
unknown
Vevey
Paradiso
Ouchy
Chablais
Morges
Riviera
Bâle
Les Lacs-Romont
Bulle
La Cote
name
numTrxns_admin
0
50000
100000
150000
200000
250000
300000



0.1.4 User addresses

There are several problems associated with user addresses. We have already noticed, and fixed, that the provided addresses in the JSON have not been *unquoted* from their web encoding. Here we continue to explore other problems that may arise in the addresses.

We want to count the number of users at one address. Because the addresses have been provided as strings, we have to be able to aggregate all address strings that describe the same address. We have written a python function to do this task, which takes the address and postal-code strings to provide a combined string taking into account some empirical disambiguation criteria such as *Av, Ave*, for *Avenue*.

Addresses are not available for all the users.

```
Number of users with available address 21659  of which only  15446  unique
```

What fraction of unique addresses have multiple users?

```
0.230545124951
```

How many users at addresses with multiple users?

```
9774
```

which corresponds to a fraction of all users with available address,

```
0.451267371531
```

Multiple users at the same address could be actual multiple people, or multiple registrations by the same person, or a database error. We can consider as an example the address with the most multiplicity of 53,

```
address:  via lambertenghi 1; 6900 , number of users:  53
```

We could say more about the multiple users at the same address if we look at their transactions. However as it turns out, we **do not have addresses for users who have registered transactions in the data**,

There are as many as 37 users assigned to the same address that also have more than subnetwork assigned. Addresses with several users might represent problems of multiple subscription. For example, if we look at addresses with more than 10 users,

we see that the user is over-whelmingly females. However, a look at the lower end of such addresses seems alright,

Out[807]:

	address	numFemales	numMales	numUsers	\
3	poudrière 24; 1950	3	0	3	
61	avenue beaulieu 20; 1004	2	1	3	
23	avenue louis-ruchonnet 31; 1003	2	1	3	
104	eichenweg 12; 1718	2	1	3	
67	rue saint-rochemin 5; 1004	2	1	3	

	subnetworks
3	set([Campus, Valais Central])
61	set([Lausanne-Morges, Campus])
23	set([Lausanne-Morges, Campus])
104	set([Agglo Fribourg, Campus])
67	set([Lausanne-Morges, Campus])

These particular addresses appear sensible. There could be more than one person living at these addresses who have signed up with the bike system, albeit in different subnetworks. Or may be it is the same person with 2 different sign-ups in two different sub-networks. This raises the question: **How are users registered by the system? One individual = one signup? Or does a user need a sign-up for each subnetwork that she wants to use?** If it is the latter, then the provided *user_ids* become less useful, because the same individual will appear as different users according to the *user_ids*.

Out[808]:

	address	numFemales	numMales	numUsers
18	via lambertenghi 1; 6900	52	1	53
2288	chemin des falaises 3; 1005	52	0	52
1349	chemin des berges 12; 1022	41	0	41
2150	avenue des bains 9; 1007	37	0	37
332	via monte carmen 4; 6900	33	0	33
287	route cantonale 33; 1025	25	0	25
1444	via madonnetta 23; 6900	24	0	24
1649	place du tunnel 17; 1005	23	0	23
1826	avenue des bains 11; 1007	23	0	23
1997	rue de genève 76; 1004	22	0	22
1801	route cantonale 35; 1025	22	0	22
2534	via zurigo 1; 6900	20	1	21

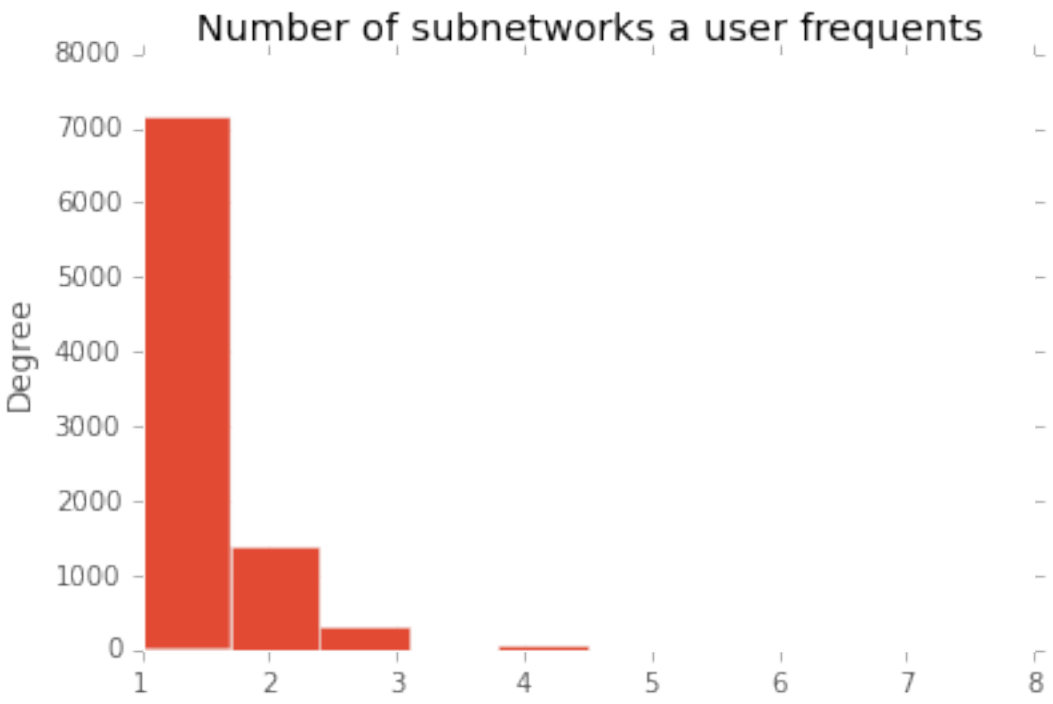
Out[809]:

	address	numFemales	numMales	numUsers
0	bonne-espérance 28; 1006	1	0	1
1	37 route cantonnale; 1025	1	0	1
2	avenue de la dôle 4; 1005	1	0	1
3	abbesses 21; 2012	1	0	1
4	chemin de ponfilet 100; 1093	0	1	1

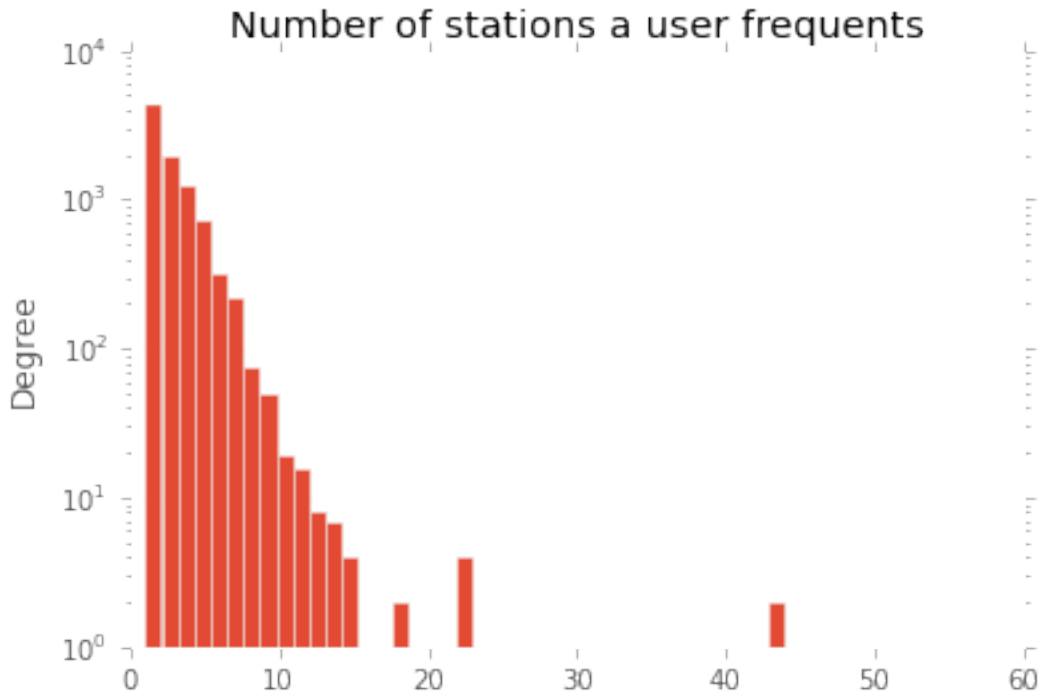
0.1.5 Network Usage

In addition to how many transactions, users for each subnetwork, we can look at the number of stations and geographic subnetwork that a user uses.

Out[77]: <matplotlib.axes._subplots.AxesSubplot at 0x113969410>



Out[78]: <matplotlib.axes._subplots.AxesSubplot at 0x115d75750>



0.2 Appendix

Table characterizing subnetworks

Out[76]:

name	id	name	numStations	numTrxns_admin	numTrxns_geo	numUsers
La Cote	1	La Cote	13	0	71292	0
Bulle	3	Bulle	2	0	566	0
Les Lacs-Romont	4	Les Lacs-Romont	9	0	4964	0
Bâle	5	Bâle	0	0	0	0
Riviera	11	Riviera	5	0	11576	0
Morges	15	Morges	0	0	0	0
Chablais	6	Chablais	10	0	2377	3
Ouchy	16	Ouchy	0	0	0	3
Paradiso	17	Paradiso	0	0	0	3
Vevey	14	Vevey	0	0	0	4
unknown	2011	unknown	0	0	0	152
unknown	2005	unknown	0	0	0	469
Valais Central	7	Valais Central	7	0	5077	530
Cern	18	Cern	0	0	0	721
Yverdon-les-Bains	8	Yverdon-les-Bains	9	0	33227	773
Agglo Fribourg	2	Agglo Fribourg	10	0	17630	810
Lausanne-Morges	9	Lausanne-Morges	11	0	12157	2183
Lugano-Paradiso	12	Lugano-Paradiso	13	0	66415	3425
Campus	10	Campus	15	0	65853	15871
PubliBike	13	PubliBike	0	291134	0	58927