

# SBB

Aug 11, 2019

In the [previous post](#) the network consisted of only 25 nodes. Who needs computers, right? [SBB](#), the Swiss Federal Railways, definitely do. In fact, they maintain an amazing data repository comprised of over sixty [datasets](#) with many thousand entries each. The company even launched a [Kaggle](#) competition in 2018 because there was no standard software to cover their increasing needs.

Here I would like to show the bigger picture without going into task-specific problem solving. It is about connecting the dots, making a living organism out of the network, beyond coordinate systems.

## Basel SBB



```
%matplotlib inline
import os
import numpy as np
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
from matplotlib import cm
```

## Import Data

The original dataset contains information about the yearly total number of trains that pass through each route section of Switzerland. The list contains entries for 2016, 2017 and 2018. Distinction is also made between passenger transport and cargo. For this post, I will focus

only on passenger transport in 2018. Comparative studies along the aforementioned two axes are reserved for the near future.

```
df = pd.read_csv('data/zugzahlen.csv')
```

PID	Anzahl_Zuege
SBB_GESE_CHY	126307
SBB_GIBU_ROL	97943
SBB_MIES_TAN	126101
SBB_MOR_STJ	121566
SBB_PER_ALL	98005

## Preprocessing

As is often the case with transportation networks, there are origin and destination points. Caution 🧠! These two sets of points do not have to be congruent and in this case they aren't. Let's extract the respective longitudes and latitudes and have a look.

### Longitude of origin

```
df['lon_von'] = df.geopos_von.str.split('\,').str[0]  
df['lon_von'] = df['lon_von'].map(float)
```

### Latitude of origin

```
df['lat_von'] = df.geopos_von.str.split('\,').str[1]  
df['lat_von'] = df['lat_von'].map(float)
```

### Longitude of destination

```
df['lon_bis'] = df.geopos_bis.str.split('\,').str[0]  
df['lon_bis'] = df['lon_bis'].map(float)
```

### Latitude of destination

```
df['lat_bis'] = df.geopos_bis.str.split('\,').str[1]  
df['lat_bis'] = df['lat_bis'].map(float)
```



```
G = nx.Graph(D)
```

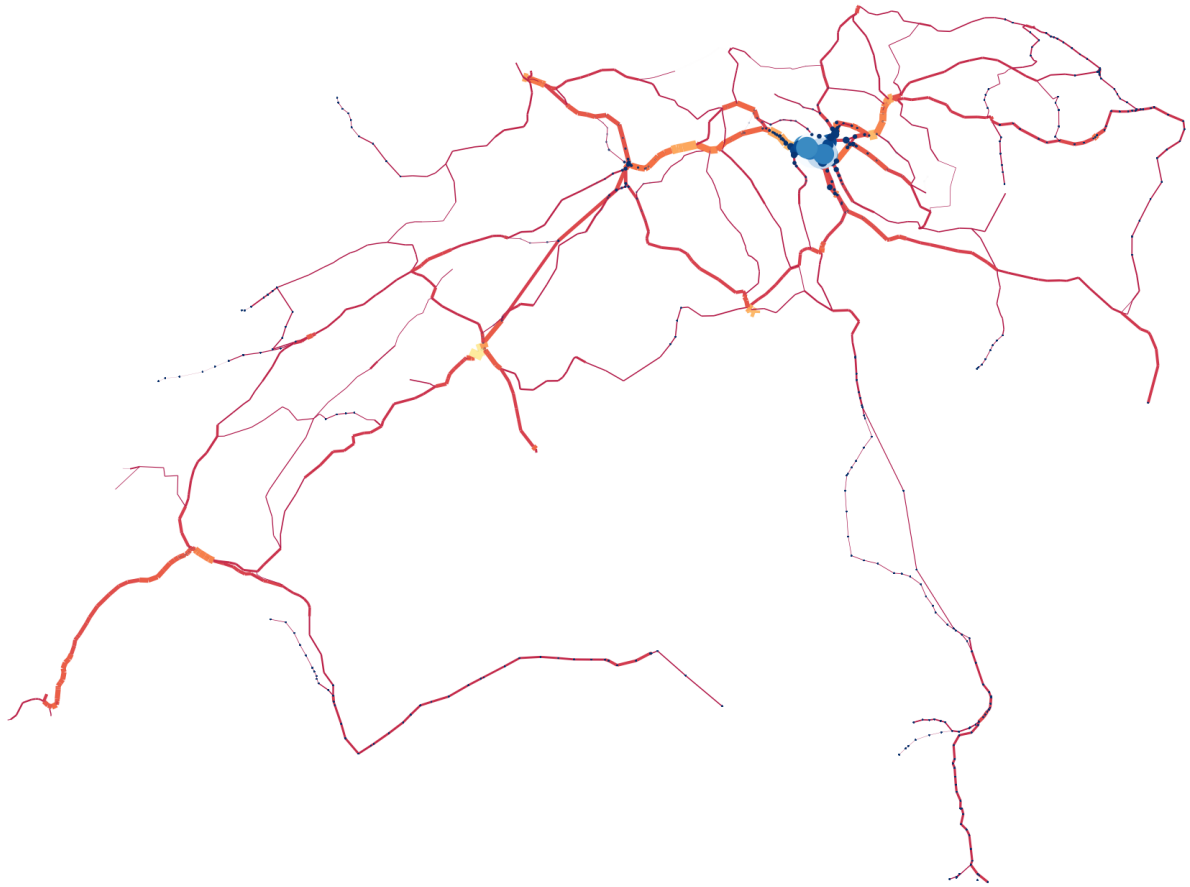
## Graph as a Map

Now the graph is fully operational and we can very easily do all kinds of fun calculations.

```
bond = np.array(list(nx.get_edge_attributes(G, 'weight').values()))
```

```
# Calculate degree centrality,  
eigenvector_centrality = nx.eigenvector_centrality(G)  
  
# Set degree centrality metrics on each node,  
nx.set_node_attributes(G, eigenvector_centrality, 'ec')  
  
# Use eigenvector centrality for visualization.  
ec = np.array(list(nx.get_node_attributes(G, 'ec').values()))
```

```
plt.figure(figsize = (12,9), dpi=150)  
nx.draw(G, pos=pos, edge_color=bond, node_color=ec, with_labels=False,  
        node_size=ec*1000, width=bond*20,  
        edge_cmap=plt.cm.Spectral, cmap=plt.cm.Blues_r)  
plt.savefig('SBB/map.jpg')
```

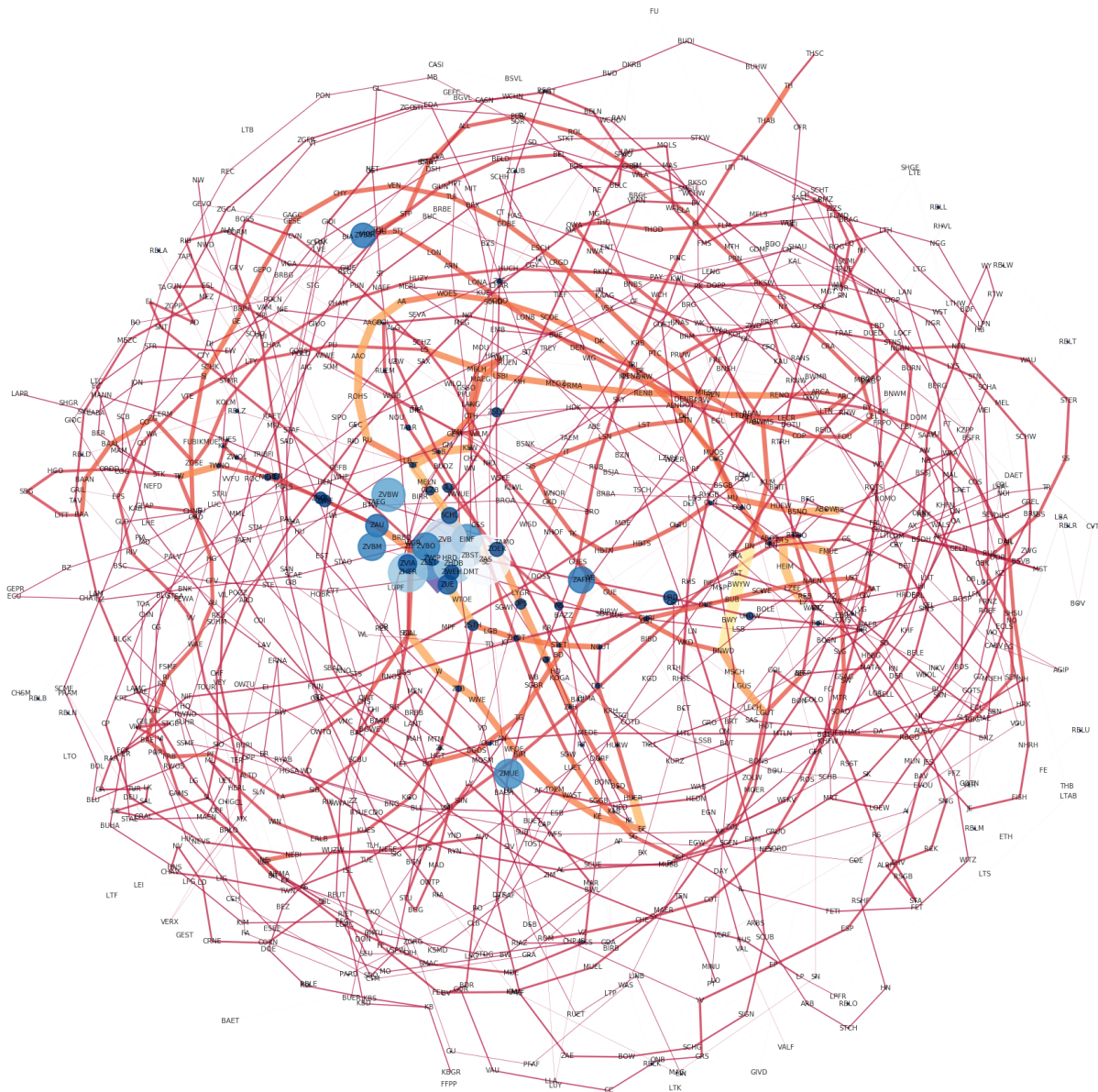


## Graph as a Network

New positions based solely on network dynamics abstracting away geography.

```
_pos = nx.spring_layout(G, seed=0)
```

```
plt.figure(figsize = (12,12), dpi=150)
nx.draw(G, pos=_pos, edge_color=bond, node_color=ec, with_labels=True,
        font_size=5, weight='weight', node_size=ec*2500, width=bond*20,
        edge_cmap=plt.cm.Spectral, cmap=plt.cm.Blues_r, alpha=0.8)
plt.savefig('SBB/atom.jpg')
```





Check out the [Jupyter notebook](#)!

P.S. The thick purple line in the core is `Langstrasse` with 336586 passenger trains in 2018.

EUCLIDorNOT

EUCLIDorNOT  
[mail@anagno.com](mailto:mail@anagno.com)

 [GAnagno](#)  
 [georg\\_anagno](#)

Images, numbers, sound and text are euclidean. Graphs, networks and manifolds are geometric.