The GitHub Network

Limingrui Wan, Danyu Zhang & Daniel Alonso

May 2nd, 2021

Description of the network

Our dataset corresponds to the GitHub.com network of developers. This data was collected from the public API in June 2019. Each node is a developer with at least 10 repositories starred and each edge is a mutual follower relationship between them. The vertex features are extracted based on the location, repositories starred, employer and e-mail address.

Source

The dataset was optained from the Stanford University SNAP website.. This dataset originally come from a paper published the 28th of september, 2019 called Multi-scale Attributed Node Embedding. The github repository for that project can be found here.

Characteristics of the network

Vertices/Nodes

Our network has the following amount of vertices/nodes:

```
#> [1] 37700
```

Degrees of vertices

Top vertex degrees

The vertices with the largest degrees (top 10) are:

```
#>      dalinhuang99             nfultz        addyosmani           Bunlong
#>              9458               7085               3324              2958
#> gabrielpconceicao
#>              2468
```

Who are they?

As these are individuals, we could peak inside GitHub.com and check the public profile of these individuals.

- Top user: **dalinhuang99**

The reason why this user might have so many followers could be the fact that he's followed a very large amount of users (160k as of May 4th 2021).

The user also seems to be a top 4% Stack overflow participant. However, the user has had no activity since July 17th 2018.

- Second top user: **nfultz**

This next user seems to have several useful tutorials/content in his github pages hosted static site. Also seems to have several active repositories where the user does some collaborative work.

There is no immediate apparent reason as to why the user has garnered such a large following.

- Third top user: **addyosmani**

This user has actually garnered a larger following than shown on the dataset. The user is an engineer at Google, working specifically on Google Chrome and the user has a significantly larger following on other social media (259.5k on twitter).

- Fourth top user: **Bunlong**

The user seems to be very active on github, committing code basically every day. The user also follows about 24.4k other users, and has created a couple projects that seem to be somewhat public as well.

- Fifth top user: **gabrielpconceicao**

The user also seems to be following a very large amount of users (31.8k following), not as active as the rest in the list, so the amount of followers could've come from following a large amount as well.

Bottom vertex degrees

The following amount of users have a degree of exactly 1 (one follower):

```
#> [1] 5045
```

Sum of vertex degrees

Calculated as follows:

$$\sum_{v=1}^{N} d_v = 2L$$

Corresponds to twice the size of the graph:

```
#> [1] 578006
```

Average degree

Calculated as follows:

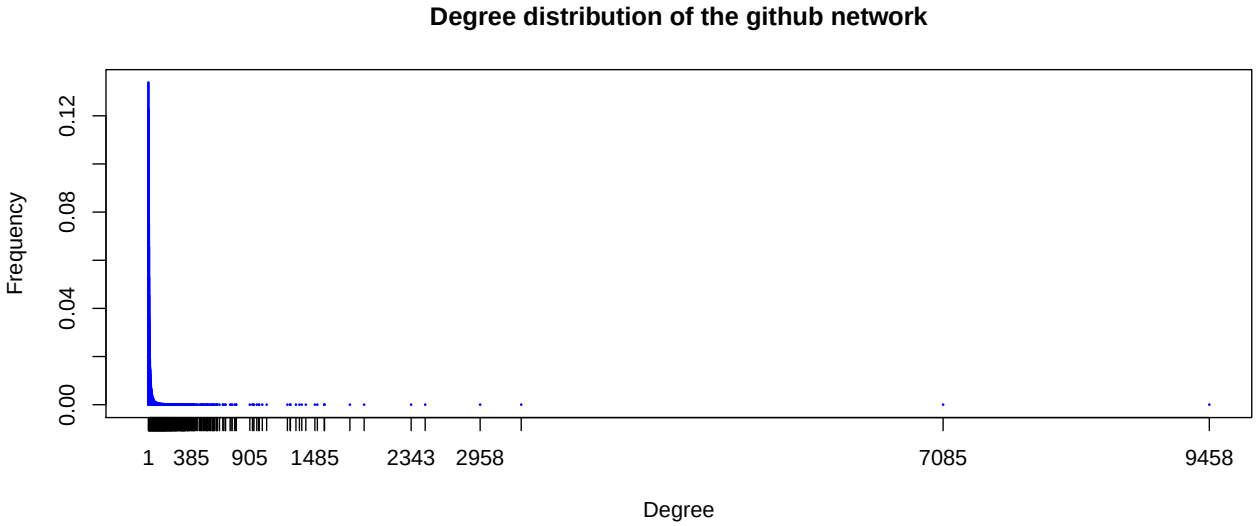
$$\frac{1}{N} * \sum_{v=1}^{N} d_v = 2\frac{L}{N}$$

Represents the average amount of mutual followers among all users in the network (nodes):

```
#> [1] 15.33172
```

Degree distribution

We can see the degree distribution of our graph as follows:



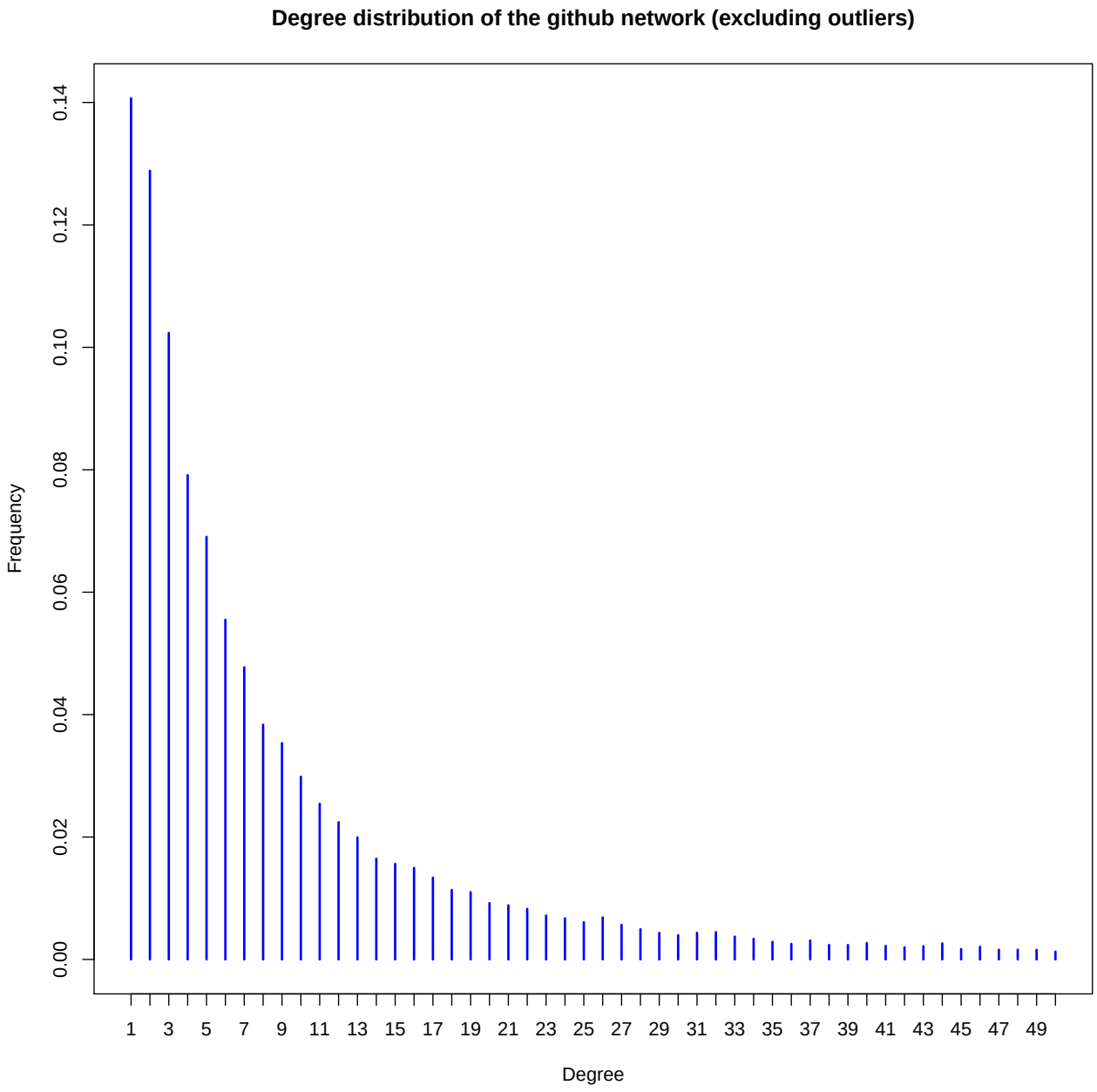
We can notice that our degree distribution is extremely right-skewed.

The reason for this we can clearly see by looking at a table of our degree frequencies (top 10):

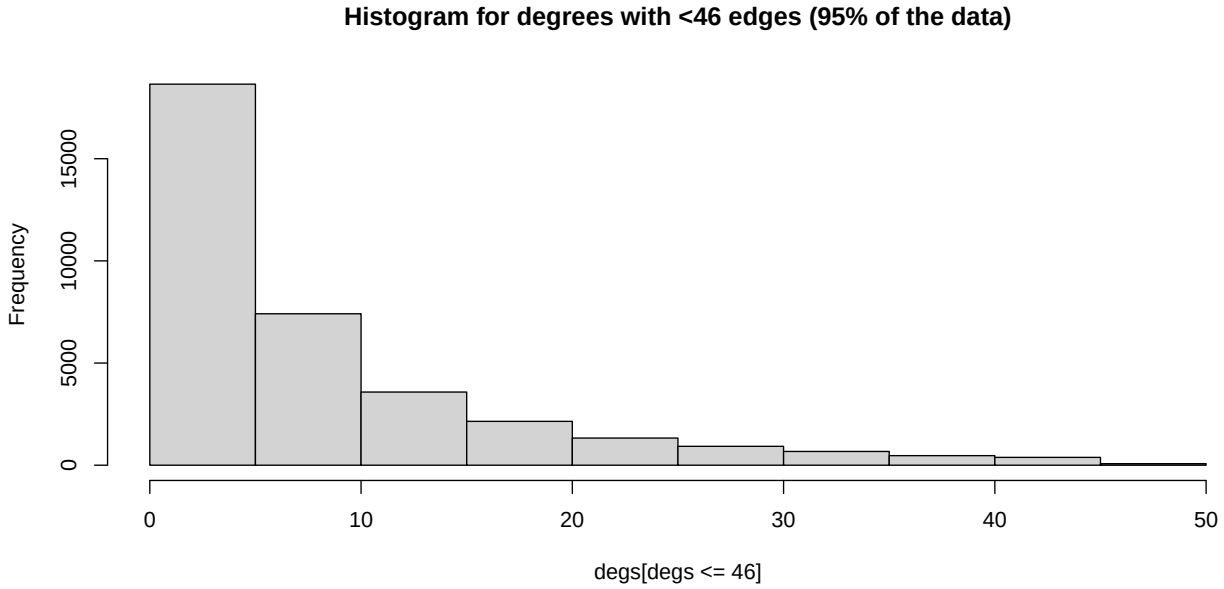
```
#>
#>    1    2    3    4    5    6    7    8    9   10
#> 5045 4620 3670 2837 2476 1990 1711 1375 1267 1070
```

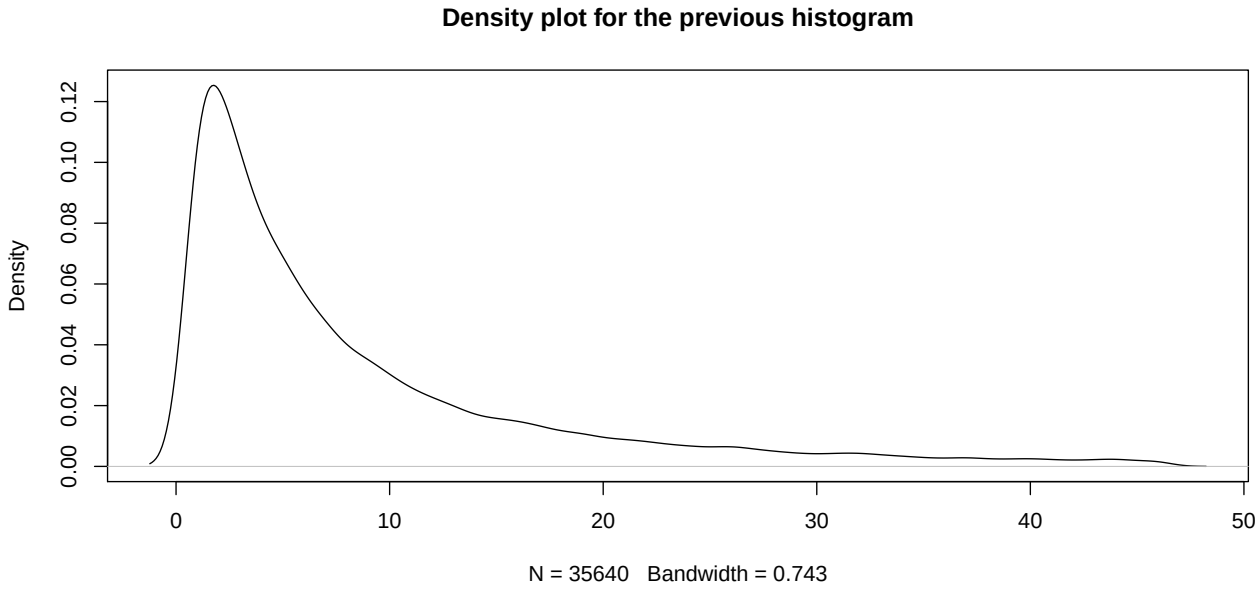
We can see the rightmost tail of our plot (top 50 frequencies), excluding the biggest outliers.

Here we can see a bit over 95% of the data points in our graph:



We can also create a histogram and density of our degree distributions, but for practicality and visual purposes, we avoid plotting any frequency above 46 edges.





Edges/Links

Our network has the following amount of edges (the size of the graph):

```
#> [1] 289003
```

The edges of the GitHub network are not weighted, because the edges represent a mutual follower relationship on GitHub, which would essentially always have a weight of 1.

Additionally, our network does not have any loop, therefore it's not a multigraph.

Connectedness

We can notice that our graph is connected, therefore every vertex is reachable from every other:

```
#> [1] TRUE
```

Diameter

Diameter is the shortest distance between the two most distant nodes in the network and with the *diameter* function we can obtain the value of the longest geodesic distance in the network.

This diameter corresponds to the following set of 12 nodes:

haochenli –> **ChiuMungZitAlexander** –> **IrvingZha0** –> **harryworld** –> **NigelEarle** –> **getify** –> **indrajithbandara** –> **kaizenagility** –> **artificialsoph** –> **kemacdonald** –> **jpriniski** –> **SOUMAJYOTI**

With a diameter of 11 (longest geodesic distance).

Farthest vertices

The farthest vertices corresponds to the previously mentioned first and last node, which are:

- Start: **haochenli**
- End: **SOUMAJYOTI**

Adjacency Matrix

For size purposes, it makes no sense to show the adjancency matrix of this network, as it would turn out to be a matrix of size 37,700 by 37,700.

Graphical representations