Tutorial: network analysis in Gephi

1. Introduction

Welcome to this workshop! In this hands-on part of the session, you'll be creating your own letter-sending network visualization in Gephi. We'll be working with a dataset containing 20,020 letters sent between the 16th and 18th century. The letters are all sent within a social network of scholars and scientists. From about 1500 to around 1800, the 'citizens' of this European knowledge-based civil society referred to their own community as the 'Republic of Letters'.

The people who constituted this 'Republic' have left behind hundreds of thousands of letters. Our dataset represents a slice of the exchange of knowledge that took place between them. By looking at these letters through the lens of network analysis, we can identify communities were involved in the exchange, who were key players, and how the different actors were positioned in relation to each other. The data was annotated in a crowd-sourced campaign that tasked volunteers with transcribing and annotating metadata about the letters, such as date of sending, sender, receiver, placenames, etc.

2. Getting to know the data

Let's start by getting to know the dataset. The data is spread across two different csv files:

- persons_CEN_1200to1800_cleanedSkillnet.csv

- letters_cen.csv

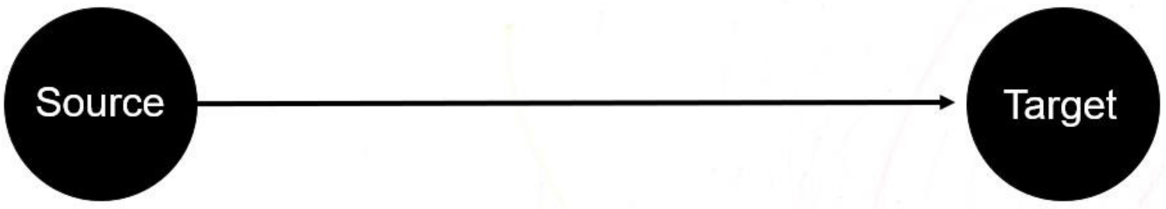
Open the files in a spreadsheet application of your choice (Excel, Numbers, Google Spreadsheets, etc.) and take a look around. What information can be found in this dataset? What types of insights do you think you can gain from doing a network analysis on this data?

3. Nodes and edges tables

Every network in Gephi consists of two data tables: a nodes table and an edges table.

The nodes table contains all information about our nodes. The nodes table **always** contains an **Id** column, which is used to identify the node. In this case, nodes represent a sender or receiver of one or more letters, and the Id column therefore shows their unique identifier. The full name of the person is under the column Label. Nodes tables can be enriched with extra information, too. We'll go into that later.

The edges table describes the connections between the nodes. Every edges tables contains a **Source** and a **Target** column. In a directed network, the Source is the node from which the connection originates, while the Target is the 'receiver' of the connection.

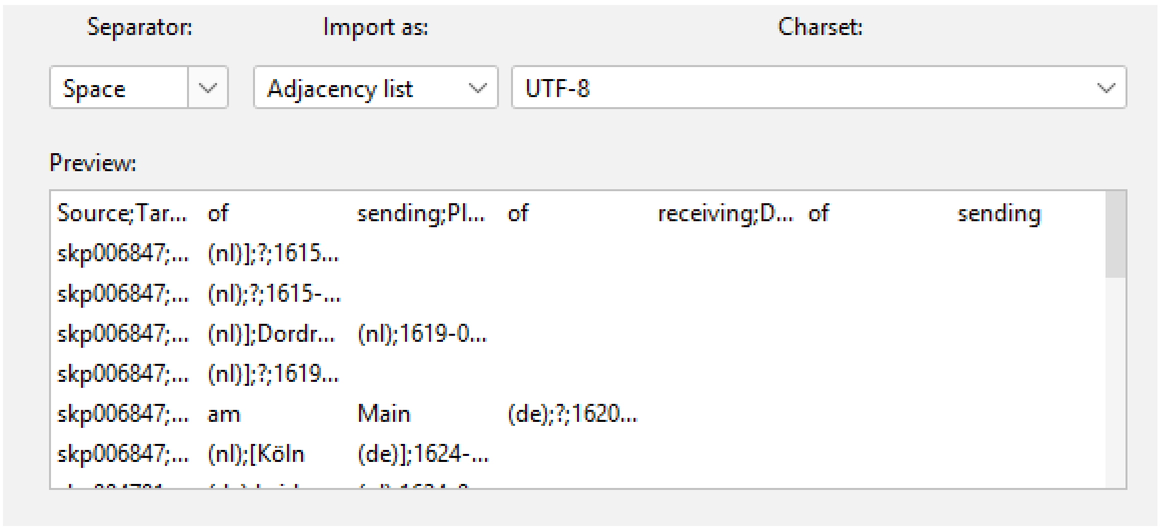


Open the nodes and edges tables in a spreadsheet application and compare the two. What do the connections in the edges table represent?

4. Importing the datasets

Time to make a network! Open Gephi and choose to make a New Project. We now need to import our edges and nodes tables. Go to File ➔ Import Spreadsheet and select edges.csv.

Gephi should automatically recognize that this is an edges tables. If not, it might show something like this:

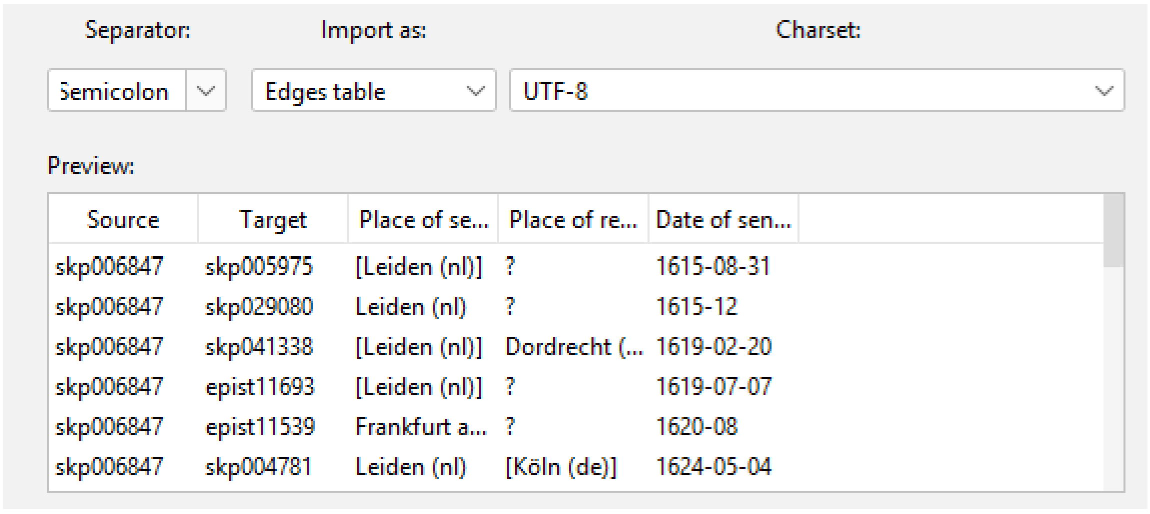


Separator: Space
Import as: Adjacency list
Charset: UTF-8

Preview:

```
Source;Tar...  of             sending;Pl...  of             receiving;D...  of     sending
skp006847;...  (nl)];?;1615...
skp006847;...  (nl);?;1615-...
skp006847;...  (nl)];Dordr...  (nl);1619-0...
skp006847;...  (nl)];?;1619...
skp006847;...  am             Main           (de);?;1620...
skp006847;...  (nl);[Köln     (de)];1624-...
```

Doesn't look right. We need to help Gephi a little by specifying the Separator (Semicolon) and the it should be imported as an Edges table:

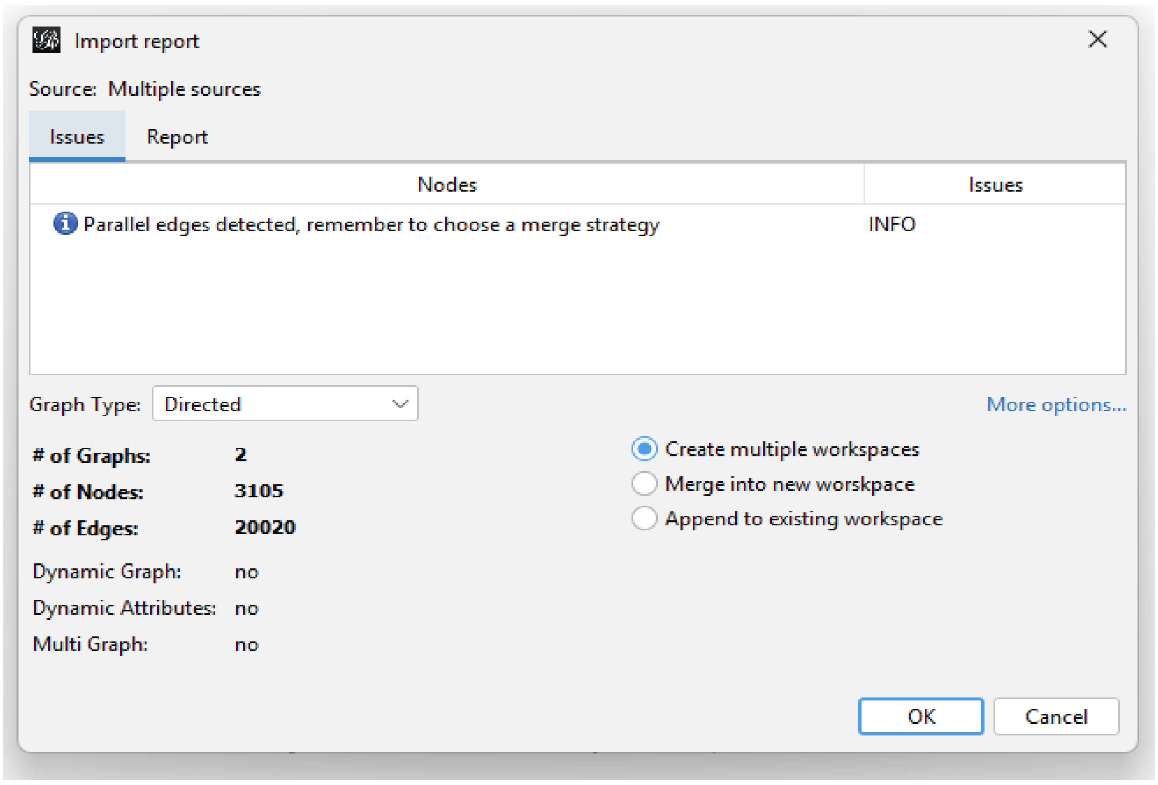


Separator: Semicolon
Import as: Edges table
Charset: UTF-8

Preview:

Source	Target	Place of se...	Place of re...	Date of sen...
skp006847	skp005975	[Leiden (nl)]	?	1615-08-31
skp006847	skp029080	Leiden (nl)	?	1615-12
skp006847	skp041338	[Leiden (nl)]	Dordrecht (...	1619-02-20
skp006847	epist11693	[Leiden (nl)]	?	1619-07-07
skp006847	epist11539	Frankfurt a...	?	1620-08
skp006847	skp004781	Leiden (nl)	[Köln (de)]	1624-05-04

Choose Next and wait for Gephi to show the import report.


Import report

Source: Multiple sources

Issues | Report

Nodes	Issues
Parallel edges detected, remember to choose a merge strategy	INFO

Graph Type: Directed

More options...

# of Graphs:	**2**
# of Nodes:	**3105**
# of Edges:	**20020**
Dynamic Graph:	no
Dynamic Attributes:	no
Multi Graph:	no

[✓ Create multiple workspaces] Merge into new worskpace Append to existing workspace

OK Cancel

Here we can decide a few key elements of our network, such as the type of network (directed, indirected or mixed), how parallel edges should be managed, and whether nodes should be able to connect to themselves (self-loops).

Our edges represent sent letters. Think about what this means for these parameters and set them accordingly. Not sure? Feel free to ask your instructor to help.

Next, let's do the same for the nodes table. Import the spreadsheet. If all went well, you shouldn't need to change any of the parameters. **Choose to Append to existing workspace, so that the nodes and edges table are both imported in the same workspace.** Not doing this will result in your nodes and edges tables being opened in two separate tabs.

Valuable tips and tricks (courtesy of Marjolein Krijgsman)

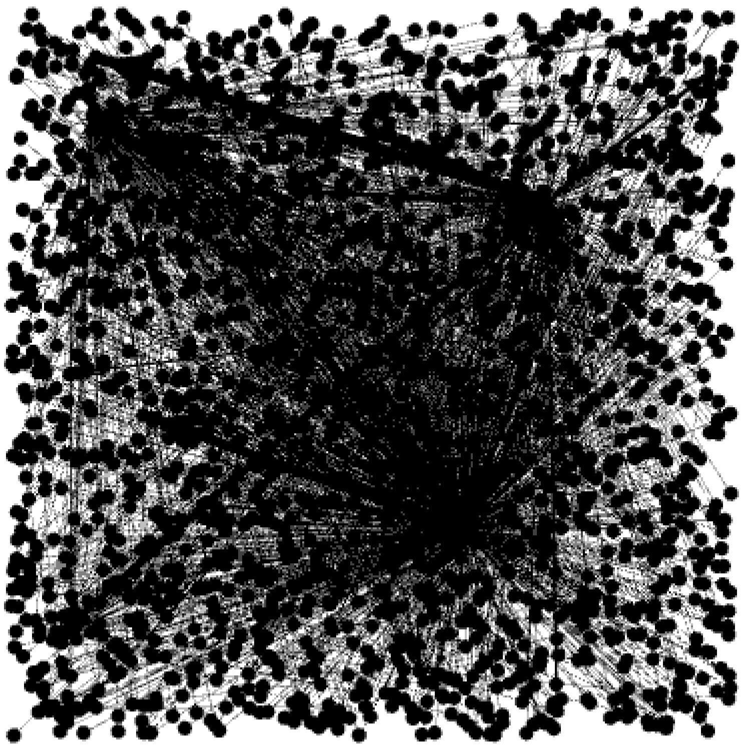
- Gephi has no 'undo' or 'redo' button. CTRL + Z does not work!
- **Save your project often (use the Save As option)!** Also save intermediate results, as there is no undo button. If you make big changes to your visualization that do not turn out well, you will still have your old save.
- In Gephi you can use various algorithms. ForceAtlas2 is the most used algorithm. If you use Gephi for academic research, you will need to understand how the algorithm works on at least a basic level. To do this, you can read up on the algorithms in the papers written by the developers.
- For others to understand a Gephi visualization, you will need

to provide additional contextual information.

- Hovering over features in Gephi will result in a small yellow box that explains shortly what the features can do.
- Don't be afraid to try out different options and see how your visualizations change accordingly. To master Gephi you will need to poke around the software to get to know the different functions.

5. The shape of a network to come

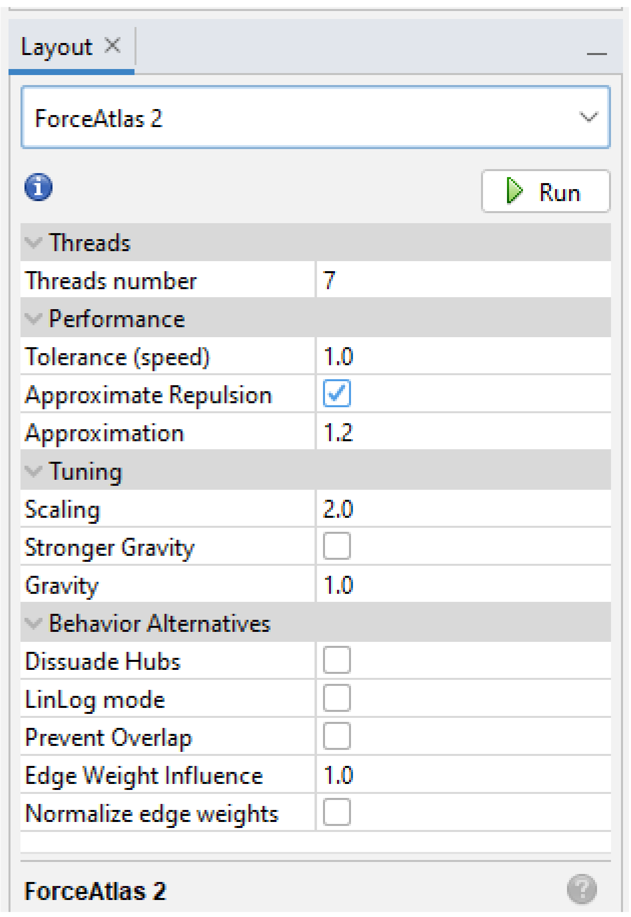
Most likely, this is will be staring at your from the screen right now:



No worries! This is the shape of an untouched network. Contained within this Pollock-like monstrosity are all your nodes and edges. In the upper-right corner, in the Context window, you can check exactly how many of them there are.

Let's start with a basic spatialization of the network. We'll be using the force-directed lay-out algorithm ForceAtlas2 for this. In essence, ForceAtlas2 treats nodes as if they were connected to each other by rubber bands. Stronger connections means that the nodes are attracted to one another, while loose (or weak) connections result in the nodes pushing each other away. This way tightly connected groups of nodes will cluster together, resulting in a network that can be read quite intuitively.

Go to the Lay-out window and choose ForceAtlas2 from the list of options.


Layout

ForceAtlas 2

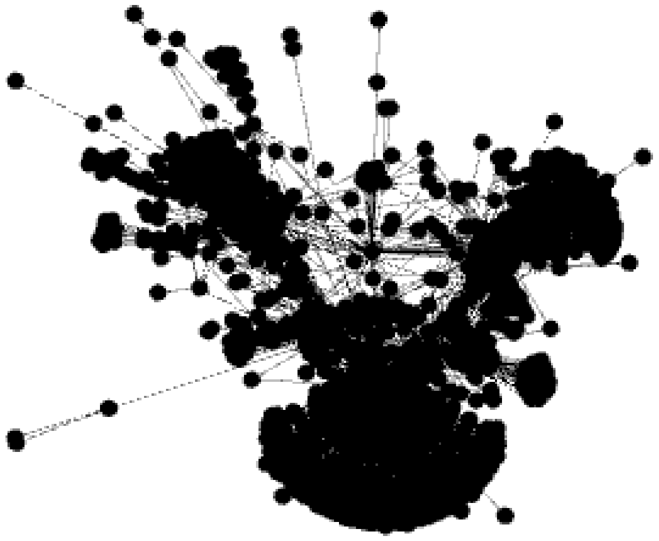
Run

Threads	
Threads number	7
Performance	
Tolerance (speed)	1.0
Approximate Repulsion	✓
Approximation	1.2
Tuning	
Scaling	2.0
Stronger Gravity	☐
Gravity	1.0
Behavior Alternatives	
Dissuade Hubs	☐
LinLog mode	☐
Prevent Overlap	☐
Edge Weight Influence	1.0
Normalize edge weights	☐

ForceAtlas 2

We want to start by roughly spatializing our network, and running the more precise algorithms later on. Tick the Approximate Repulsion and Dissuade Hubs boxes and click Run. It will look like the Big Bang will take place on your screen. Let this run for a little while, until the network barely moves anymore, then Stop the algorithm. **Please note that this algorithm never finishes by itself – it's always up to you to stop it running.**

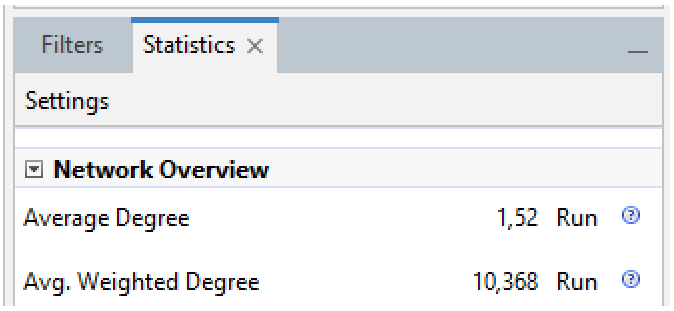
If you're feeling experimental, try ticking other options and see how the network changes.



Now that's more like it! Let's take a look around your network. You can zoom in and out by scrolling. Holding the right mouse button allows you to drag around the view.

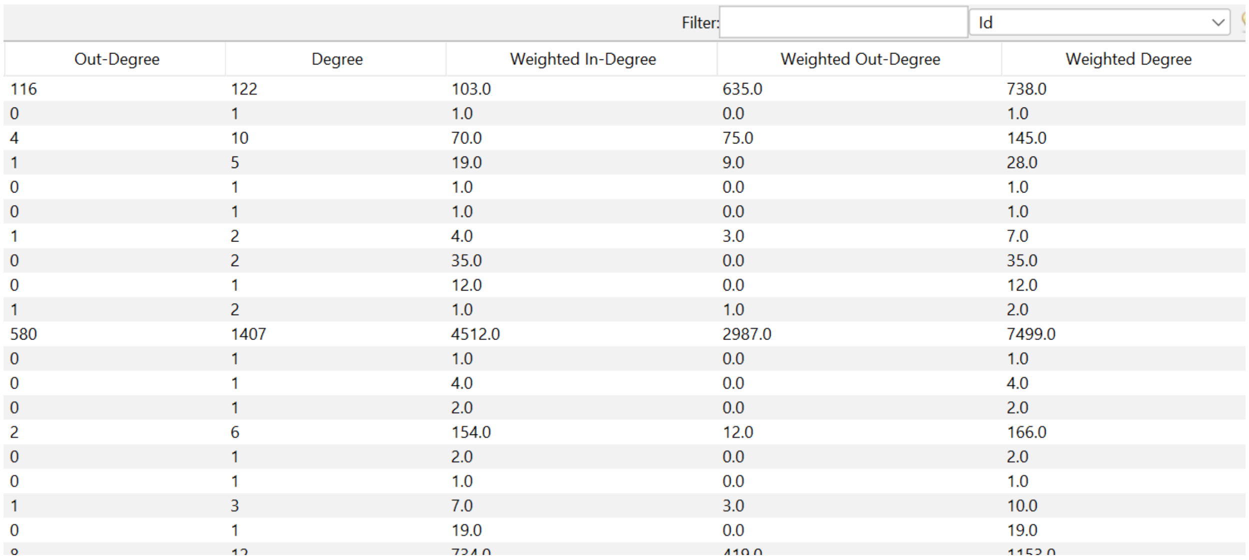
6. Running statistics

Now that our network looks like an actual network, we need to run some statistics to prepare for further analysis. Find the Statistics tab on the right-hand side and run the Average Degree and Average Weighted Degree calculations. Gephi will now calculate these metrics for each of the nodes.



Average Degree	1,52	Run	ⓘ
Avg. Weighted Degree	10,368	Run	ⓘ

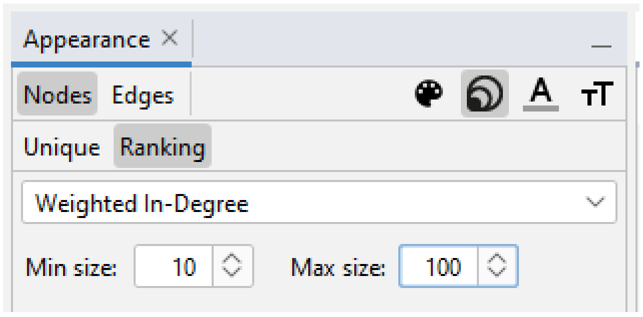
Go to the Data Laboratory tab on the top and navigate to your nodes table. Look at that: the results from our calculations have been added to the dataset!


Filter: | Id

Out-Degree	Degree	Weighted In-Degree	Weighted Out-Degree	Weighted Degree
116	122	103.0	635.0	738.0
0	1	1.0	0.0	1.0
4	10	70.0	75.0	145.0
1	5	19.0	9.0	28.0
0	1	1.0	0.0	1.0
0	1	1.0	0.0	1.0
1	2	4.0	3.0	7.0
0	2	35.0	0.0	35.0
0	1	12.0	0.0	12.0
1	2	1.0	1.0	2.0
580	1407	4512.0	2987.0	7499.0
0	1	1.0	0.0	1.0
0	1	4.0	0.0	4.0
0	1	2.0	0.0	2.0
2	6	154.0	12.0	166.0
0	1	2.0	0.0	2.0
0	1	1.0	0.0	1.0
1	3	7.0	3.0	10.0
0	1	19.0	0.0	19.0

Go back to the Overview. Let's use our calculations to transform the look of our network. We want to make nodes of users with many received letters bigger, and colorize the nodes based on how many letters they received. Go the Appearance window on the left. Here you can transform the visual appearance of the network based on your data values.

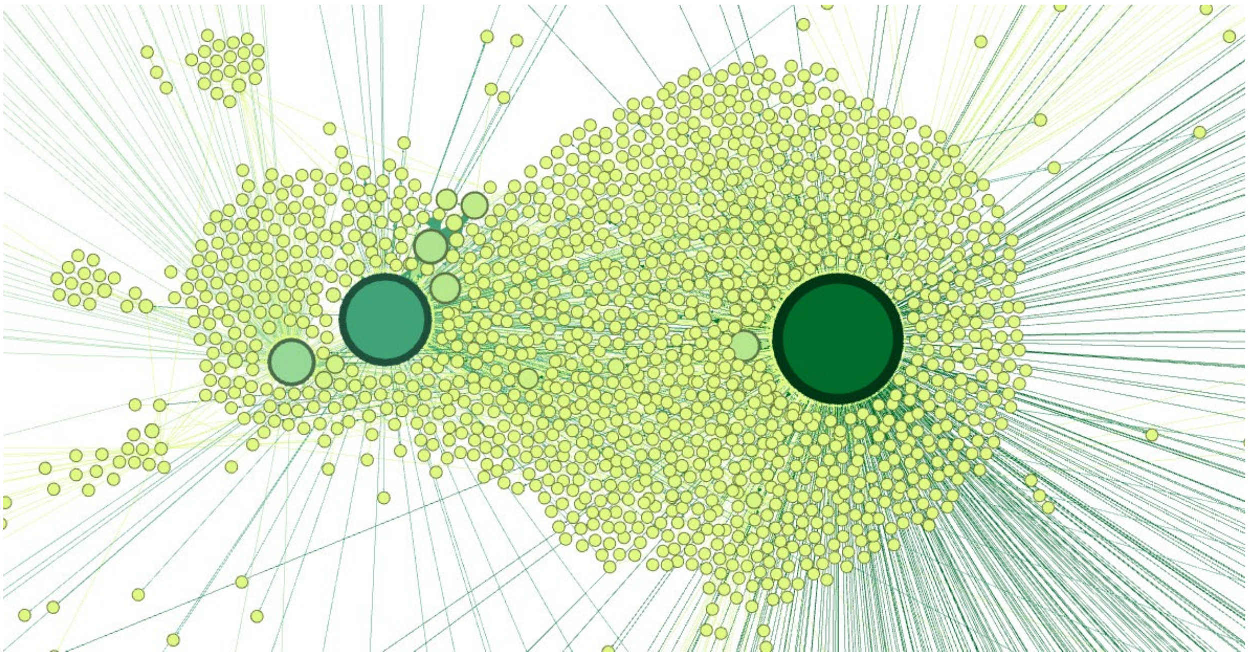
Let's start with the size of the nodes. Choose Nodes and the Size option (three circles). Go to Ranking, which gradually increases the size of a node based on a higher or lower ranking of a value. In this case, choose Weighted In-Degree (or: how many letters the node has received). The Min and Max size are up to you. For the size of this network, 10 and 100 will be appropriate. Now Apply this transformation.



Weighted In-Degree
Min size: 10
Max size: 100

Next, do the same for the color of the nodes by choosing the painter palette icon.

Because our nodes changed sizes, we need to give them some space. Go back to ForceAtlas2, turn off Approximate Repulsion (so that the algorithm will be more precise) and turn on Prevent Overlap (to make sure nodes don't cover each other) and Dissuade Hubs (so that tightly connected groups of nodes become more clearly visible). Run the algorithm for a while. Now the nodes will no longer cover each other.



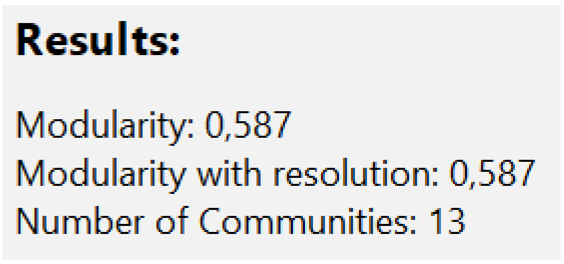
Our network is becoming more and more readable by the minute!

7. Modularity classes

One of the main reasons to do network analysis is to detect communities in a network of actors. Communities are groups of nodes that are intimately connected. It depends on the context what these groups represent. In our case, users that form a community very send letters between each other, which in most cases points to a high level of agreement. This can help us detect ideological or topical communities within a debate.

In Gephi, communities can be detected by the Modularity option under the Statistics tabs. Run Modularity. You will be asked to set a resolution. This will influence how big or small the resulting communities will be. The higher the number, the bigger the communities. Let’s start with the default resolution, 1. Run the algorithm.

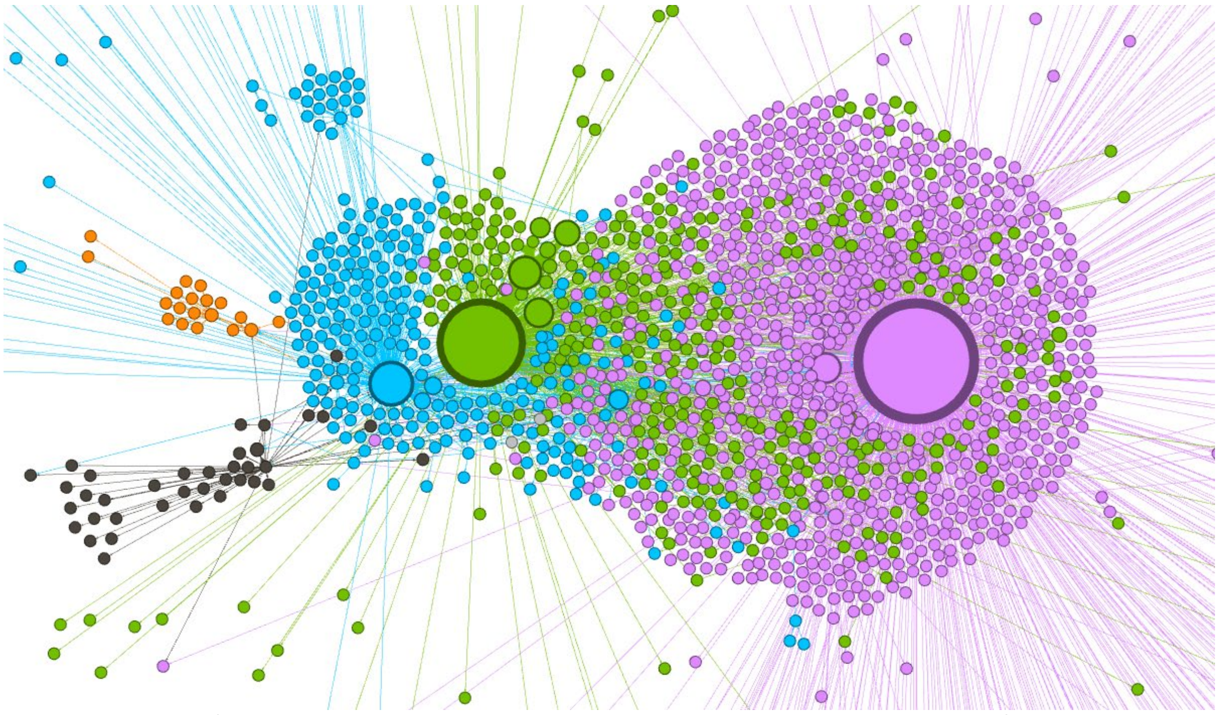
The results screen will show you the number of communities that Gephi detected.



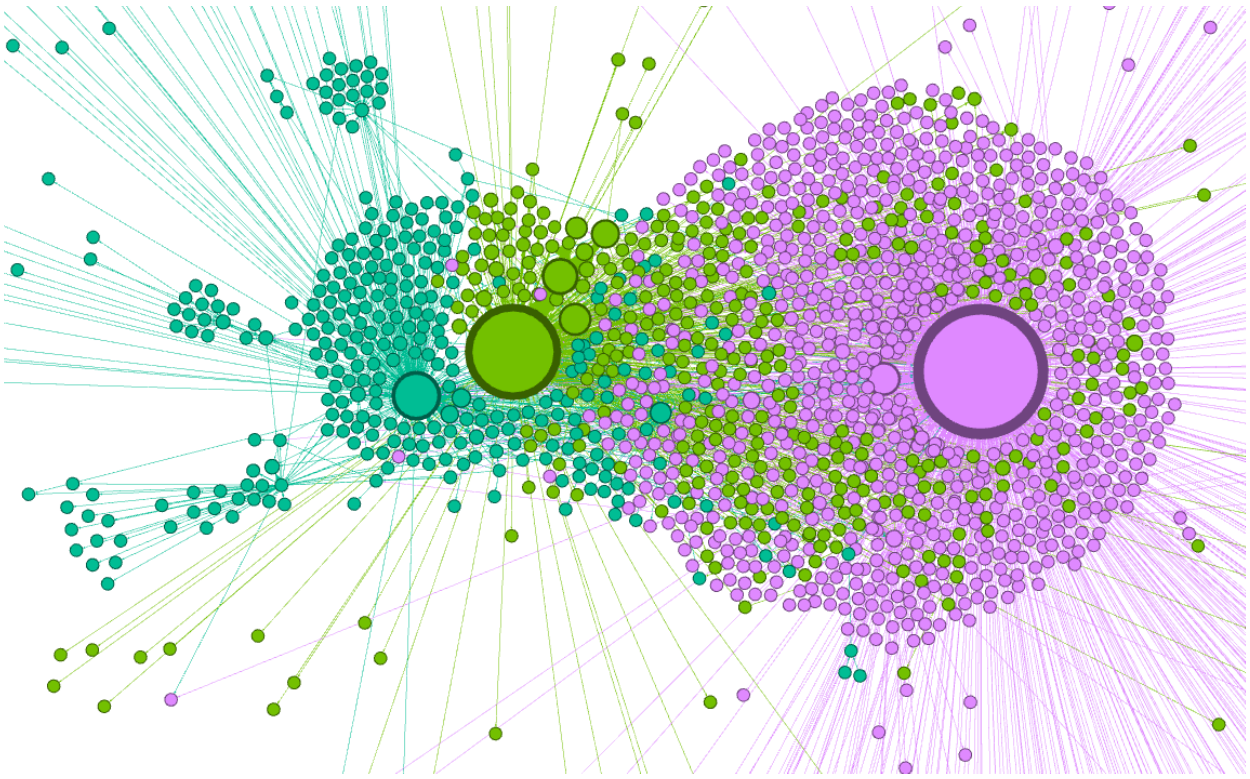
Results:

Modularity: 0,587
Modularity with resolution: 0,587
Number of Communities: 13

Colorizing the network based on the communities is one of the staples of network analysis. It helps us reveal the underlying structures of the network. Go to the Appearance tab ➔ Nodes ➔ Color ➔ Partition and select Modularity Class. Partition (as opposed to Unique or Ranking) allows us to assign a unique color to each different value for a metric. Go to Palette… and choose a nice set of colors for your communities. Now apply the colors. The result will look something like this:

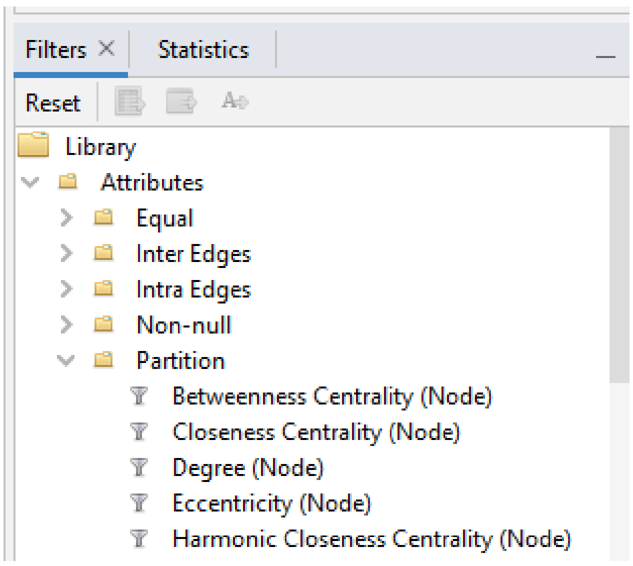

The network after running the Modularity analysis with a resolution of 1.

Sometimes, the number of communities makes it hard to parse the network. In that case, it might be useful to increase the resolution of the Modularity algorithm to find less, but bigger communities. Repeat the above steps, but with a larger resolution (for example, 1.5 or 1.8). See if the results improve. Please note that you need to regenerate the palette after each iteration of your Modularity calculation!

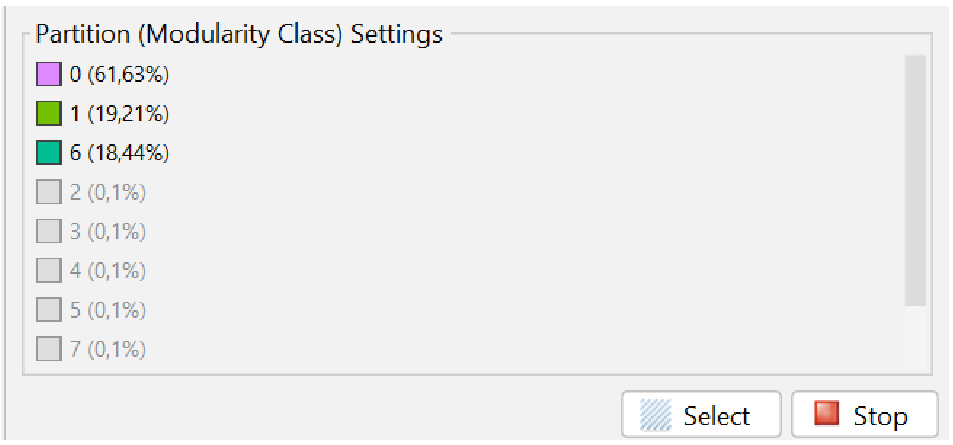

The network after running the Modularity analysis with a resolution of 1.9.

8. Using filters

Our network is still a bit messy. We can use filters to clean things up. First, let's start by only keeping the main clusters, eliminating the 'noise' on the borders of the network. Go to the Filters tab ➔ Attributes ➔ Partition.

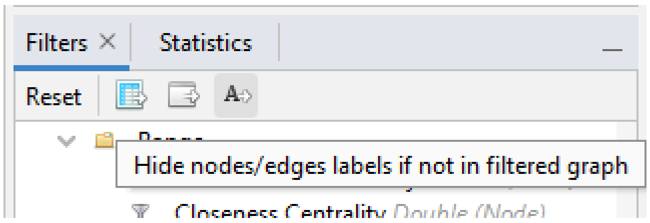


Find the Modularity Class (Node) filter and drag this to the Queries window underneath. By clicking the boxes next to the Modularity Class ID's, you can select which communities you want to keep. Start the Filter and select the biggest communities, starting from the top. The Context window on the top right shows you the percentage of nodes and edges that you have filtered out. In the case of this network, this mostly is useful for filtering out tiny, barely connected nodes on the periphery of the network.

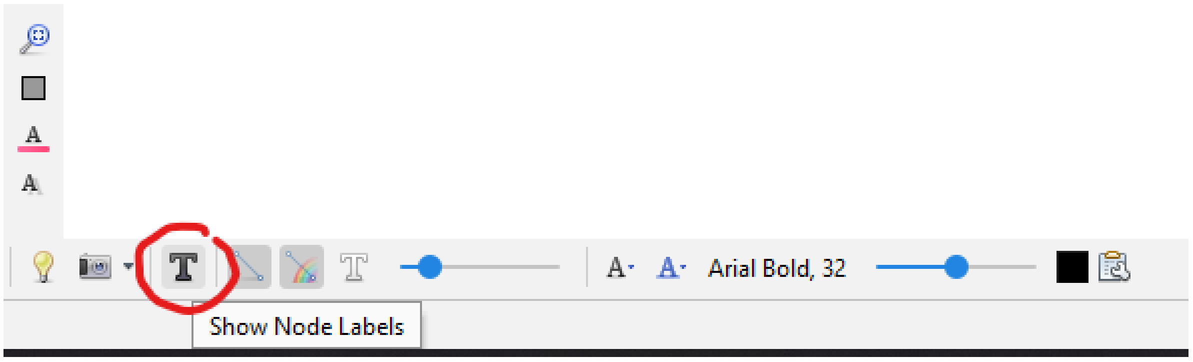


Always be aware that filtering is a very deliberate process. You inherently exclude data from your visualization once you start filtering. If you use filters in your analysis, be very clear in describing what you filtered and why.

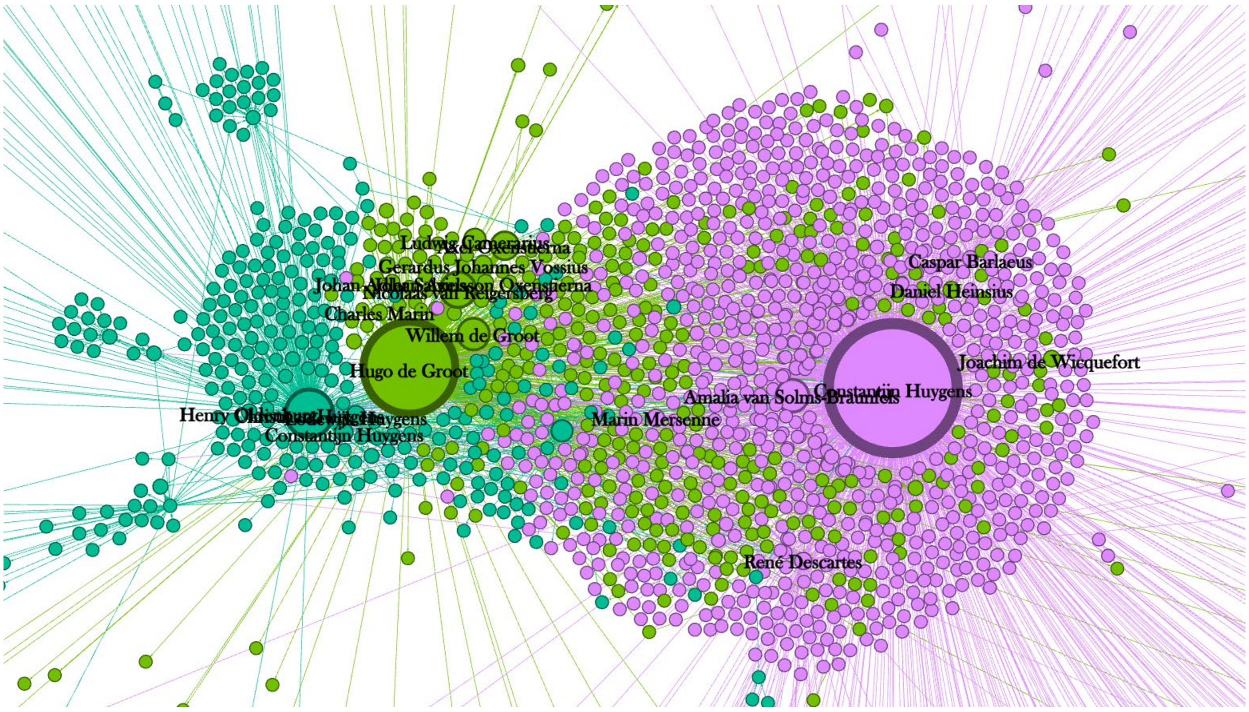
Filters can be used for different means as well. For example, we can use a filter to view the names of the biggest nodes in the network. Go to Filters ➔ Attributes ➔ Range and drag the Weighted In-Degree filter to the Queries window. This filter allows us to filter based on a range of Weighted In-Degree values, meaning the amount of letters a person received. Run the filter an start dragging the slider on the left. When you only see big nodes in your network, click the A with an arrow in the Filters window:


Filters
Statistics
Reset
Hide nodes/edges labels if not in filtered graph
Closeness Centrality Double (Node)

This option makes sure that only nodes that are now visible will show their labels (in this case, their full names, if known). Turn on the labels by clicking the black T on the bottom of the overview. You can change the size and appearance of the labels with the options on the right. Try playing around with these until you're happy with the result.



You can now turn on the Modularity Class filter again, which will return the network to the view that we made earlier.



9. Ego Network

As it stands, our visualization displays everything in our data. Sometimes you want to focus on specific nodes and their relationships, however. A very simple way of doing this is to hover your mouse over a node. It will now highlight all its direct connections.

A more advanced method is the Ego Network. The ego network is a filter in Gephi.

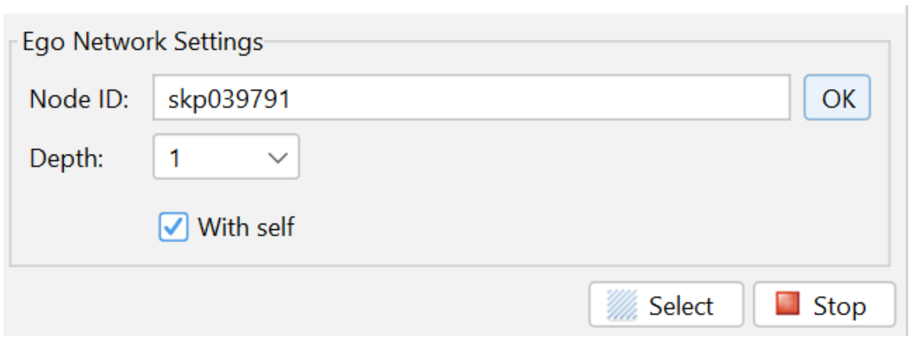
Go to the Filter window on the right-hand side, open the Topology folder and find Ego Network. Drag it to the Queries window below. This is now one of our filters.

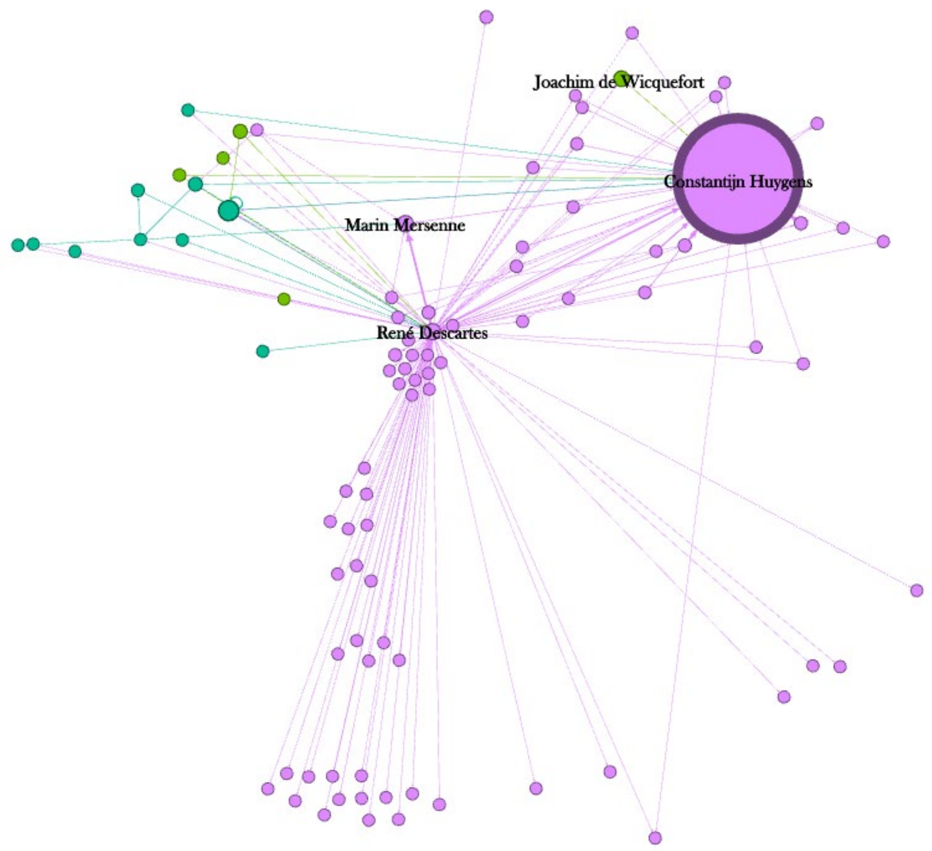
To use this filter, we need to fill in the ID of the entity we're interested in. We can find these in the Data Laboratory, but sometimes it's more handy to keep the dataset open in Excel. You can then copy the ID of the nodes you're interested in into Gephi.

Next, click OK and then Filter.

Our node and all its direct connections will now be highlighted. We can increase the depth of the ego network using the Depth dropdown menu in the filter. Setting it to 2, for example, will also display all nodes connected to the nodes shown above.

Ego networks are a very helpful way to identify patterns in your graph. It allows you to focus on specific entities and its connections, and will often reveal connections that you might not have noticed within the full graph.

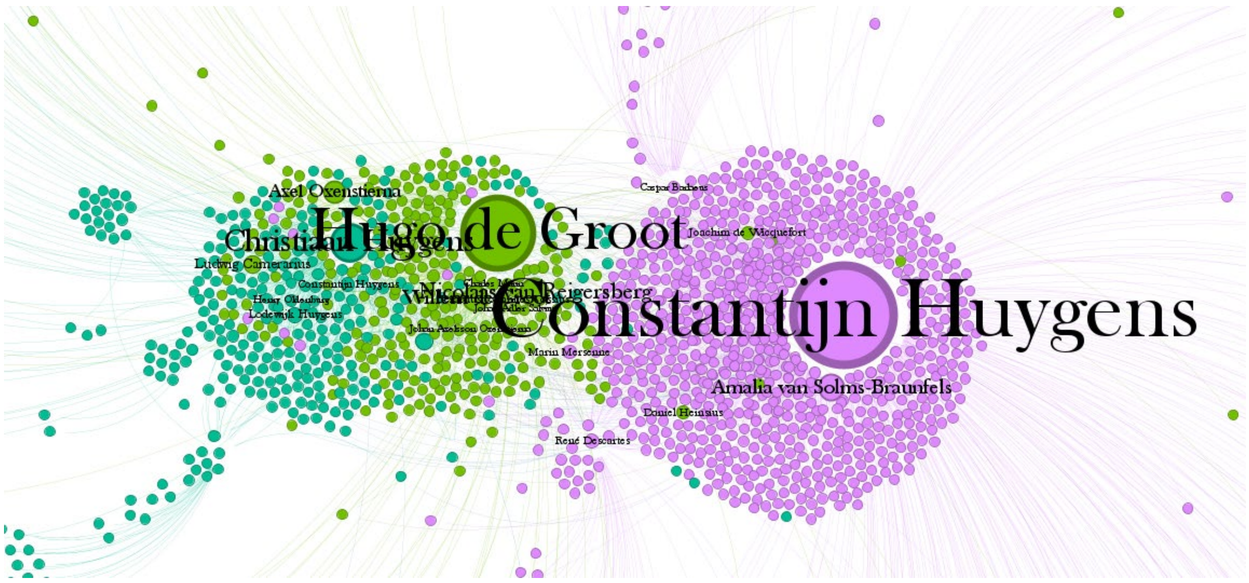

Ego Network Settings
Node ID: skp039791
Depth: 1
With self: ✓



10. Exporting the visualization

Now that we have a readable network, we can export our visualization as an image file so that we can share it with others. Go to the Preview tab at the top of the window. Here we can change how our network appears in the image. The structure of the network won't change here, only how it shows up.

Play around with the options here and see what they do. Please note that you need to click Refresh for the changes to show up. Find a combination of settings that result in a network that you think is nice and clear.



Once you're happy with how your visualization looks, we can export the image. Go to Export on the bottom left, and choose PNG. Click options and increase the resolution to be very high (for example, 16000 x 16000 pixels). Because networks can often contain tiny details and letters, you want your image file to be of high quality so that you can zoom in without loss of detail. Export the image.

11. Exporting your data

Finally, let's get the results of our network analyses out of Gephi. Go to the Data Laboratory, select the nodes table and click Export Table. The data that was visible in your nodes table will now be exported as a CSV file, which can be used for further analysis in different environments (e.g. Excel, Tableau, Python, R).

12. What's next?

This tutorial taught you the basics of Gephi by guiding you through the creation of a network visualization. You know how network data is structured, how to import that data, spatialize the network, run statistics, change the appearance of nodes based on the results of these metrics, use filters to clean up the network and export the result as an image. These steps are all you need to tackle all kinds of different networks in Gephi.

However, we have barely scraped the surface of what Gephi can do. There are many more different metrics, visualization options, and lay-out algorithms waiting for you. Gephi is very well documented, and hovering over options will give you a clear description of what each option does. It is therefore very easy to start experimenting with the possibilities of this program. Want to colorize nodes by country of origin? Make a dynamic timeline? It's all possible.