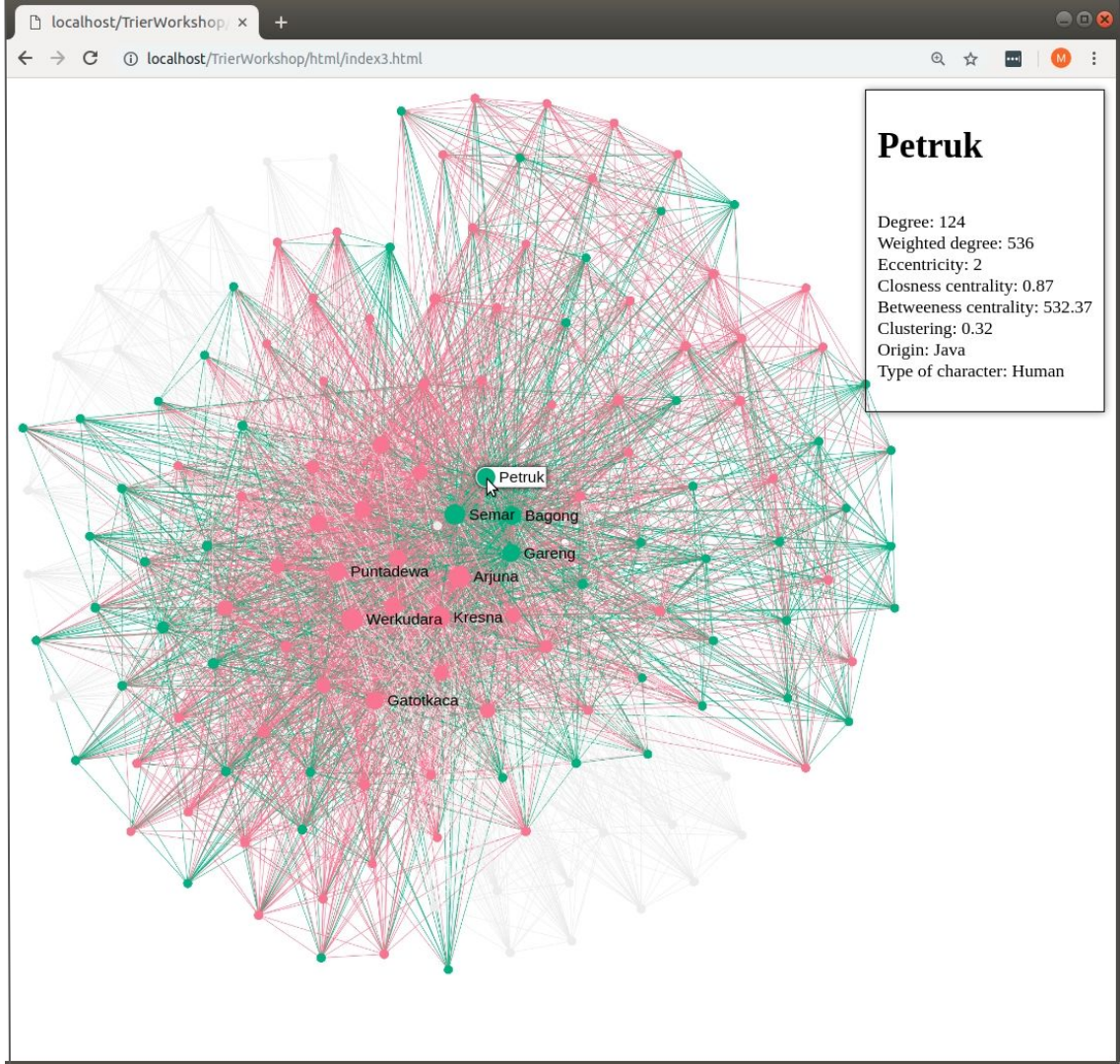
Interactive network visualizations

With Gephi and JavaScript

Miguel Escobar Varela, PhD
Assistant Professor, National University of Singapore
miguelescobar.com
@miguelJogja



Workhop materials

https://github.com/villaorlado/TrierWorkshop

Wayang network visualization

https://github.com/villaorlado/wayangnetworks

Motivation

Network analysis is becoming increasingly common in the Digital Humanities. But the most common ways of communicating results is including unreadable screenshots of hairballs, or reporting quantitative data. Sometimes the most interesting features of the network only emerge when one is able to navigate through an interactive network and to hyperlink network nodes to additional data. To achieve this kind of interactivity we will embed interactive networks in web projects with JavaScript.

Objectives

Make network diagrams in Gephi

Export those networks as GEXF files

Embed this files in a web project with sigma.js

Tweak sigma.js for additional functionality

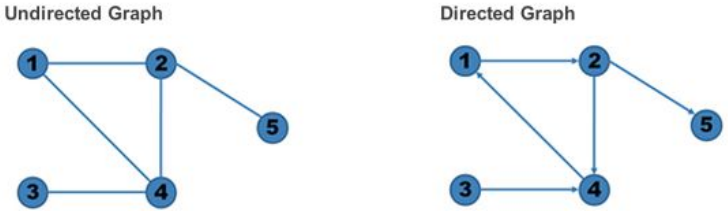
Part 1 Short overview of network analysis

Basic Network Concepts

A network consists of:

Nodes (things that are connected).

Edges (connections between those things). Specific, explicit connections between things. They can be directed or undirected.



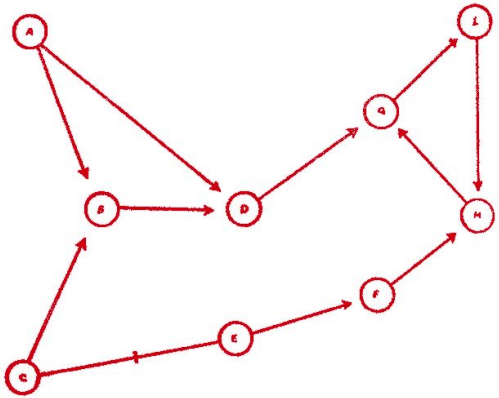
Examples: social networks, traffic networks, communication networks, citation networks.

Examples

- Citations
- Collaborations
- Friendships
- Communication networks (places linked together by roads)

Sociograms

- Jacob Levy Moreno



Moreno's Image of Who Recognized Whom Among a Collection of Babies (Moreno, 1934, p. 32).

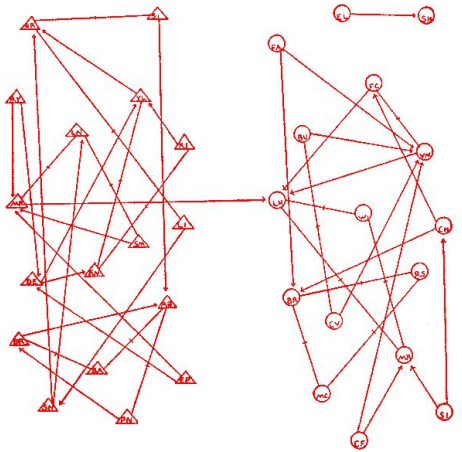
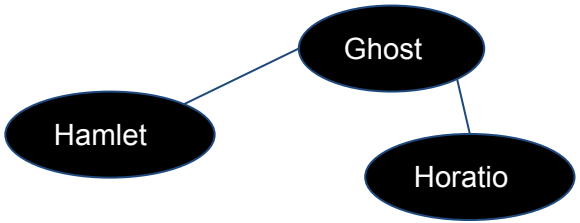


Figure 3. Friendship Choices Among Fourth Graders (from Moreno, 1934, p. 38).

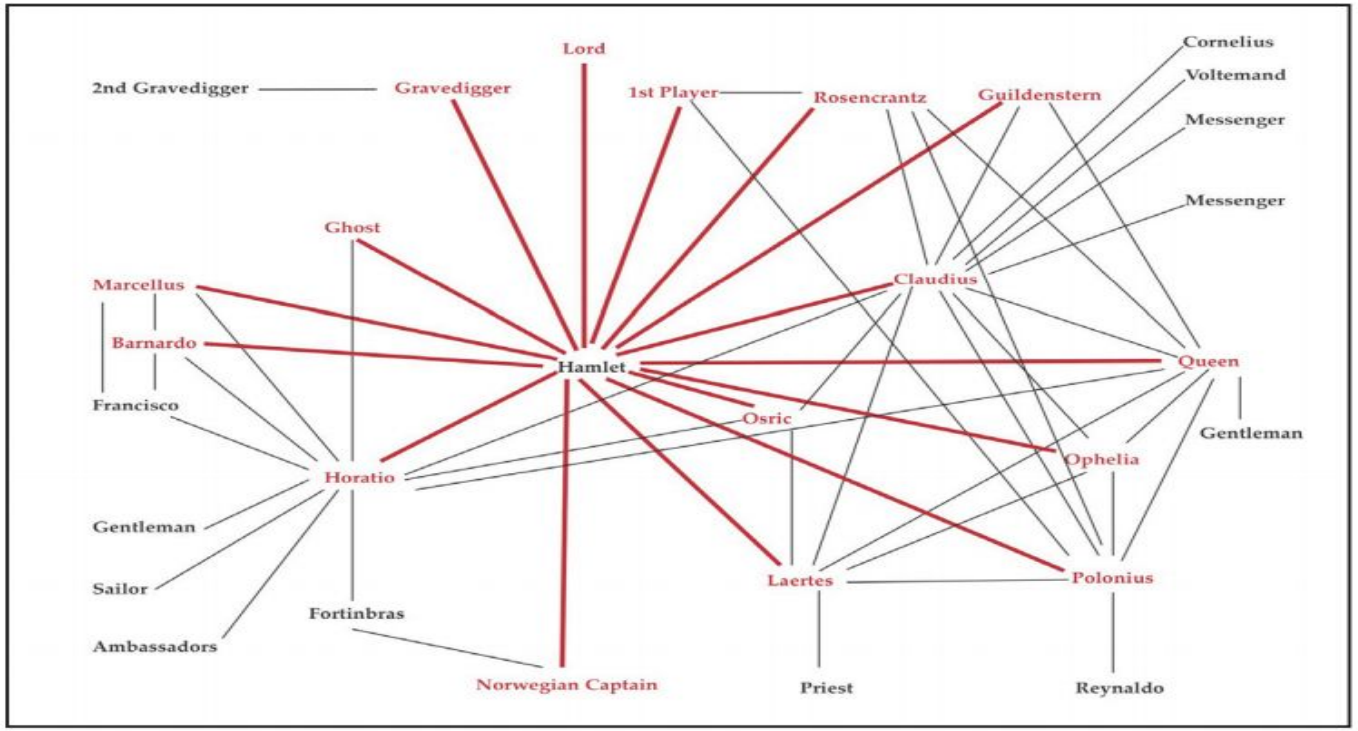
Hamlet as a network

Characters as nodes, interactions as edges.



[This example shows only the edges for the Ghost, who only interacts with two other characters]

Moretti on Hamlet

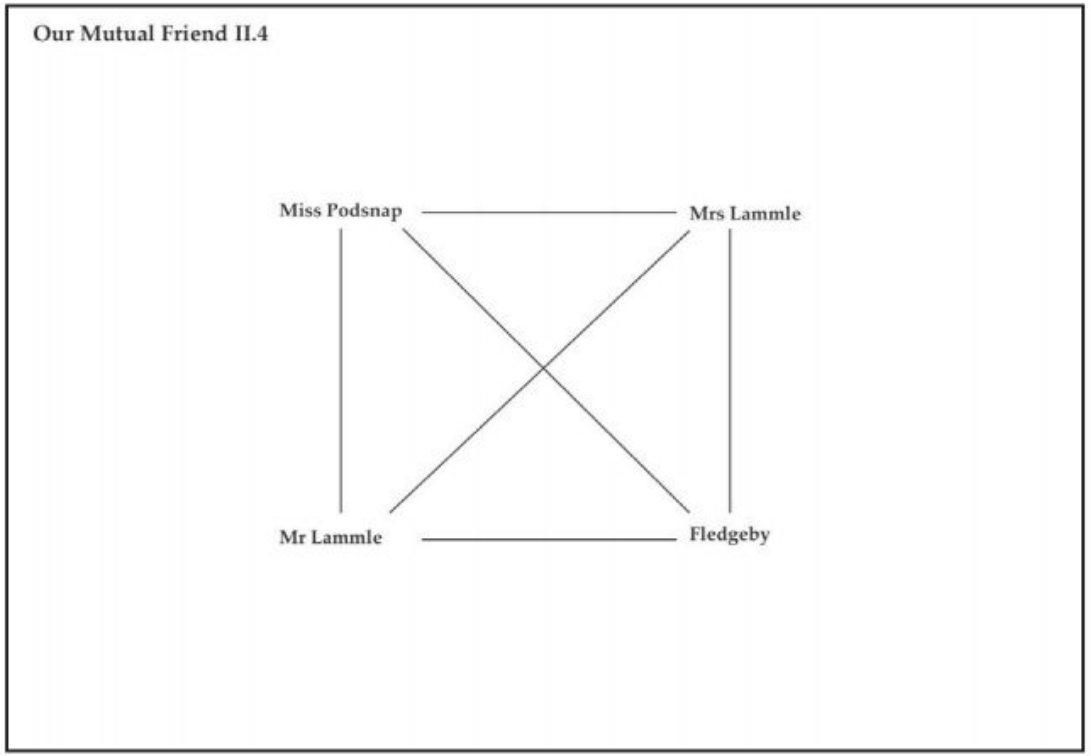


Moretti on Hamlet

Macbeth shows a ‘region of death’ in the network. Except for Osric and Horatio, all characters linked to both Claudius and Hamlet are killed: “outside that region, no one dies in Hamlet […] the tragedy is all there.” Moretti also identifies two separate components, sub-regions where each nodes is connected to every other node, which he interprets as corresponding the changing political landscape and the fight between the court and the state.

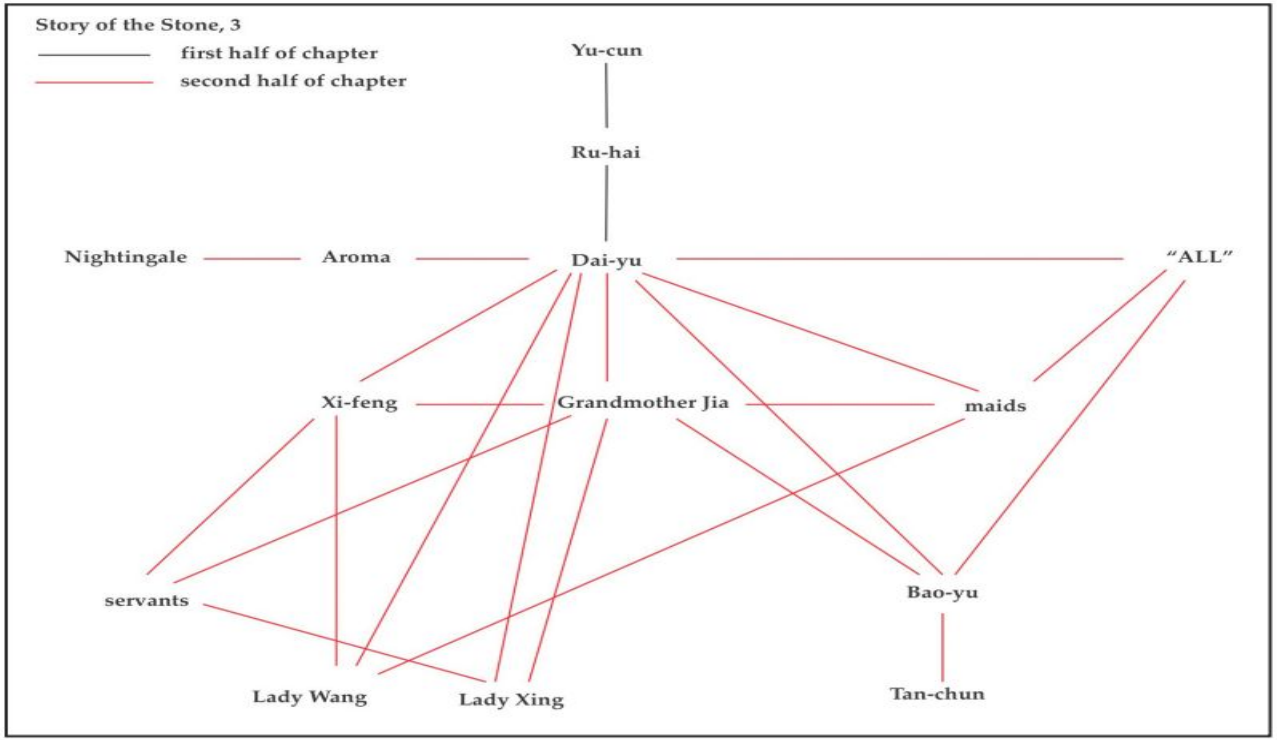
Moretti, *Network theory, plot analysis,* p272.

Moretti on Comparisons



Dickens' *Our Mutual Friend*

Moretti on Comparisons



Moretti on Comparisons

Dickens' *Our Mutual Friend* to the classical Chinese Cao Xueqin's *The Story of the Stone* (石头记), he suggest that the shape in the network of the Chinese text is determined by the concept of *guanxi* (关系), which can be understood as "connections" or a chain of "transactions that generate indebtedness." Ibid, 237.

Network measurements

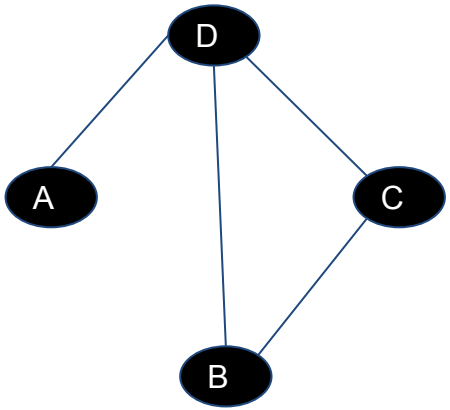
Degree. Total number of edges it has to other nodes.

Density. The portion of the potential connections in a network that are actual connections.

Average path length.The average steps (edges) to go from one node to another node.

Network measurements

Degree. Total number of edges it has to other nodes.



Degree list:

A 1

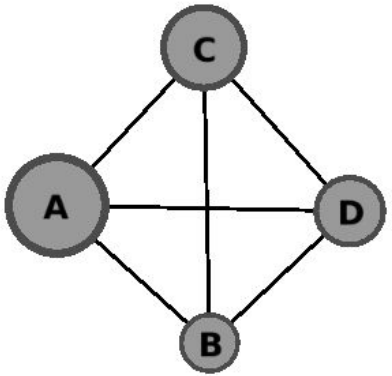
B 2

D 3

C ?

Network measurements

Density. The portion of the potential connections in a network that are actual connections.



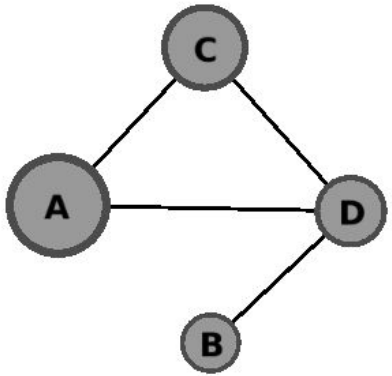
$$\text{Potential connections} = \frac{n * (n-1)}{2}$$

$$\text{Density} = \frac{\text{Actual connections}}{\text{Potential connections}}$$

Potenail connections = 4 * 3 / 2 = 6
Actual connections = 6
Density = 6/6 = 1

Network measurements

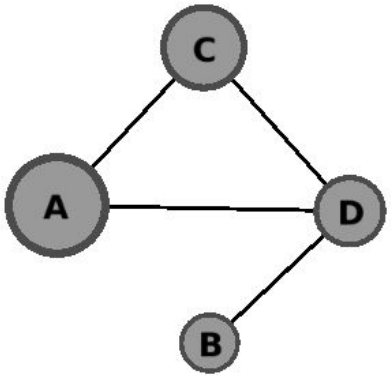
Average Path Length. Average number of steps along the shortest paths for all possible pairs of network nodes.



Node pairs	Path length
A, B	2
A, C	1
A, D	1
B, C	2
B, D	1
C, D	1
Average	**1.33**

Network measurements

Density. The portion of the potential connections in a network that are actual connections.



Potential connections = $\frac{n * (n-1)}{2}$

n is the number of nodes

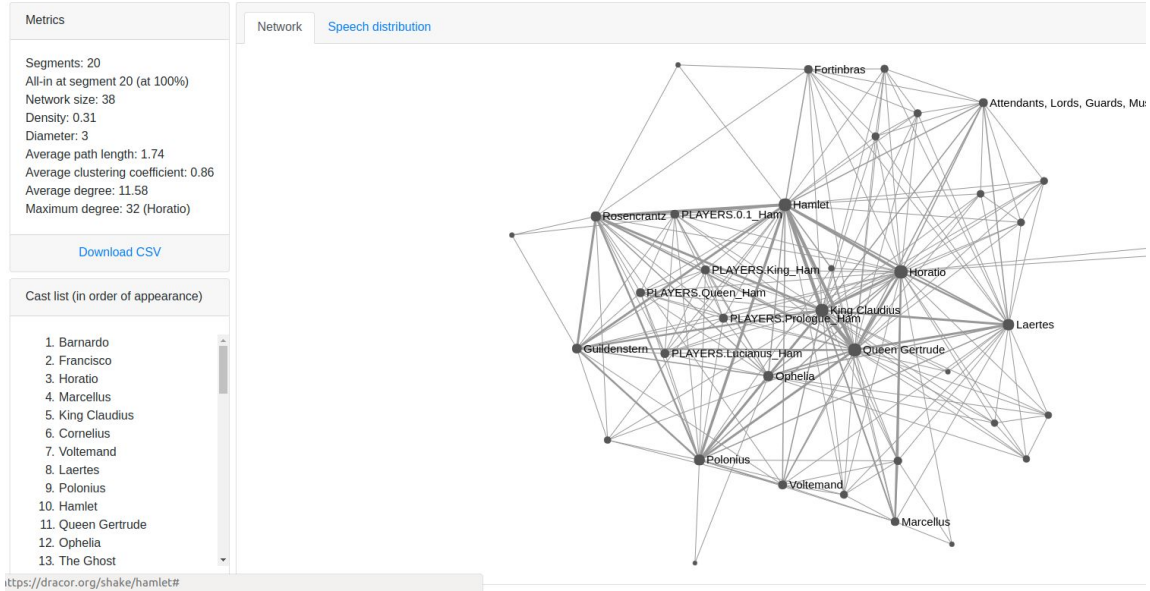
Density = $\frac{\text{Actual connections}}{\text{Potential connections}}$

Potential connections = 4 * 3 / 2 = 6
Actual connections = 4
Density = 4/6 = 0.667

Analysis of Shakespeare networks

William Shakespeare

Hamlet



https://dracor.org/shake

Network analysis of German drama

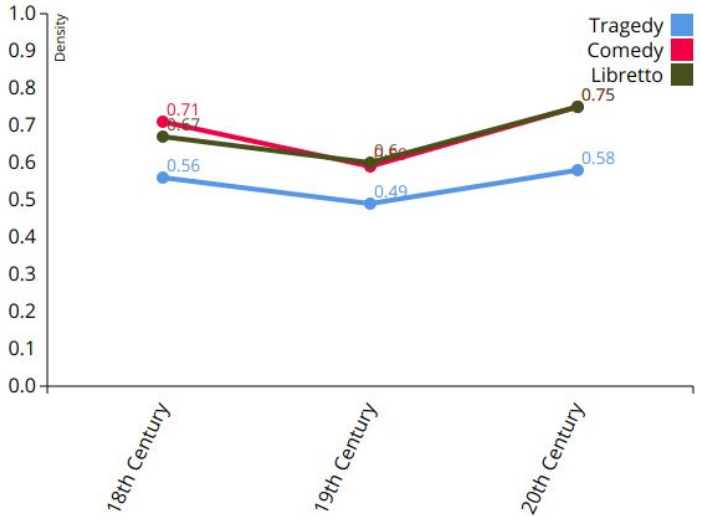
Table 1: Network Measures, by Genre

	N=	Number of Characters (Median)	Max Degree (Median)	Average Degree (Average)	Density (Average)	Average Path Length (Average)
Corpus	465	16	13	9,01	0,59	1,46
Tragedy	101	19	16	9,57	0,52	1,56
Comedy	92	14	11	8,61	0,67	1,36
Libretto	56	16	13,5	9,09	0,64	1,39
Other	216	17	14	8,88	0,59	1,48

https://dlina.github.io/Network-Values-by-Genre/

Network analysis of German drama

Fig. 5: Density (Mean), by Genre and Century



https://dlina.github.io/Network-Values-by-Genre/

Using Gephi

Which data to use?

Loading data [my wayang data]. Loading edgelists and attributes.

Visualizations. Calculate basic features.

Arrange the project visually.

Export as GEXF

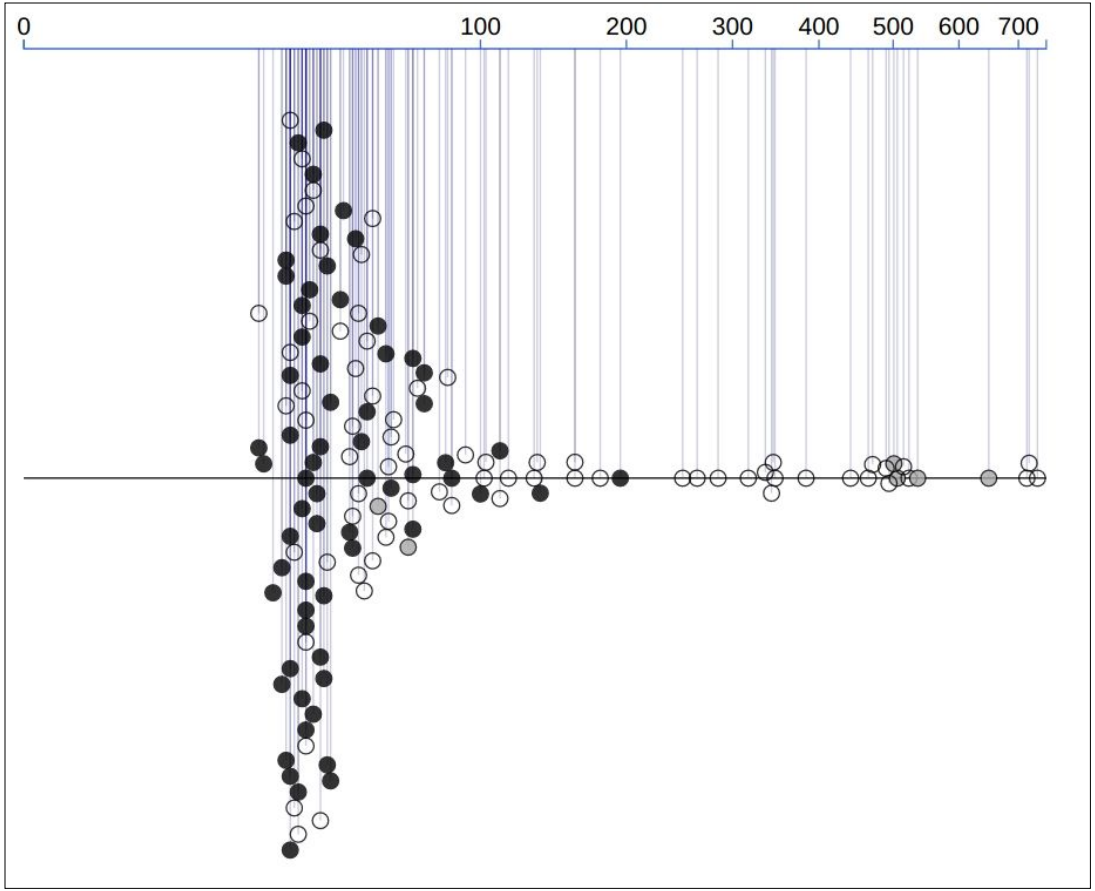
Three types of characters

Bima (Indian in origin)
Indian characters 53%
Avg. Wtd. Degree: 153.47

Dewa Ruci (Javanese in origin)
Javanese Characters 47%
Avg Wtd Degree 73.36

Gareng (Javanese clown-servant)

Although the percentage of Javanese and Indian characters is comparable, Javanese characters have disproportionately lower degrees.



The different characters according to their weighted degree. The x axis is in an exponential scale (factor = 0.3). Black circles are Javanese, white are Indian and gray are Javanese Punokawan.

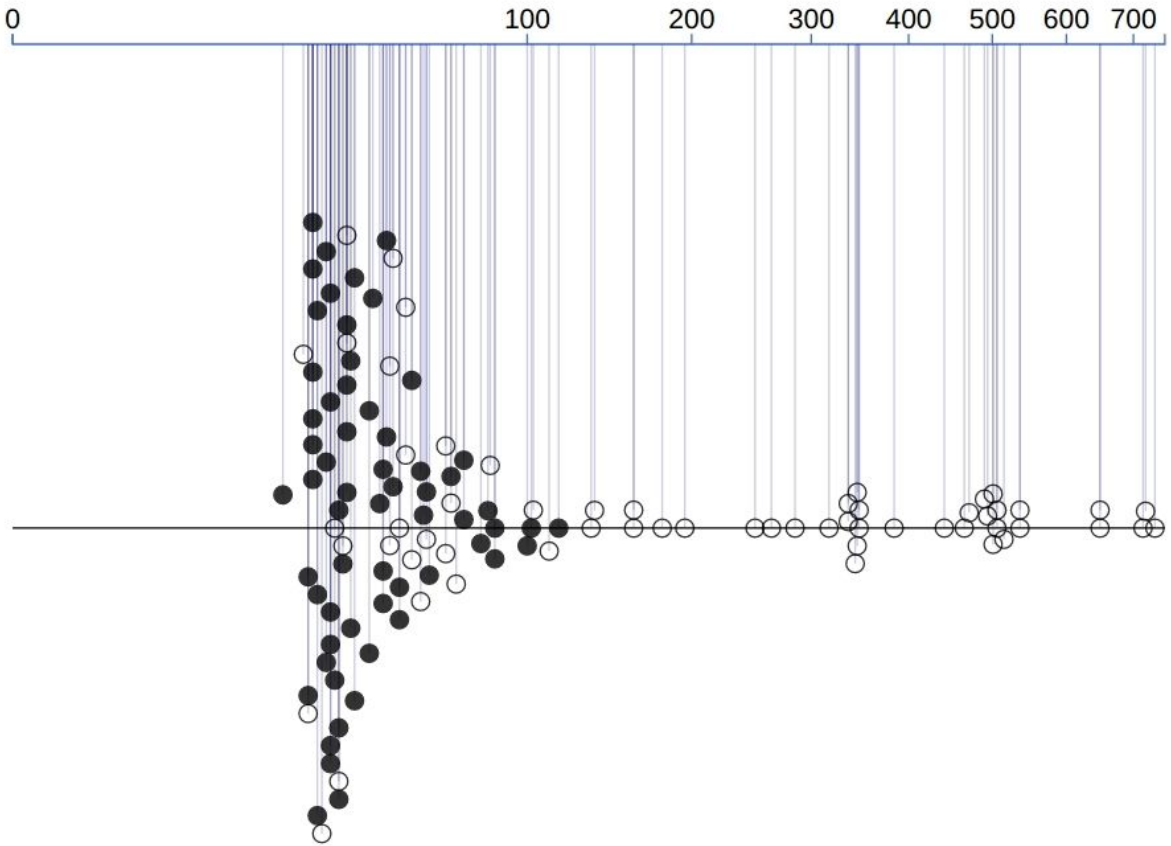
Puppet representing Abimanyu.

What to do if you don’t have enough puppets?

Abimanyu must always be represented by this puppet.

But other characters, such as Angganjani could also be represented by this puppet

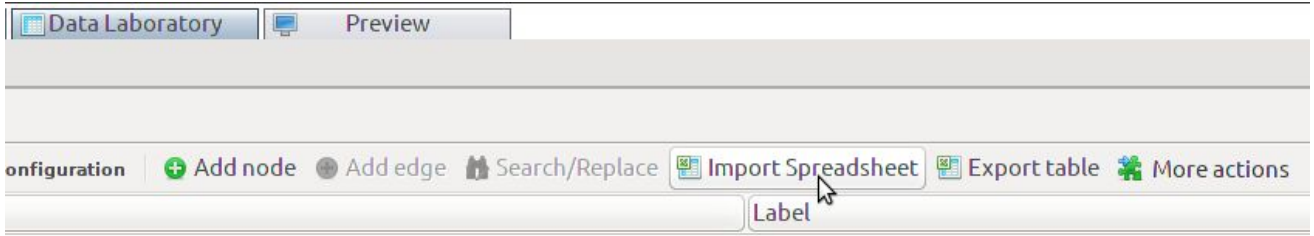
Characters with lower degrees are interchangeable in performance, high degree characters are not.



The different characters according to their weighted degree. The x axis is in an exponential scale (factor = 0.3). Black circles are those where puppet interchange is permissible and white circles are those where it is not.

Part 2 Working with Gephi

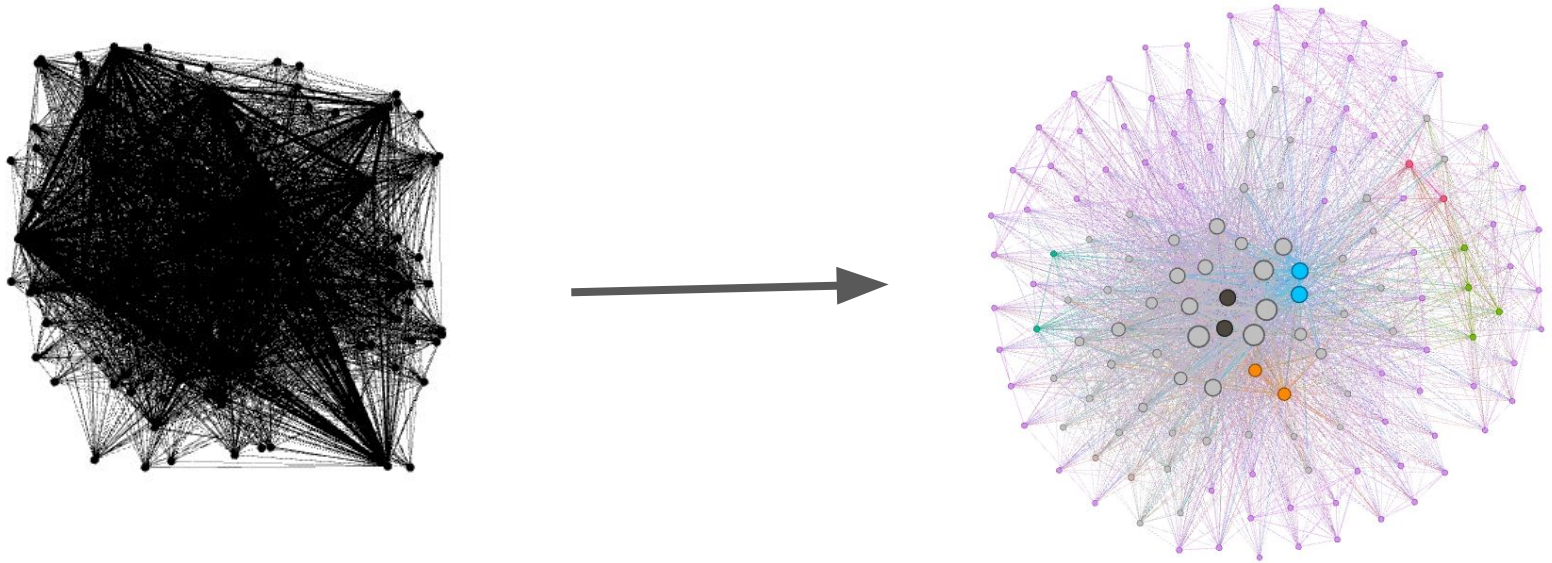
Importing the data



Import gephi/data/wayang_edgeList.csv

Make sure you include ‘weight’ and ‘type’ data

Visualizing the network

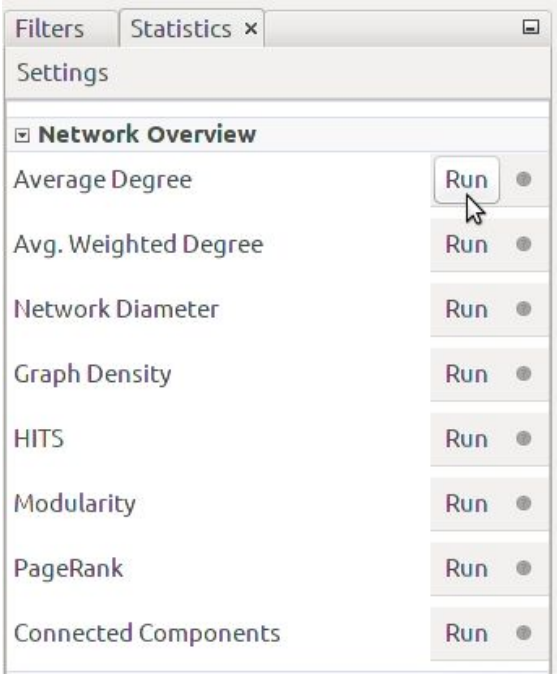


We can apply some rules for color, spatial distribution and size to make the network diagram easier to understand.

Let’s run some statistics

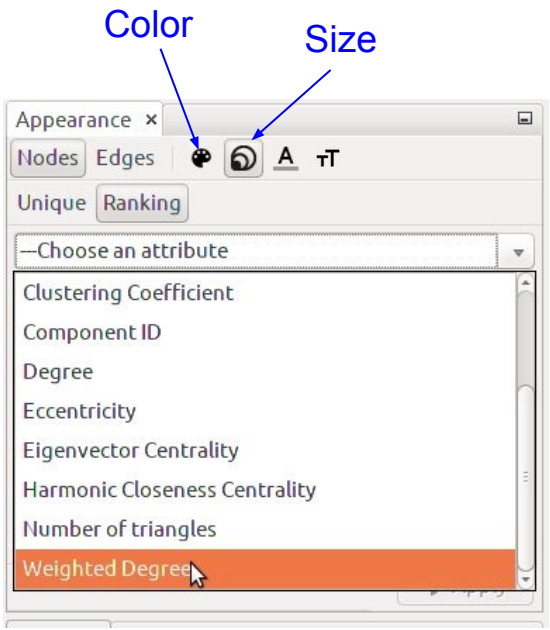
Use of these measurements

- They will help us better understand the network.
- Useful for styling (visual rules based on measurements).
- Will be eventually exported into our web project.



Styling the diagram

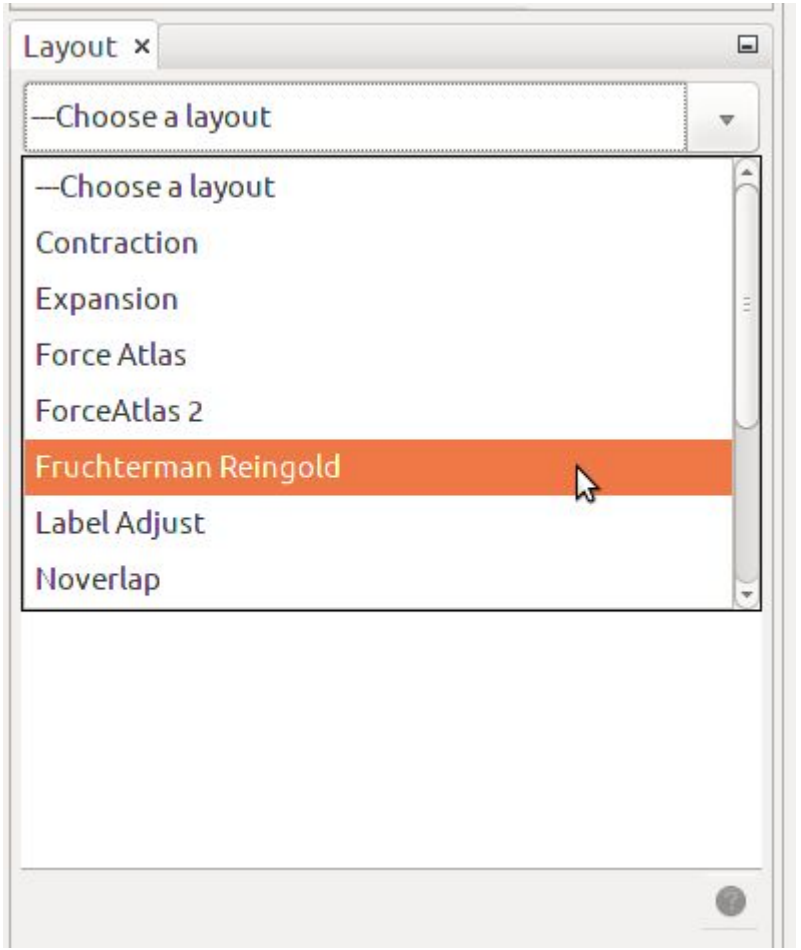
We can choose color and size parameters based on the measurements we calculated



Automating the layout

Different algorithms can help with the spatial arrangement of the nodes

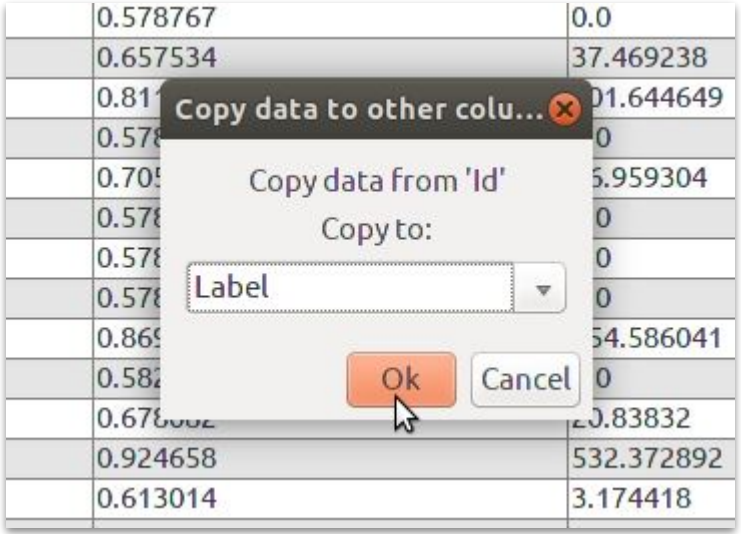
* This is purely for visual convenience and the arrangement has no network-theoretical meaning



Adding text labels

1) In the **nodes** table of the data laboratory tab, copy the ID to the Label column
2) Back in overview tab, select “Show node labels”

1



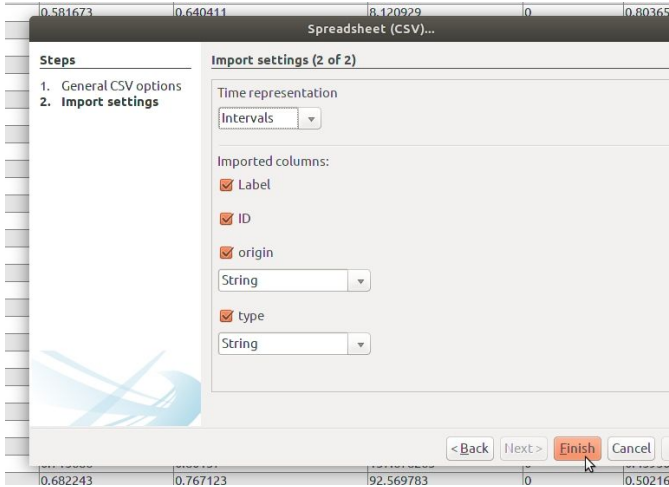
2

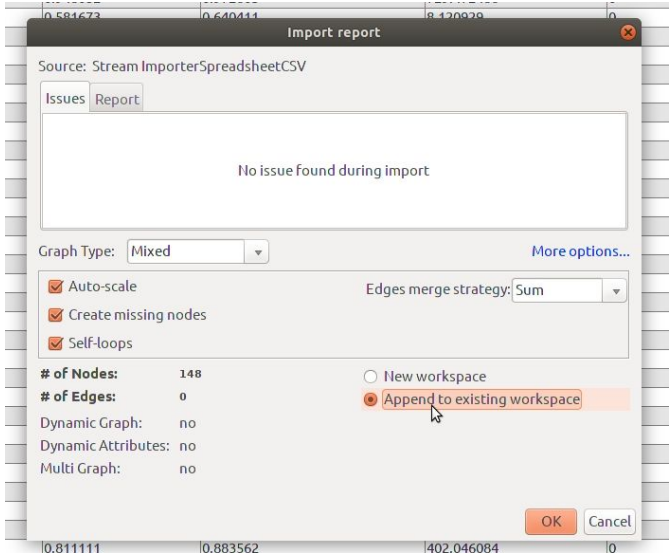
Importing additional metadata

In the Data Laboratory tab, Nodes table, click “Import Spreadsheet” and select gephi/data/wayang_nodeInfo.csv

Select all the available columns.

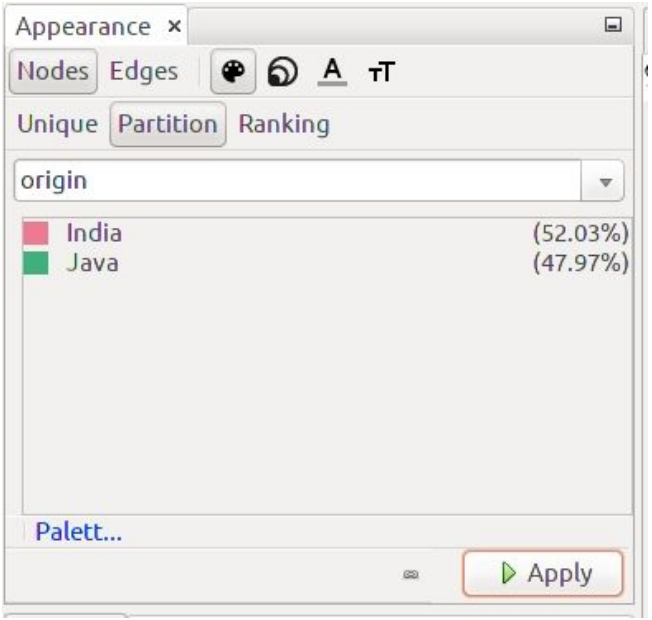
Make sure you select “append to existing workspace”.





Styling with additional metadata

We can determine the node color based on the origin of the wayang characters (India or Java)

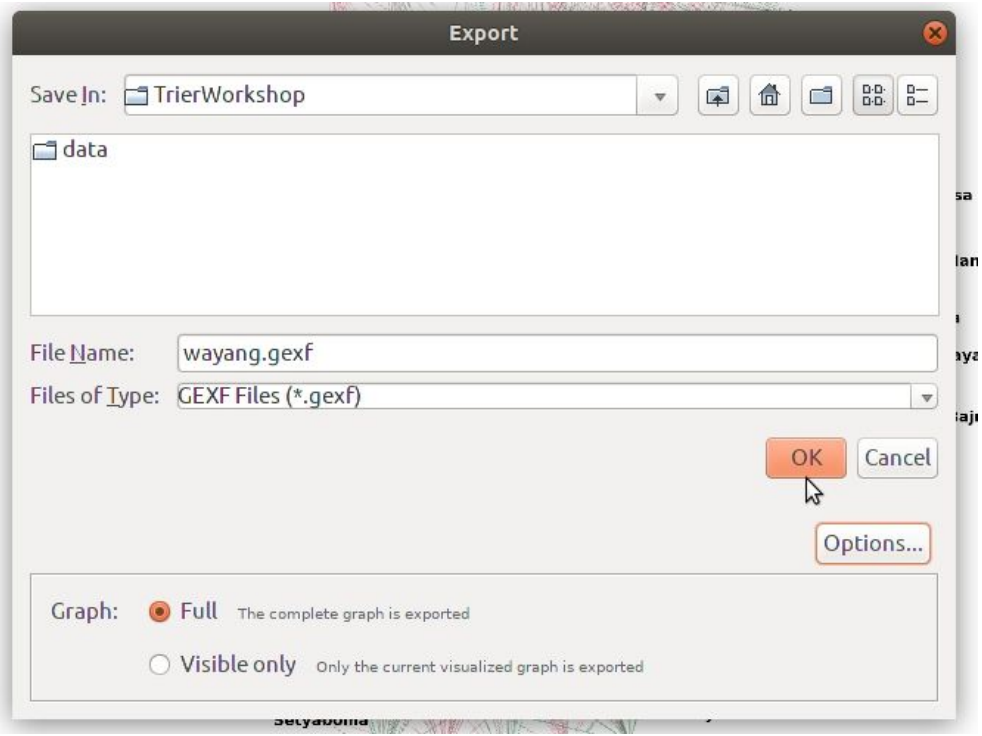


Exporting the data

File -> Export -> Graph file…

Make sure you choose the GEXF format.

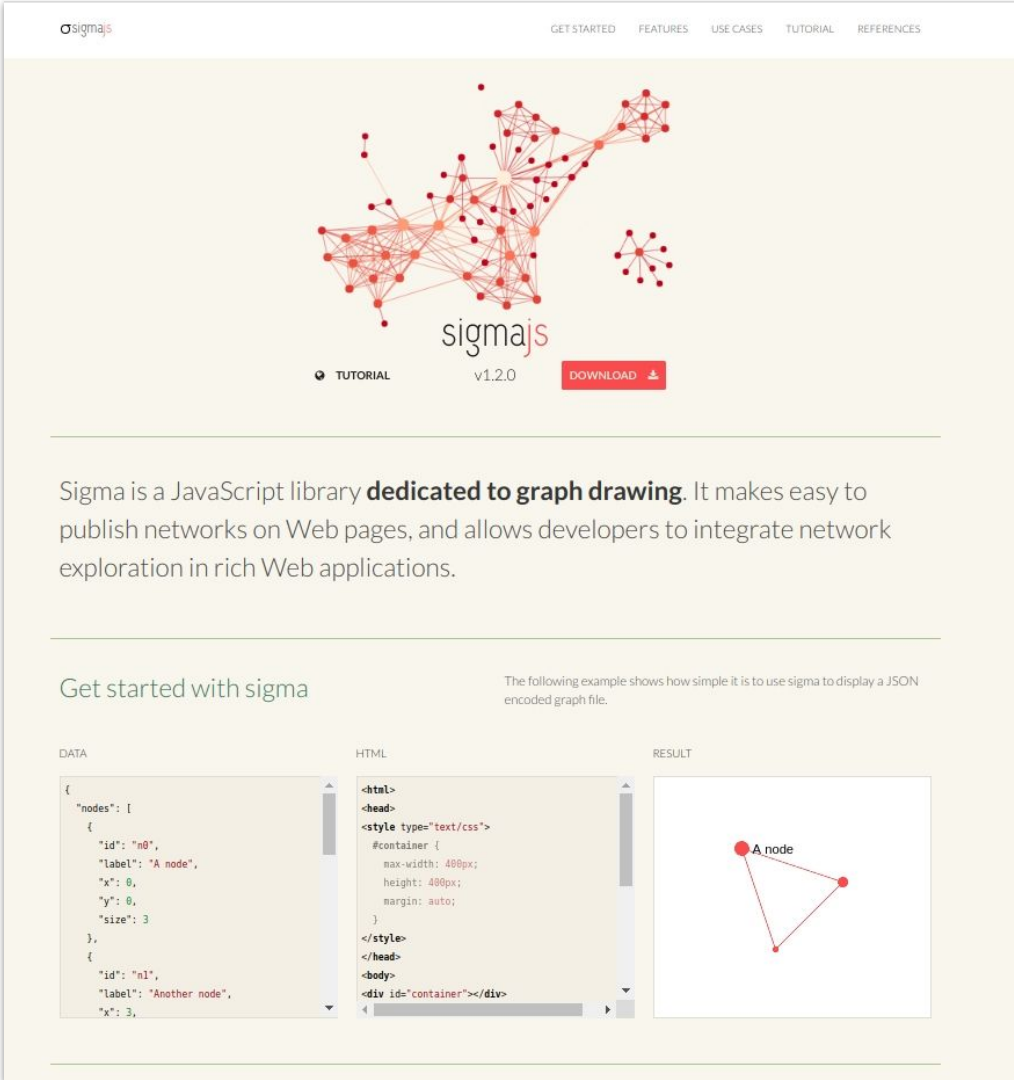
Save it inside html/data/ (different from gephi/data/)



Part 3 Creating an interactive web visualization

Sigma.js

http://sigmajs.org/



Structure of the gexf parser

The gexf method takes three arguments:

1) the URL of the gexf file
2) An object with:
    a) The ID of the HTML container for the graph
    b) An object with additional settings
3) Additional functions

```
sigma.parsers.gexf(
    'yourFile.gexf',
    {
        container: 'ID of html container',
        settings: {/*additional settings*/}
    },
    function(s) {/*additional functions*/}
);
```

Loading local files

Option 1: Use your localhost, if you have configured this

Option 2: Upload to a remote server (i.e., GitHub pages)

Option 3: Run your browser with flags:

Windows:

Run -> chrome.exe --allow-file-access-from-files

MAC:

Open terminal -> open /Applications/Google\ Chrome.app --args --allow-file-access-from-files

*Make sure you close all instances of Chrome before doing this

```
<!DOCTYPE html>
<html>
<head>
<style>
</style>
</head>

<body>

<div id="sigma-container"></div>
<div id="infoWindow"></div>

<script src="js/jquery.js"></script>
<script src="js/sigma.js"></script>
<script src="js/sigma.parsers.gexf.min.js"></script>

<script>

//Additional method to determine a node's neighbors
sigma.classes.graph.addMethod('neighbors', function(nodeId) {
    var k,
    neighbors = {},
    index = this.allNeighborsIndex[nodeId] || {};

    for (k in index)
        neighbors[k] = this.nodesIndex[k];
        return neighbors;
});

//This is where we load the Gephi file
sigma.parsers.gexf(
    '',
    { // Here is the ID of the DOM element that will contain the graph:
        container: '',
            settings: {
                minNodeSize: 2,
                maxNodeSize: 10,
    }
    },
```

Sample html file

Open html/index.html in your chosen editor

Point to the correct data file and ID of the HTML element

Your data file: data/wayang.gexf

The container id: sigma-container

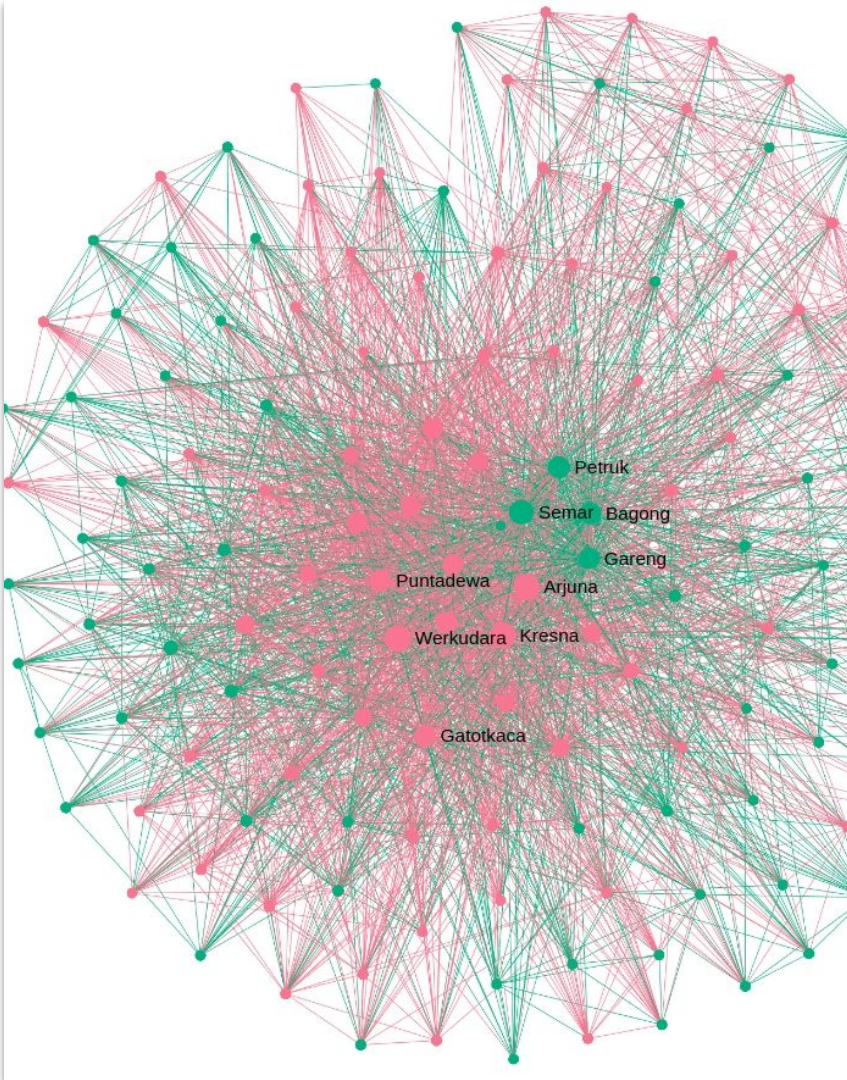
The diagram isn’t visible!

Make sure you give it dimensions via CSS

For example:

```
#sigma-container{
        position:relative;
        width:800px;
        height:800px;
}
```

See working file at html/index2.html



Additional settings

https://github.com/jacomyal/sigma.js/wiki/Settings

defaultNodeColor	String	`#000`	The default color of nodes
defaultLabelSize	String	`14`	The default size of text used to draw node labels
edgeColor	String	`"source"`	When no `color` property is defined on the edge, determines how to determine the color. `"source"` and `"target"` use that node's color, respectively, while `"default"` just uses the **defaultEdgeColor** setting.
minArrowSize	Number	`0`	Defines the minimal edge's arrow display size.
font	String	`"arial"`	Defines the default font
fontStyle	String	`""`	A CSS descriptor for drawing the chosen font
labelColor	String	`"default"`	If set to `"node"`, the label will be drawn as the same computed color for the node. Otherwise, the default label color is used.
labelSize	String	`"fixed"`	Indicates how to choose the label's size. Avaliable values: `"fixed"`, `"proportional"`
labelSizeRatio	Number	`1`	The ratio between the font size of the label and the node size.
labelThreshold	Number	`8`	The minimum size a node must have on screen to see its label displayed. This does not affect hovering behavior.
webglOversamplingRatio	Number	`2`	The oversampling factor used in the WebGL renderer.

Adding interactivity

Add the code just before s.refresh();

```
90              //colors for the edges included in th
91              s.graph.edges().forEach(function(e) {
92                  if (toKeep[e.source] && toKeep[e
93                      e.color = e.originalColor;
94                  }
95                  else{
96                      e.color = '#eee';
97                  }
98              });
99
100             //display additional information
101             explanation = "<p><h1>" + e.data.node
102             explanation += "<p><br/>Degree: " + e
103             explanation += "<br/>Weighted degree
104             explanation += "<br/>Eccentricity: "
105             explanation += "<br/>Closeness centra
106             explanation += "<br/>Betweenness cent
107             explanation += "<br/>Clustering: " +
108             explanation += "<br/>Origin: " + e.da
109             explanation += "<br/>Type of characte
110             $("#infoWindow").html(explanation);
111
112             // Since the data has been modified,
113             // call the refresh method to make th
114             // update effective.
115             s.refresh();
116         });
```

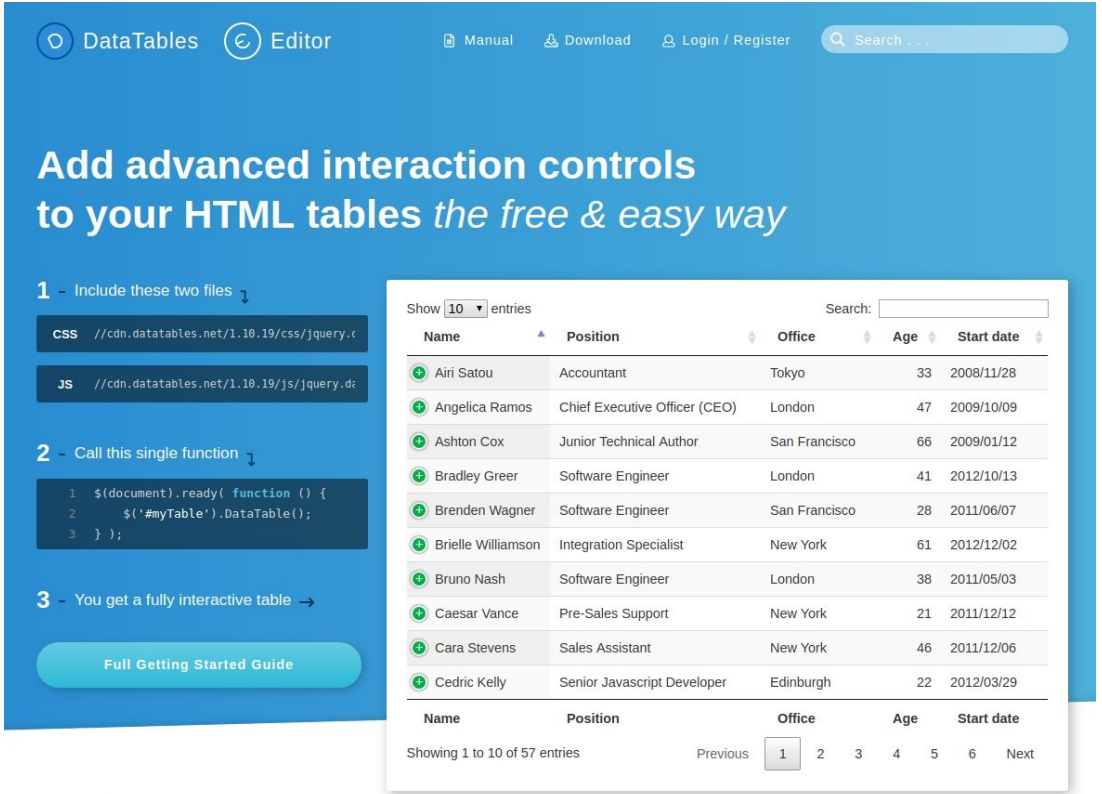
Adding interactivity

- To navigate the XML file:
  e.data.node.attributes.**YourAttribute** or e.data.node.attributes.[“**Your Attribute”]**
- You can reference measurements calculated by Gephi as well as metadata that you imported
- Use the JavaScript function .toFixed(n) to limit the numbers to n decimal places
- Use the jQuery function $(“#yourDivID”).html(yourVariable) to display this additional information

```
//display additional information
explanation = "<p><h1>" + e.data.node.label +"</h1>";
explanation += "<p><br/>Degree: " + e.data.node.attributes.degree;
explanation += "<br/>Weighted degree: " + e.data.node.attributes["weighted degree"];
explanation += "<br/>Eccentricity: " + e.data.node.attributes.eccentricity;
explanation += "<br/>Closeness centrality: " + parseFloat(e.data.node.attributes.closnesscentrality).toFixed(2);
explanation += "<br/>Betweenness centrality: " + e.data.node.attributes.betweenesscentrality.toFixed(2);
explanation += "<br/>Clustering: " + e.data.node.attributes.clustering.toFixed(2);
explanation += "<br/>Origin: " + e.data.node.attributes.origin;
explanation += "<br/>Type of character: " + e.data.node.attributes.type;
$("#infoWindow").html(explanation);
```

Visualizing the data as tables

https://datatables.net/



Adding interactive tables

Export table from Gephi as .csv file

Modify the code in table.html to load your CSV file as a JSON object

```
<script src="js/jquery.js"></script>
<script src="js/dataTables.min.js"></script>

<table id="dataTable" class="display" cellspacing="0" width="100%">
    <thead>
        <tr>
        </tr>
    </thead>
</table>

<script>
//container JSON object
var json = {"data":{}};

$.ajax({
    url: "data/example.csv", //the name of the csv file
    async: false,
    dataType: "text",
    success: function (csv) {
        toJSON(csv);
    }
});

function toJSON (data){
    lines = data.match(/[^\r\n]+/g);
    //populate the array
    dataArray = []
    var i;
    for (i= 1; i<lines.length; i++){
        dataArray.push(lines[i].split(","));
    }
    json["data"] = dataArray;
    //make the table
    headers = lines[0].split(",");
    for (j in headers){
        $('<th></th>').text(headers[j]).appendTo($("#dataTable>thead>tr"));
    }
    table = $("#dataTable").DataTable(json);
}

</script>
</body>
</html>
```

Additional exercise

Do you have any network data to visualize interactively?

You can download data from this source: https://dracor.org/

Troubleshooting

On Ubuntu 18.04 default Java is installed via packages: default-jre, default-jre-headless

These depend on: openjdk-11-jre, openjdk-11-jre-headless, which are Java 10.

With such packages installed, no charts are plotted, and the Options > Visualizations > OpenGL page shows no detail of graphic card and drivers.

Reverting back to Java 8 restores Gephi functionality:
sudo apt remove openjdk-11-jre openjdk-11-jre-headless
sudo apt install openjdk-8-jre