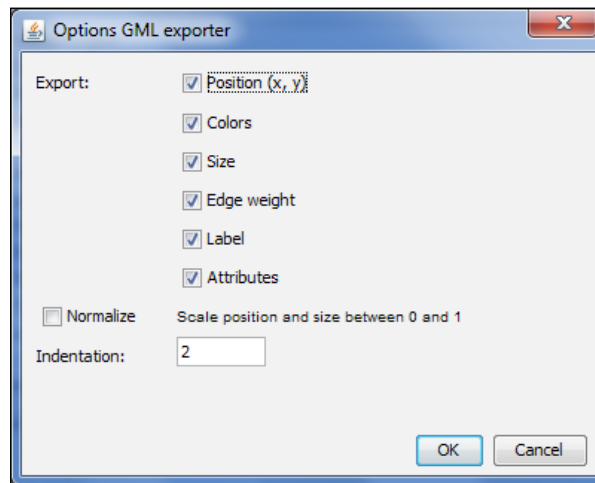


## GML files

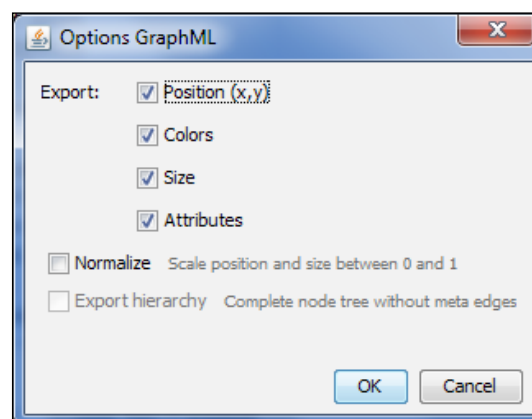
**Graph Modeling Language (GML)** is a hierarchical ASCII-based format for describing graphs. The syntax bears a very slight resemblance to JSON. The export options are nearly identical to those of GEXF:



GML file options

## GraphML files

GraphML is another XML-based format, so it will look very familiar to some users. There are specific elements (at the graph, node, and edge levels) that make the syntax very easy to interpret. Gephi offers partial support for this format, but once again allows users to export their graph files to GraphML with the following options:

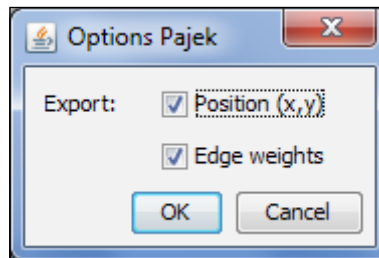


GraphML file options

For the full GraphML specification, visit the GraphML site at <http://graphml.graphdrawing.org/>.

## NET files

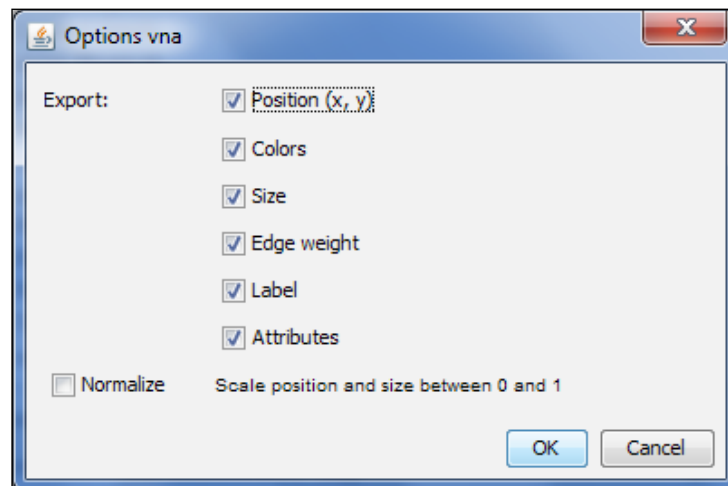
One of the most prominent network graph solutions is the Pajek program, found at <http://pajek.imfm.si/doku.php?id=pajek>. Pajek uses the .NET format, which Gephi provides two simple export options for:



NET file options for Pajek

## VNA files

The VNA format is used by a program titled NetDraw and resembles the Pajek .net syntax. Gephi provides a number of export options for this, which are as follows:



VNA options for NetDraw

To learn more about .vna and NetDraw, visit <https://sites.google.com/site/netdrawsoftware/home>.

Before moving to our next section, note that Gephi also provides plugins that enable geo-based data exports for Google Earth (.kml/.kmz data), as well as shapefile exports that can be used in GIS software as QGIS or MapWindow. For more information, visit the Gephi Marketplace and search for the ExportToEarth, Export to SHP, and Google Maps Exporter plugins. The main requirement to use these is the presence of latitude and longitude (lat/lon) attributes in your network data.

## Image exporters

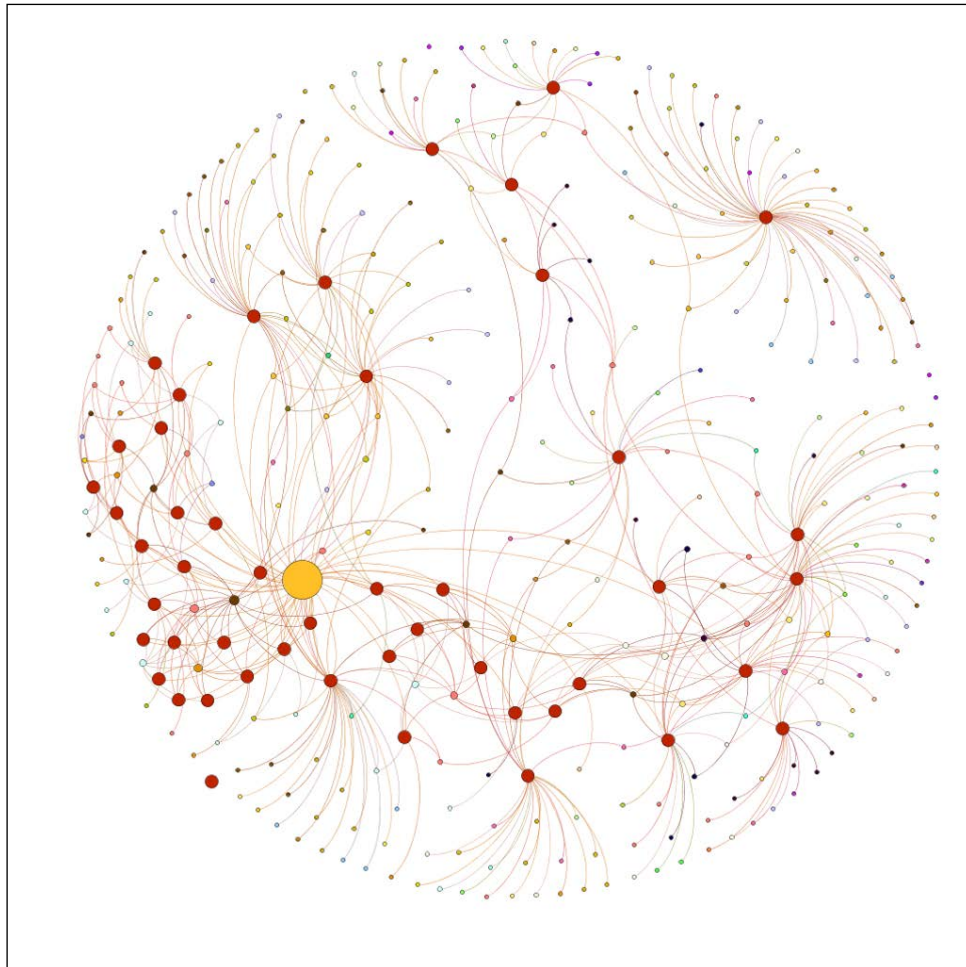
In some cases, your goal will be to produce finished graph output that does not require users to view a network using Gephi or other network graph software. In many of these instances, the graph might be intended for a print publication, or you might simply want to create a specific view of your network for sharing via social media or e-mail. For these cases, the three image export options provided by Gephi offer easy solutions to your needs. Let's look at each of these options, with a focus on the benefits and limitations of each.

### PNG export

When your goal is to simply share a static version of an existing graph project, creating a .png file is hard to top. This will be the quickest means to share your work with others, with the understanding that the file is not designed for further interaction. On the plus side, .png files scale well and are easily displayed on a blog or website.

Let's have a quick look at the .png export process, which can be initiated by navigating to **File | Export | SVG/PDF/PNG file**. Alternatively, if you are in the **Preview** window, there is a **SVG/PDF/PNG** button near the bottom of the screen that will execute the same process. Suppose we take our example of Miles Davis network and elect to create a .png version for quick sharing on a blog or through social media.

We're going to use the **Preview** window and set **Preview Settings** to use the **Default Curved** option, which will display better in this context (versus the black background I used for the Web). Now refresh the window to display with the new format and then export the data to .png using the advice from the previous paragraph. Save the file and then open it in any program where you can view image files. Here's our output image, which should look precisely like the version from the preview window:



PNG export of Miles Davis network

As you can see, this results in a high-quality image of the network that can be shared through various channels.

## SVG export

If you need to move beyond the fixed image provided by a .png export, then you might consider the **Scalable Vector Graphics** (.svg) option. SVG provides an XML-based format that can be edited using either text editors or the more common, image editing software. SVG is also suitable for the Web, as it scales well when zooming or panning a visualization.

In this section, we'll take the same Miles Davis network, save it as a .svg file, and then do some editing using the open source Inkscape program. Following the steps we take might help you determine whether .svg is the best option for post-Gephi editing, or whether you would prefer .pdf, which will be covered in the next section.

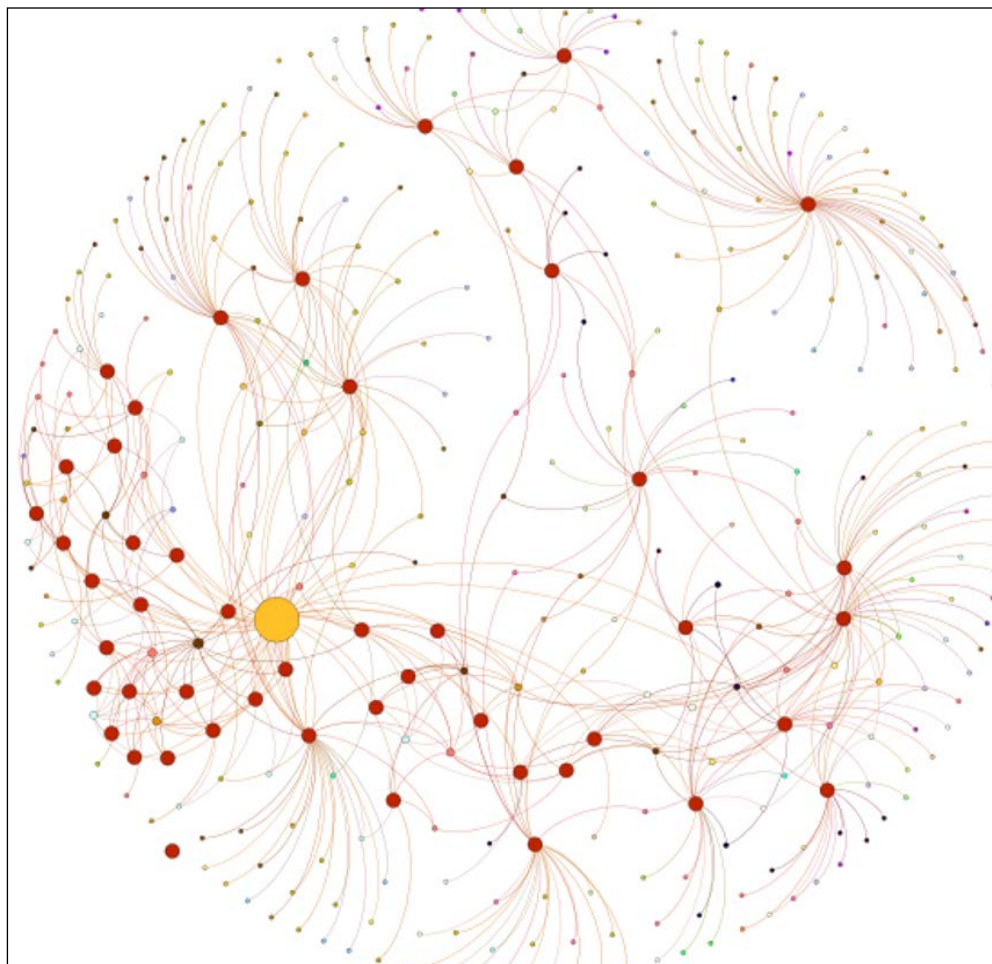
## Editing an SVG file with Inkscape

So let's begin by repeating the file saving process, but by electing the SVG option in place of PNG. After saving the file, we're going to open it in Inkscape. Feel free to follow along with your own version of Inkscape, or by following a similar process in Adobe Illustrator or any other image editing software.

We already noted that these programs can be used to finish what we already started in Gephi, especially if we are seeking to create a finished graph for print purposes. We could simply leave the graph alone while adding text and titles in the editing software, or we can use the tool to truly edit the graph itself. I'm going to showcase each of these options.

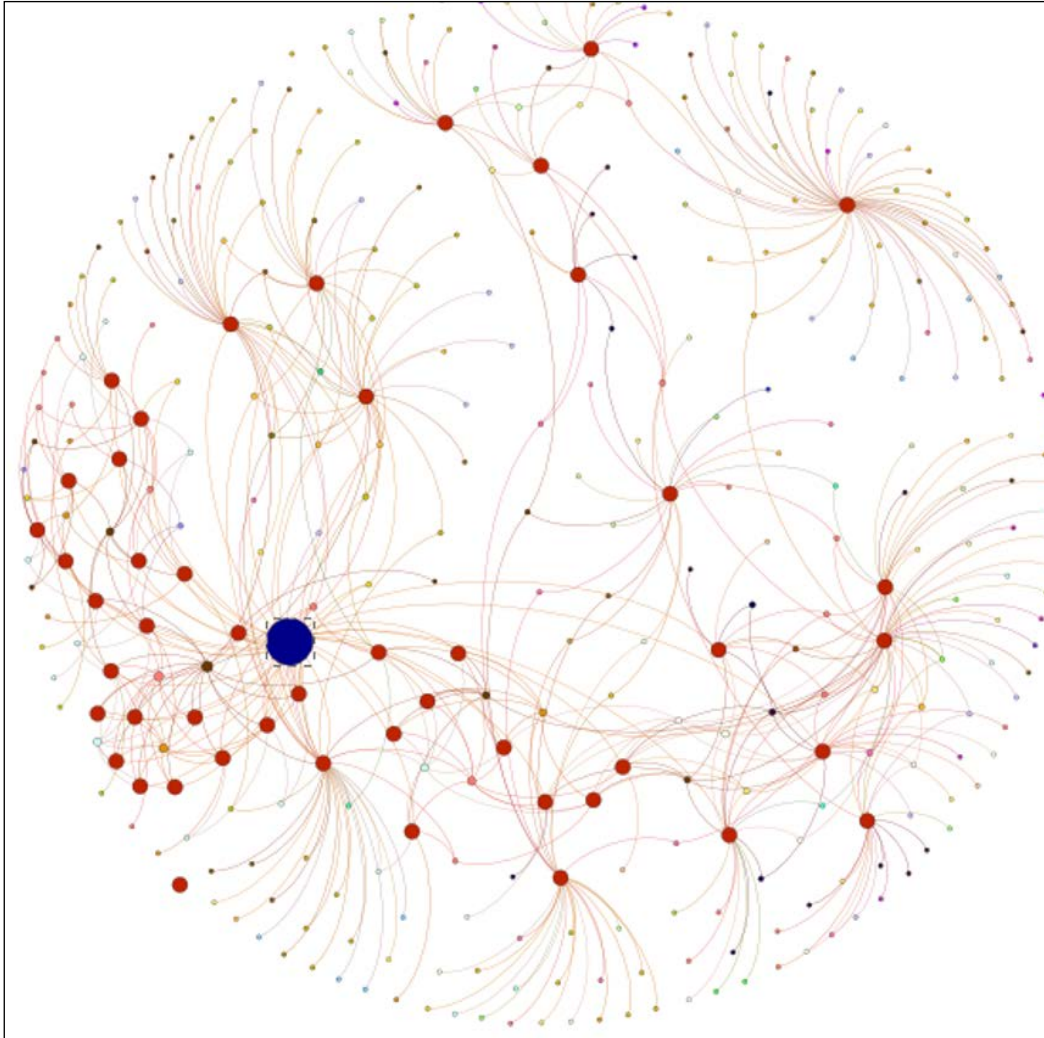
We'll begin with the actual editing of our graph, which can be done by ungrouping the graph elements. This will allow us to edit each and every element in the graph, although it is far more likely that we will want to call out just a few special highlights. So with the .svg file created, we'll open it in Inkscape, which will give us the same sort of view we saw with the .png output.

The key step comes when we click on the **Edit Path by Nodes** icon (or the *F2* key). This allows us to start selecting individual graph nodes and edges for editing. So we will start with a network like this:



SVG export of Miles Davis network

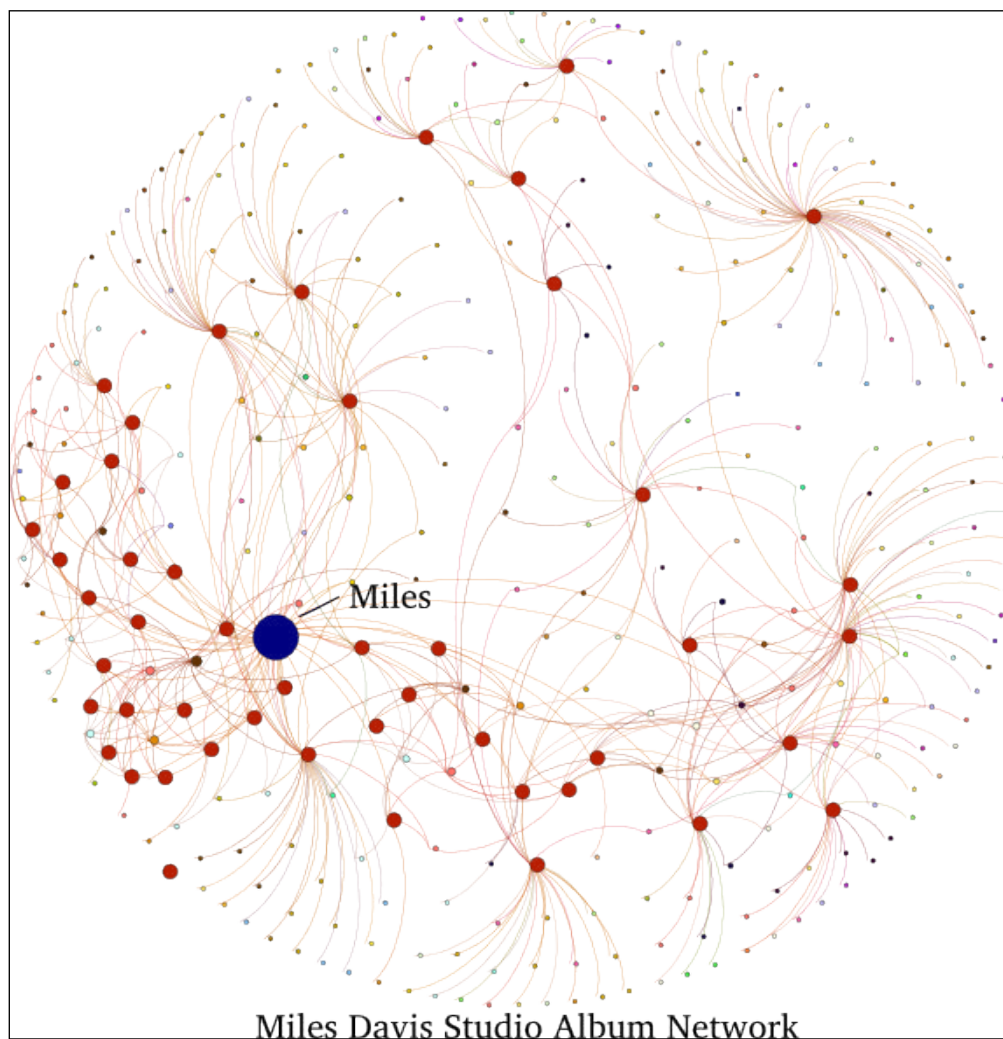
Suppose that we are not pleased with the yellow shade of the Miles Davis node and would like to change it to blue. All we need to do is select the node using the mouse and then select a new color from the palette at the bottom of the Inkscape window, which will result in the following graph:



SVG export edited using Inkscape



Let's make a couple more edits to demonstrate how easily text can be added to the graph. Using the **Create and Edit text objects** icon (or the *F8* key) from the Inkscape toolbar, we'll add a small title beneath the network followed by a large label calling out the Miles Davis icon, just in case it isn't obvious enough from the large size and distinct color. We'll also throw in a line as a pointer for the label. Here's our enhanced graph:



Adding text and titles with Inkscape



This should give you some idea for what can be done to enhance Gephi networks using image editing software. While these were simple edits, there is clearly potential for much more.

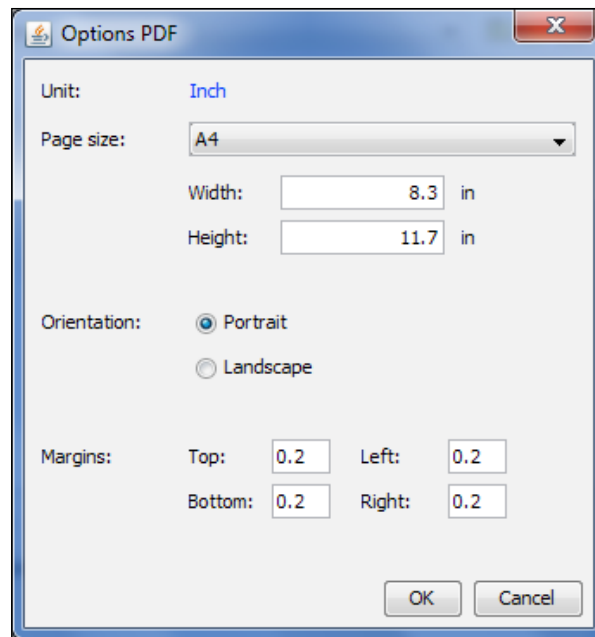
## PDF export

The Adobe Acrobat PDF format has long been a standard for sharing files on the Web or through e-mail, as it lets all the end users afford the ability to view output without requiring multiple software platforms on their own machine. For the purposes of the Gephi user, it serves as an output option that can produce high quality files for end users to view a network graph. That in itself might be enough to recommend this option, but there is a potentially much more valuable function served by the PDF format, which we'll now discuss.

PDF files can be readily edited using applications such as Adobe Illustrator or its open source alternative, Inkscape. This makes the PDF format highly useful in cases where you wish to tweak your graph beyond Gephi. Much of what you can do within these software platforms is technically possible in Gephi, such as customizing the size and coloring of specific graph elements. However, the ability to add titles and text, strategically position labels, and add images, among a host of other possibilities, makes the PDF export option very useful for Gephi content producers. By the way, there is a Gephi plugin in the early development stages designed for a similar purpose—the **Image Preview** tool, found at <https://marketplace.gephi.org/plugin/image-preview/>.

We'll follow a similar process to what we did with the SVG files in Inkscape. One primary difference is the way in which items are ungrouped; whereas all elements were immediately available using the SVG approach, the PDF might differ somewhat, requiring a few steps to ungroup the file (depending on the export settings). Note that there will often be some transparent layers that need to be removed before all elements can be completely ungrouped.

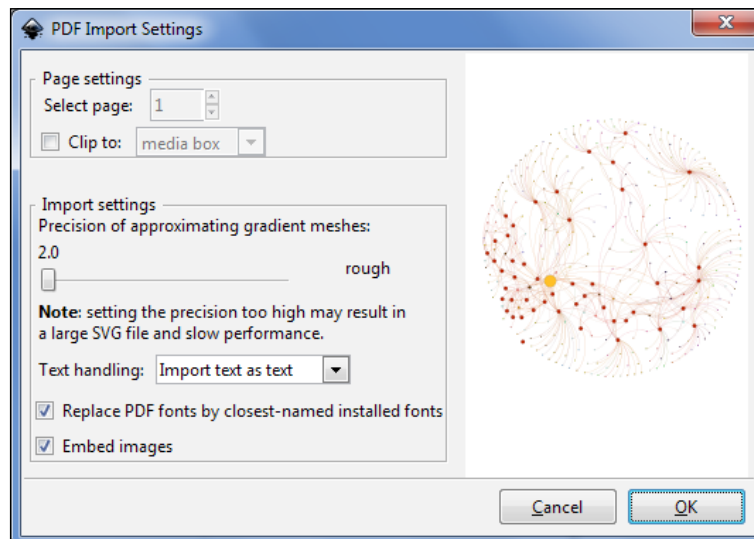
When you elect to use the PDF export, there are some available options, mostly with respect to formatting of the file, as shown in the following screenshot:



PDF export options

## Editing a PDF file in Inkscape

Let's pick it up with the PDF file created and ready for manipulating in Inkscape. When we open the file in Inkscape, another dialog screen pops up, which is as follows:



PDF import in Inkscape

Here we can determine how to handle text, whether we should clip the file on import, and how precise we want the file (which will have some cost in the file size).

In this case, I'll go with the default options shown in the preceding screenshot, which will result in a file very similar to what we saw with SVG. We can now begin editing the file in the same fashion we did previously.

## Web exporters

Several Gephi plugins can be used to create interactive graphs on the Web. We'll focus on three of these, in order from simplest to most complex:

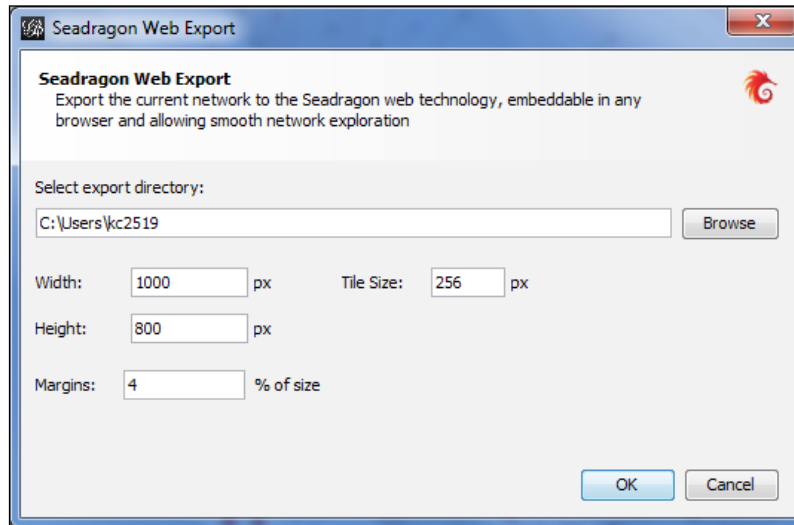
- **Seadragon Web Export:** This uses the Seadragon software's zoom capabilities to enable traversing large networks using zoom and pan capabilities
- **Sigma.js Exporter:** This is based on the `Sigma.js` software, which facilitates the creation of interactive web-based network graphs using a template-driven approach
- **Loxa Web Site Export:** This also uses `Sigma.js` as the basis for a slightly more sophisticated graph implementation that enables user interaction using filtering and zooming plus the possibility of multiple layered views

Now let's take a tour through each of these options, detailing how they work and what they can deliver for a finished project.

## Seadragon Web Export

The Seadragon option is ideal if you want to provide a modest level of interactivity for users. Zoom and pan are the primary features of Seadragon, which makes it suitable for navigating large networks that would prove difficult to decipher as a static graph. What you won't get is any sort of statistical output or additional information about the network.

Let's take a look at the basic options provided by the Seadragon plugin, and then we'll walk through an example later in the chapter. When we select the Seadragon option, this screen pops up:



Export dialog box for Seadragon plugin

Here's where the dimensions of the graph can be specified, ideally well suited to the needs of the user. Remember that users can zoom and pan, so there is no need to make the original graph larger than necessary. After selecting the **OK** button, Gephi will export the project to the specified folder location.

We'll see what the output actually looks like, as well as how to interact with it in an upcoming section.

## Sigma.js Exporter

If Seadragon doesn't provide the capabilities you're looking for, then it might be time to step up to SigmaExporter, which allows you to create templates that can be used as the base for multiple network instances. We'll spend plenty of time with this option, as it delivers a powerful experience for web users.

Let's assume that we wish to export an existing Gephi network to the Web using SigmaExporter. We'll begin by navigating to the **File | Export | Sigma.js template** menu. The template approach allows us to create specific reference text that can be used again and again, tweaking as needed. Here's what we see after making the menu selection:

**Sigma.js Export**

**Export to Sigma.js template**

C:\Users\kc2519\Documents\Business\Visual-Baseball\Applications\Gephi\ Browse...

**Legend**

Node\* Musicians

Edge\* Played on a Record

Color\* Instrument

**Branding**

Logo (url)

Link

Author\* Ken Cherven

Title\* Miles Davis Studio A

**Features**

☒ Include search? ☐ Group edges by direction?

Hover behavior Dim

Group Selector? None

Image attribute? None (Default)

**Attributes**

Coming soon

**Short Description\***

The Miles Davis discography browser shows all 48 studio albums recorded by the legendary Miles Davis. Miles is a display, surrounded by nodes representing each of the recordings, and all of the musicians who were part of an

**Long Description\***

<p>Weight is simply a measure of the number of albums played on by a musician/instrument combination. Node sizes reflect this value. </p>

<p>A degree refers to a connection to another node, so the degree measure provides a total count of the number of connections one node possesses. </p>

<p>Closeness Centrality is a measure of the average distance a node is from all other members of the network, using the shortest distance path. Therefore, a lower

</h4>

☐ Replace node ids with numbers

OK Cancel

Export options for SigmaExporter

First, we need to specify where the file will be saved. Be careful not to overwrite prior exports, which will be done by default. To avoid this, you can simply add a subfolder for each export within your default directory.

As you can see, there are multiple sections where values can be set, beginning with the **Legend** section. Each of these settings is mandatory for good reason—they deliver critical information to the graph viewer. The options in the **Legend** section can be described as follows:

- In the **Legend** section, there are three options, starting with the **Node** setting, which is essential for the display. This is used to tell viewers what the nodes represent—musicians in our case, but this could just as easily be individuals, places, or dozens of other possibilities.
- The **Edge** setting is also required, and is used to explain what the network connections indicate. In this example, we will specify that edges indicate that musicians *Played on a Recording*.
- We will also use **Color** to encode graph information—in this case it represents the *instrument* a musician played.

Several **Branding** options (for personalizing the graph) are also offered, which are especially useful when you want to link back to your own website or use your own logo. This is also where you specify author and title information; after all, you should get some credit after you've spent time creating a great Gephi visualization.

Several selections can be made in the **Features** portion of the template as well. We'll summarize those here:

- Our first selection, **Include search** allows us to provide users with search capability, which is especially useful when deploying a large network. We'll see just how useful this can be in the next section where we will actually deploy the network graph.
- In the case where we have a directed graph, we have the ability to **Group edges by direction**, making it easier for users to navigate the deployed network
- There is also an option for **Hover behavior**, a simple setting that determines what occurs when users hover over a node. The simple choice is to dim nodes, or to leave them in a normal state. The advantage of using the dim setting is the ability to dim the unneeded sections of the network, which allows a better focus on the relevant areas.

- The **Group Selector** allows you to partition the network using one of the available variables. This could be done using color, centrality measures (if available), or some other categorization.
- Finally, we have an **Image attribute** setting, in the event your network has an image you wish to utilize, based on one of the columns in your dataset. This image will then be used in the **Information Pane** area of the final graph.

Let's move on to a couple more useful options in the template. First up is the **Short Description** text area, where we can provide users with a synopsis of what the network is about. This is where you want to be concise in explaining the network – a couple of sentences should be sufficient.

Our last selection is one where we can really customize the look and feel of the graph, by providing a useful **Long Description**. As you might have noted from the preceding screenshot, HTML styling can be employed here to adjust text sizes, set paragraphs, and so on. This will help give your graph a polished feel by providing essential information for users seeking further information.

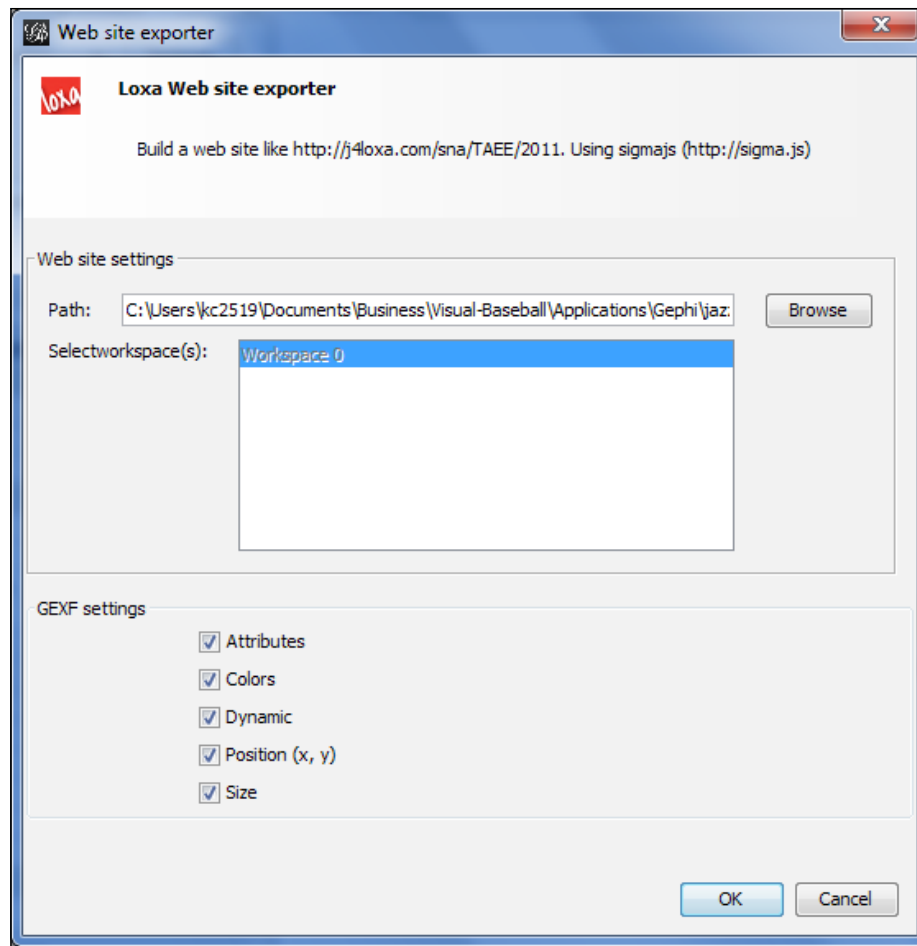
In the next section we'll see how to utilize these settings to great effect as we deploy an interactive graph.

## Loxa Web Site Export

Another available export choice is the Loxa tool, which uses Sigma.js while providing some alternative options not found in SigmaExporter. We'll walk through the available selections just as we did with the prior tools.

We'll begin again by initiating an export of the existing project, reviewing each of the options as we go. Navigate to the **File | Export | Web site exporter** menu item, which will give you the following screenshot:





Export options for Loxa Web Site Export

You'll notice the relative lack of options compared to SigmaExporter, although there are a handful of options at the bottom of the screen you can choose to export (or not) – **Attributes**, **Colors**, **Dynamic**, **Position**, and **Size**.

In contrast to SigmaExporter, much of the descriptive work for Loxa projects is done behind the scenes by navigating to your project export location. There you will find not only the primary display page (`index.html`), but also two output pages that can be edited in any text editor. The **About** page lets you provide basic information about you or your organization. The **Info** page is where the real meat of the information is supplied, which allows you to specify sources, objectives, metrics, references, and any other information deemed essential to the display.

We'll walk through the process of creating a custom info page when we build our network graph in the next section.

## Exporting a web graph

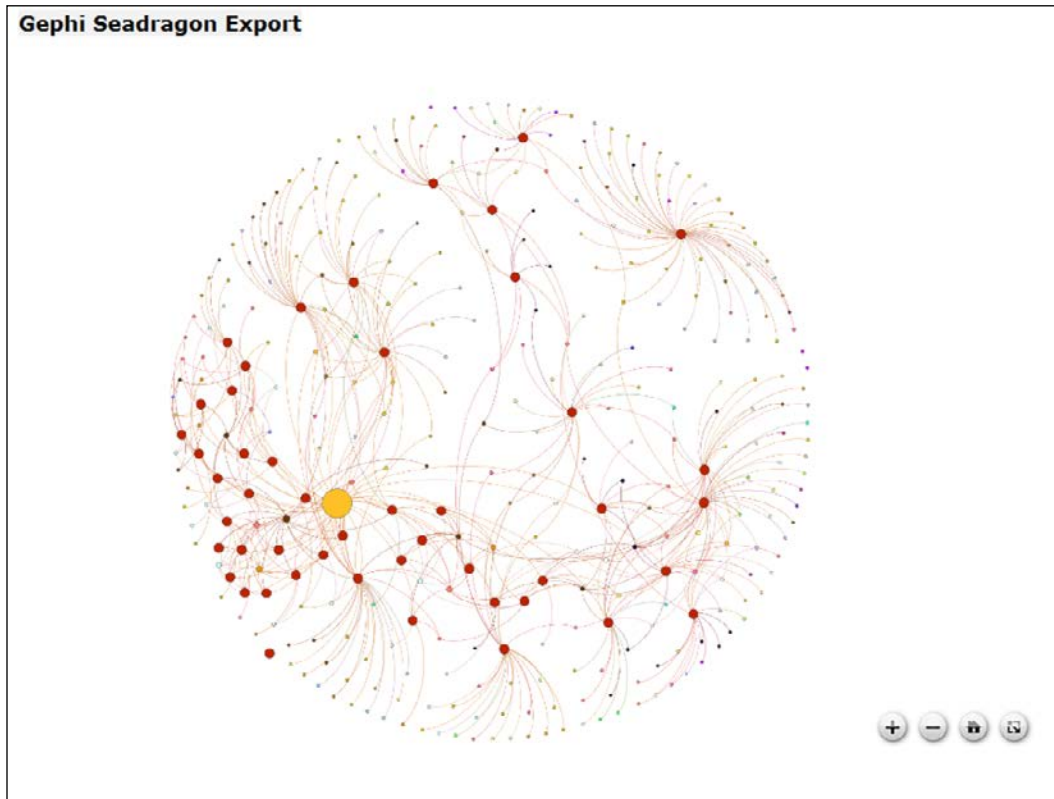
We've just learned about the three primary options for exporting an interactive graph to the Web. We'll walk through the process using each of the three choices. I'll also provide some examples of other graphs created using the **SigmaExporter** option, which I hope will give you an idea of the possibilities.

Let's begin with the Seadragon tool. If you want to follow along, make sure the plugin is installed in your version of Gephi. If not, download and install it using the Gephi plugins option from the **Plugins** menu under **Tools**. For this example, we'll use the familiar project I created that explores all 48 studio albums of jazz legend Miles Davis. The file can be found at <https://app.box.com/s/177yit0fdovz1czgcecp>.

## Seadragon

Our first example will be using Seadragon. Follow the process discussed earlier to export the graph by navigating to the **File | Export | Seadragon Web** menu selection. Set your location, size, and margin options in the dialog screen, and create the graph. Remember that if you export the project to a local file, some browsers will not open the graph due to their restrictions on JavaScript files. If you are a Chrome user, you'll want to switch to Firefox for this example, or you can use a local web server powered by Python or Java, as two possibilities.

Let's display the finished graph in the browser:



Network output in Seadragon

You can see the controls whenever you hover over the graph. The plus sign enables zooming in, while the minus sign does the reverse. The home icon will return the graph to its original centered position, and the final icon on the right lets you toggle between fullscreen and the original graph size.

Before we move on to the more powerful Sigma and Loxa options, I want to make you aware of the ability to do some style editing behind the scenes for your Seadragon file. All styling information is embedded directly in the `.html` file created by the Seadragon plugin, although you could create a separate CSS file to refer to if you wish. Here's what the style code snippet looked like when I first exported the above file:

```
<style type="text/css">
    body {
        margin: 0px;
    }
    #seadragon {
        width: 800px;
        height: 600px;
        background-color: Black;
    }
</style>
```

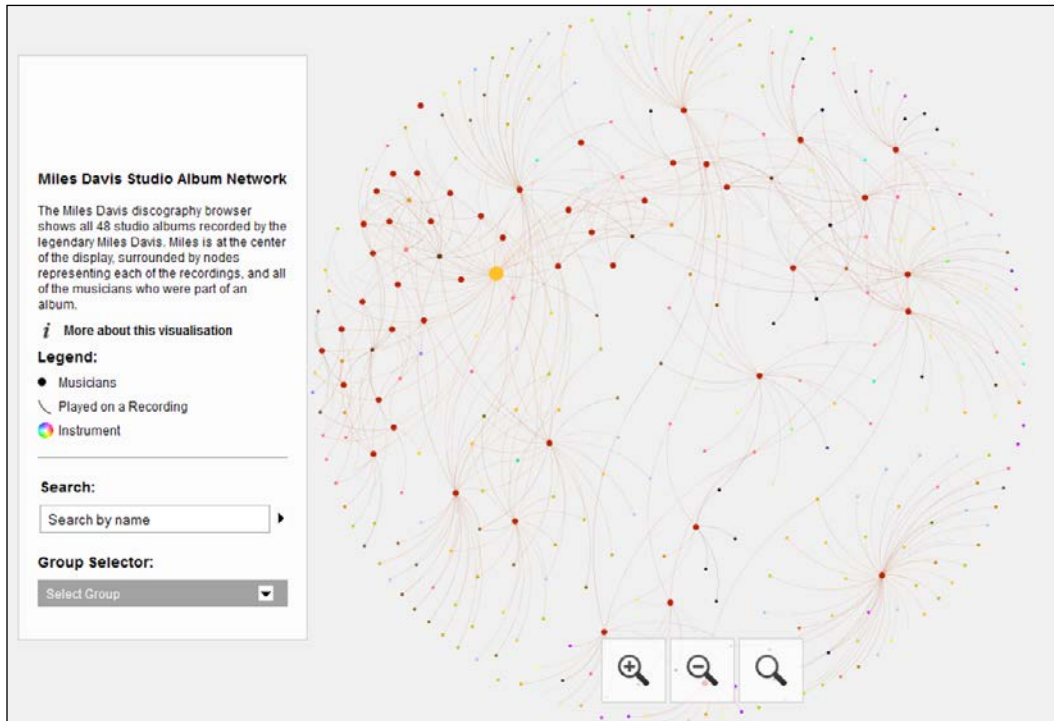
Here you see some very simple settings that instruct the browser how to display the page—simple width, height, and background color elements. If you have experience with CSS styling, feel free to add more elements using your favorite text editor, but given the relatively limited scope of Seadragon this might not be terribly useful. Perhaps a title element and a textbox or legend would be nice, but these will be easier to implement in the Sigma and Loxa solutions.

Seadragon is a fun tool for manipulating the graph, even if it falls short of the feature sets of the Sigma and Loxa exporters. However, if you want to create a quick, simple network graph for the web browser, Seadragon might be sufficient for your needs.

## **SigmaExporter**

Earlier we saw the many choices available for exporting a network graph using the SigmaExporter tool. Now we'll follow the process right through graph creation. We'll also use the group selection, search functionality, and information pane to learn more about the network. If you wish to follow along, make sure to install the SigmaExporter tool from the available plugins list in Gephi.

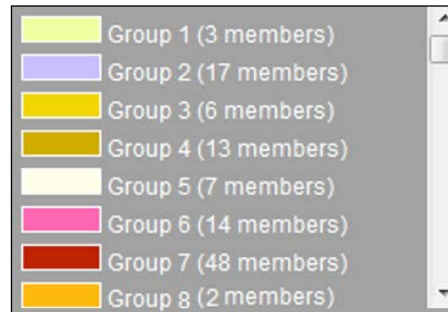
So what do we get if we just pursue the initial settings shown earlier in the chapter? Here's the result:



Miles Davis network in SigmaExporter

We have a nice graph that resembles what we just saw in Seadragon, but with additional information courtesy of the template we reviewed in the last section. The graph has an overview, a legend, search box, and group selector based on our partitioning decision (instruments in this case). In addition, if a user clicks on the information icon, the long description text area we saw earlier will be loaded, providing further context on the visualization. So you can begin to see the incremental power these options provide when compared with the Seadragon alternative.

Now let's see how we can tap into some of the Sigma functionality to begin traversing and filtering the network. If we click on the **Group Selector** drop-down menu, we'll see a host of options to choose from—in this case, they refer to specific instruments:



The Group Selector menu

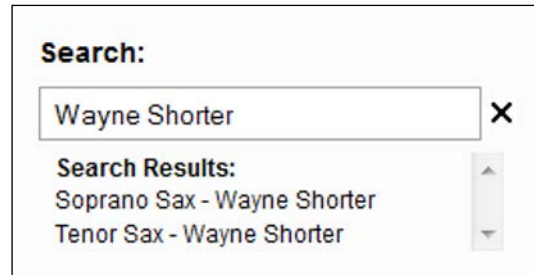
Let's select **Group 4 (13 members)** and see the results in the information pane:



Information pane showing group results

We can now see each of the French horn players employed by Miles Davis on his albums—13 of them, as this was a commonly used instrument in some of Miles' earlier ensembles. We could select any of these musicians using either the information pane or the graph to explore the albums they played on and to which other musicians they are connected. Since this is a bipartite network, each musician is connected directly to albums only, and not other musicians. Selecting any given album will reveal all of the musicians who played on that particular recording, so it's a two-step process to see linkages between musicians.

Now let's use the search tool to learn more about a specific musician. In this example, we'll enter Wayne Shorter into the search box, which results in the following:



**Search:**

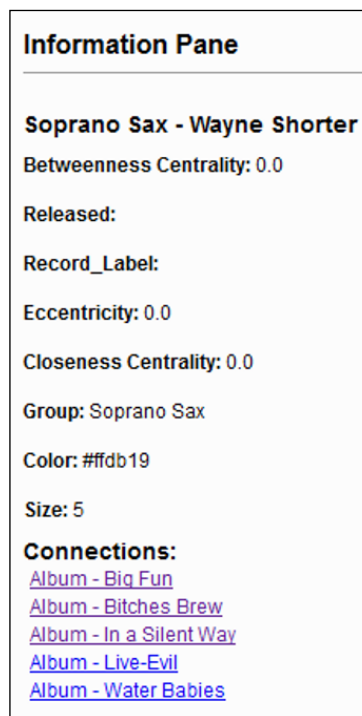
Wayne Shorter X

**Search Results:**

- Soprano Sax - Wayne Shorter
- Tenor Sax - Wayne Shorter

Sigma search option

We have two matches, as **Wayne Shorter** played both soprano and tenor saxophone on various recordings. Selecting the **Soprano Sax - Wayne Shorter** item results in the following information pane:



**Information Pane**

---

**Soprano Sax - Wayne Shorter**

Betweenness Centrality: 0.0

Released:

Record\_Label:

Eccentricity: 0.0

Closeness Centrality: 0.0

Group: Soprano Sax

Color: #ffdb19

Size: 5

**Connections:**

- [Album - Big Fun](#)
- [Album - Bitches Brew](#)
- [Album - In a Silent Way](#)
- [Album - Live-Evil](#)
- [Album - Water Babies](#)

Information pane for individual musicians



Notice that some of the fields here are either empty or meaningless for our analysis. Since Wayne Shorter is a musician, the `Released` and `Record_Label` fields are null; if we select any one of the recordings in the network, those fields will be populated. Notice also that two centrality measures return values of 0; this is because of the bipartite nature of the network. We could elect to remove these fields from the final result if we so choose, by editing the underlying `data.json` file to remove the unnecessary elements.

We will then see the more interesting information we're after, such as the group classification, the color used for the specific group, and most importantly, each of the recordings where Wayne Shorter played soprano saxophone. Each of these album titles is clickable, which will load new information into the graph display when selected.

A moment ago, I referred to how our results would differ in the event an album (rather than a musician) was selected in the graph. Suppose we selected the album titled **Walkin'** by clicking on the appropriate graph node:

**Information Pane**

**Album - Walkin'**  
**Betweenness Centrality:** 0.0  
**Released:** 1954  
**Record\_Label:** Prestige  
**Eccentricity:** 0.0  
**Closeness Centrality:** 0.0  
**Group:** Album  
**Color:** #e50000  
**Size:** 25  
**Connections:**  
[Alto Sax - David Schildkraut](#)  
[Bass - Percy Heath](#)  
[Drums - Kenny Clarke](#)  
[Piano - Horace Silver](#)  
[Tenor Sax - Lucky Thompson](#)  
[Trombone - J.J. Johnson](#)  
[Trumpet - Miles Davis](#)

Information pane for albums

Now we will see the year the recording was released as well as the label that produced the album. We will also see each of the musicians who played on the recording, each in the form of clickable links. Selecting the **Bass - Percy Heath** link will then display every recording he played on, providing further insight into the network content. This ability to navigate the network from either the graph window or the information pane makes for a very powerful, easy to use visualization.

Another powerful capability within the Sigma plugin is the ability to create advanced styling. While there are a few ways to customize the output via the basic template, there are many more options behind the scenes. To access these files, go to the directory where your visualization is parked (it will be an `index.html` file), and locate the `css` folder.

Inside the `css` folder you will find two files—a primary file titled `style.css` and a second one called `tablet.css`. Most of your tweaking will be performed in the `style.css` file, where you have dozens of elements that can be adjusted. If you aren't familiar with CSS, it's worth learning the basics, as it will allow you to truly customize your output in so many ways. Let's have a look at a small snippet of CSS code from the file:

```
#maintitle h1 {
    display: none;
}

#mainpanel {
    margin-top: 50px;
    margin-left: 25px;
    background:#fff;
    background-color:rgba(255,255,255,0.8);
    border:1px solid #ccc;
    z-index:20;
    position:fixed;
    top:0;
}

#mainpanel .b1 {
    padding: 0px 0 0 0;
}

#mainpanel .col {
    width: 240px;
    padding: 18px 18px 18px 18px;
```

```
margin: 0;

}

#title {
  font-weight: bold;
}

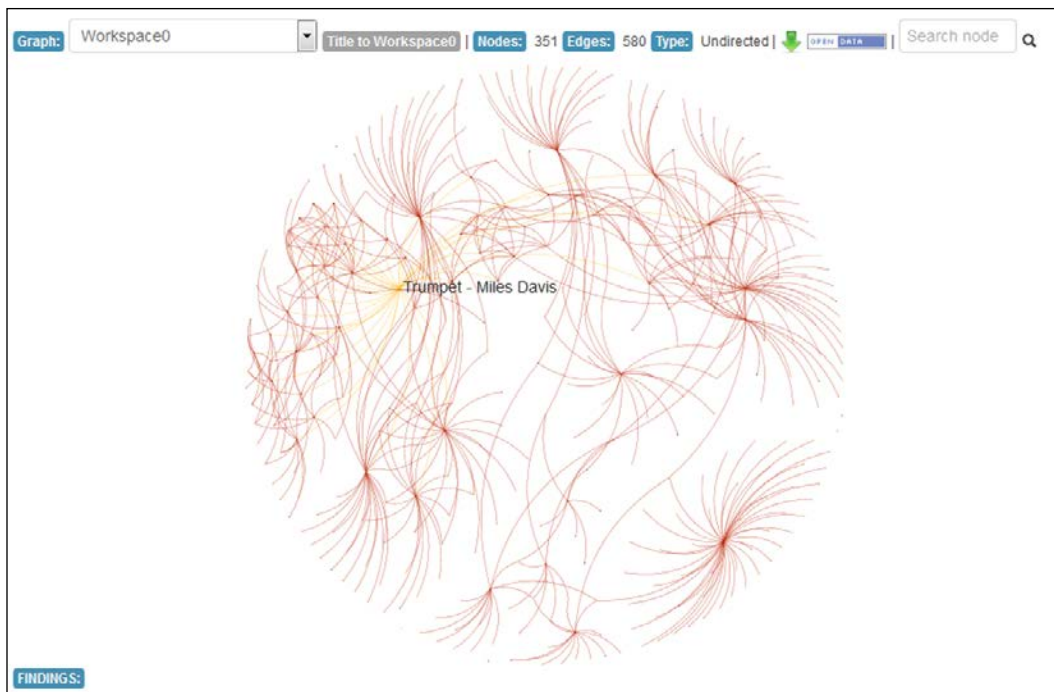
#titletext {
  padding: 6px 0 10px 0;
}
```

Just in these few selections you begin to get a sense of the possibilities, such as being able to customize the width, margin, and padding for your main information panel. You could also change the positioning of the panel, adjust its coloring, and decide whether to create a border surrounding the panel. This is just a very small glimpse into the style file — there are literally hundreds of adjustments you could make to create a unique look and feel for your published visualization. Even more edits can be made within the `config.json` and `sigma.js` files, for instance. It is important to remember that any edits you make will be overwritten if you need to recreate the network, unless you create a new folder location for each version.

## Loxa Web Site Export

We just used SigmaExporter to create multiple versions of a single network graph; now we'll follow a similar process using the Loxa plugin. If you wish to create your own graphs, make sure to install the Loxa Web Site Export tool from the available plugins list in Gephi. Our primary difference for this example will be the need to edit some of the Loxa files using a text editor, so get your favorite tool ready if you wish to do a little tweaking.

If you wish to follow along, you'll need to have the Loxa plugin installed — you should find it in your available plugins directory. The export process for the tool was detailed earlier, and is just as simple as for the other web exporters. Remember that there weren't a lot of settings to modify in the dialog screen (as there were for SigmaExporter), so we're going to proceed with the default options checked. Depending on the speed of your computer and the complexity of the network, the export could range from a few seconds to upwards of a minute. Locate the exported file and open it up in a non-Chrome local browser session. The results will be along these lines:



Default Loxa display window

Your version might have a darker background; I have adjusted the background color by editing some of the inline styling within the `index.html` file produced by Loxa. While more than 90 percent of the styles are set using a standalone CSS file, a few of the rudimentary elements are stationed within the main page, like so:

```
<style type="text/css">
  label {
    display: inline-block;
    width: 5em;
  }
/* sigma.js context : */
.sigma-parent {
  position: relative;
  border-radius: 4px;
  -moz-border-radius: 4px;
  -webkit-border-radius: 4px;
  background: #666;
  height: 530px;
}
.sigma-expand {
  position: absolute;
```

```
width: 100%;
height: 100%;
top: 0;
left: 0;
}
.buttons-container{
padding-bottom: 8px;
padding-top: 12px;
}
</style>
```

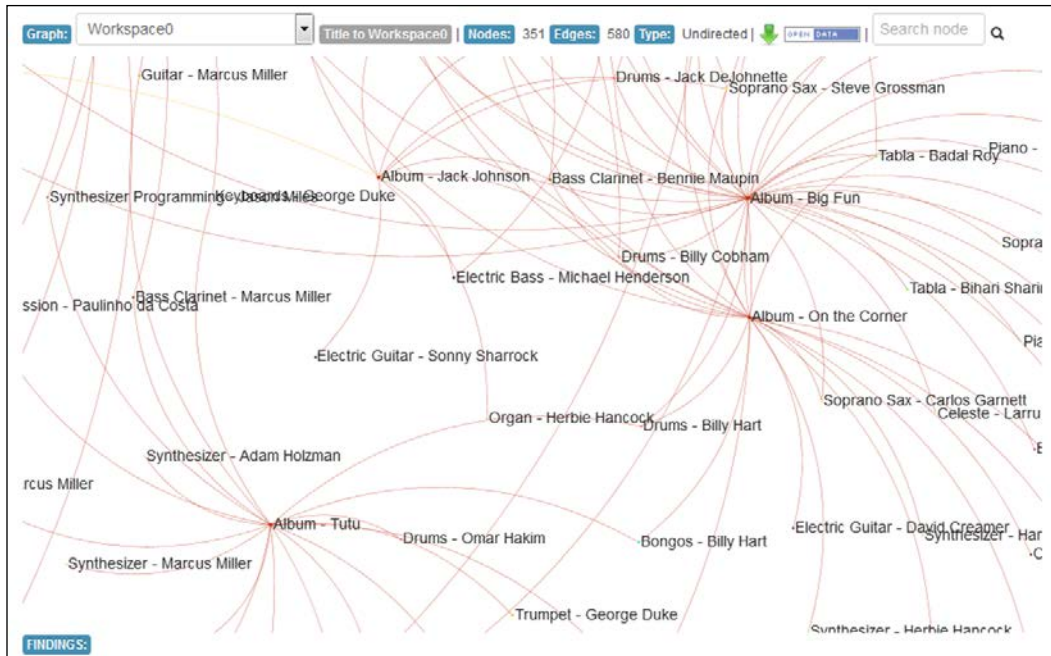
The initial sigma-parent element had a background setting of #222 (almost black) which I have changed to #fff (white) for the benefit of the print settings in the book. I have also tweaked the label styling by editing the `loxawebsite-0.9.1.js` file (your version might differ). Node and edge styling can also be manipulated by adjusting settings within this file. Use your own discretion when making changes to the styling.

Taking a look at the Loxa display window, there are a few important items to note:

- Notice that at the top of the graph there are three tabs — the **SNA** page, **About analysis** tab, and **About us** page that can be customized to provide additional information about the network. The **SNA** page refers to the actual graph, while the **About analysis** tab is intended for descriptive information about the project, much as the SigmaExporter template that provided descriptive fields. Finally, the **About us** page serves as a placeholder for information about the authors of the visualization. We'll come back to these in our next example.
- Next, there is a graph dropdown that shows all workspaces belonging to the current project. In this case, we have but a single workspace, but Loxa can accommodate multiple instances within a single project. We will discuss more on this shortly.
- Basic network information is displayed at the top of the graph, such as the numbers of nodes and edges, and whether the network is directed or undirected.
- The download image arrow will convert the network to a PDF file for offline viewing.
- Finally, there is a search box function that assumes that the search functionality is enabled



Now we are able to see each of the album titles displayed automatically, but no musicians yet as the visualization would become very crowded. Zooming in another level does just that, however, the individual musicians from each recording are now labeled along with the recordings, as shown in the following screenshot:



Loxa display with additional level of zoom

This capability, along with the ability to drag-and-drop the graph, makes for a very powerful, highly interactive environment that will provide a great experience for your users.

Before we conclude this chapter, I wanted to go back to a couple of the previously discussed tabs in the Loxa output and show you how these can be customized to make for a more polished visualization. In our initial example, you saw the plugin defaults, which serve as placeholders for additional information. We're now going to provide some of that information by editing the `about.html` and `info.html` files found in the project folder. These will be minor edits done in a matter of a few minutes. The capability to provide much more information is at your disposal.

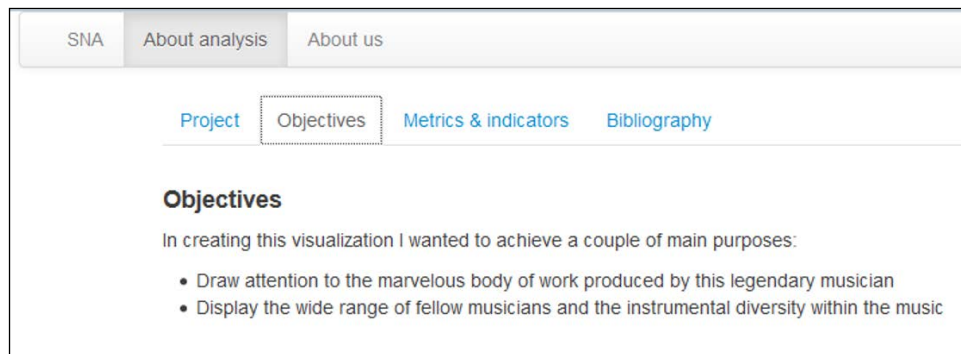


Here is a look at a few of the newly customized pages, with placeholder text updated to match the project:



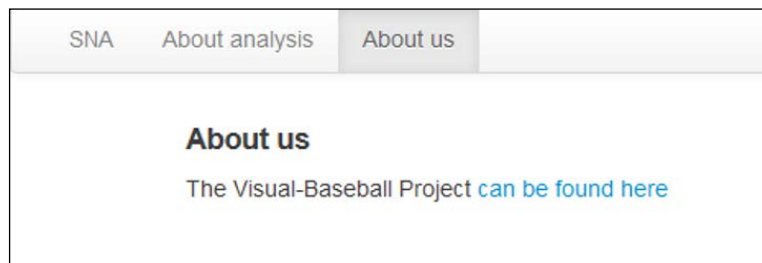
The customized About the Project page

The **Objectives** tab can also be customized with the goal of stating why the visualization exists, as well as any secondary aims behind the network graph:



The customized Objectives page

More customization can be done on the **About us** tab. While I just placed a website link here, much more could be done to tell the world who you are:



The customized About us page

The sky is the limit for customizing your Loxa project—all you need is some very basic HTML and CSS skills to create a unique, informative and interactive project.

## Summary

In this chapter you learned how to take advantage of the many available tools that will take your network graphs beyond the Gephi application and into the hands of external users. We learned about three main categories of exporters in Gephi.

We discussed graph file exporters, which export your network data into formats that are readily accessible by other network graph software, as well as by Gephi. We went on to discussing image exporters, which facilitate turning your graph into static files via PNG, SVG, or PDF exports. We also elaborated on web exporters, which turn your Gephi network graphs into interactive displays for the Web.

We also covered how SVG and PDF exports could be further edited using a tool such as Adobe Illustrator or Inkscape and shared some examples using Inkscape.

Our final sections covered the process of creating and exporting network graphs for the Web, including sections on editing the underlying templates and source files to achieve a finished, professional look for our graphs.

In *Chapter 10, Putting It All Together*, we'll take a look at using the best practices from this book to begin building your own networks, and we'll also have a brief discussion about the future of network analysis.

# 10

## Putting It All Together

Until this point in the book, we have focused on many aspects of Gephi that can be used to create effective and interactive network visualizations. In many cases, these topics were discussed in isolation within their respective chapters. We have managed to merge multiple functionalities on some occasions, but never the entirety of what Gephi can help us create. In this final chapter, the goal is to pull all of these elements together at a more holistic level and build some Gephi projects from start to finish.

We'll focus on three primary goals in this final chapter as we seek to further develop your own network analysis skills. The first two will make an effort to fully leverage Gephi's considerable capabilities, while the third section will focus on what the future of network graph analysis might encompass. Here's what we'll cover in this chapter:

- In the first section, we'll look at how we can use Gephi to interpret and enhance existing networks
- We'll spend the majority of the chapter creating a couple Gephi projects from start to finish, eventually deploying each to the Web
- Our final section will look at where network analysis stands today and where it is likely to go in the future

Let's begin by using Gephi to interpret an existing network visualization.

## Using Gephi to understand existing networks

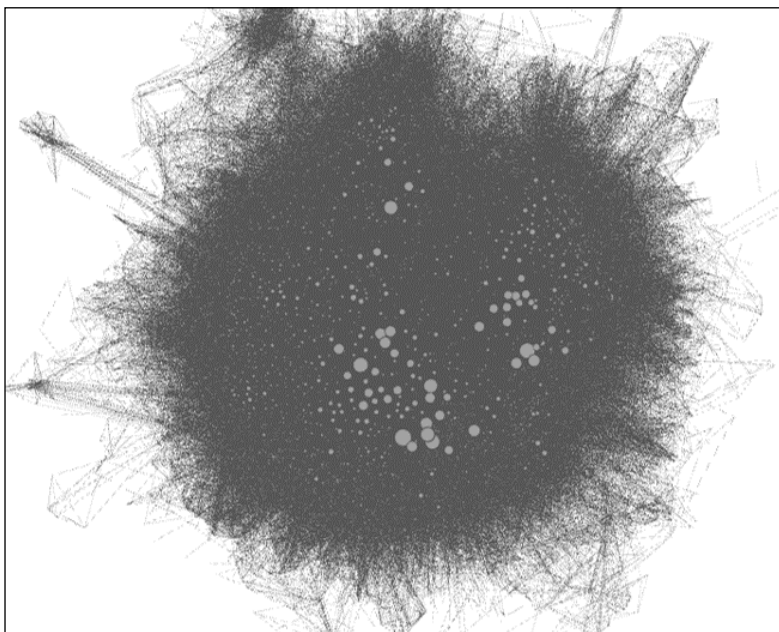
In previous chapters, we've examined a variety of techniques and functionalities present within Gephi that act on the existing network data. These included topics such as filtering, partitioning, running statistics, styling, and ultimately exporting our projects. As I mentioned in the introduction, much of this has necessarily been done in isolation, with periodic instances where multiple approaches were merged. In this section, we are going to utilize a considerable portion of the Gephi application to understand and potentially improve on some of the existing network graphs.

The hope is that performing many of these operations on existing graphs will prepare us to create better projects when we begin with our own raw data. Think of this as a bit of a warm-up exercise before we tackle the more challenging goal of building our own projects.

So let's begin by exploring a single challenging graph, and begin flexing our creative muscles using a variety of Gephi techniques. It's time to locate an existing project to modify—either `.gexf` or `.net` formats should work, or even existing the `.gephi` files. The Gephi wiki is a good place to start, although you could search for any of these file types on the Web and find some additional resources. Let's get started.

The example we'll walk through, the **Marvel Social network**, is available in the `.gephi` format on the Gephi wiki at [https://wiki.gephi.org/index.php/Datasets#Social\\_networks](https://wiki.gephi.org/index.php/Datasets#Social_networks). The network represents thousands of Marvel comic book characters and how they have interacted with one another throughout decades of comic book editions. This is a rather dense network of roughly 10,000 nodes and nearly 180,000 edges, which gives us a graph that is nearly impenetrable at first glance. At a minimum, we would like to take this network and give the user some tools to help them navigate the graph effectively. Beyond that, the ability to make the graph more visually appealing would also be a good idea, as long as it is not simply for aesthetic purposes alone. In other words, let's make it look nicer within the context of increased efficiency for the user.

Here's our starting point of the network graph:



The Marvel hero network graph

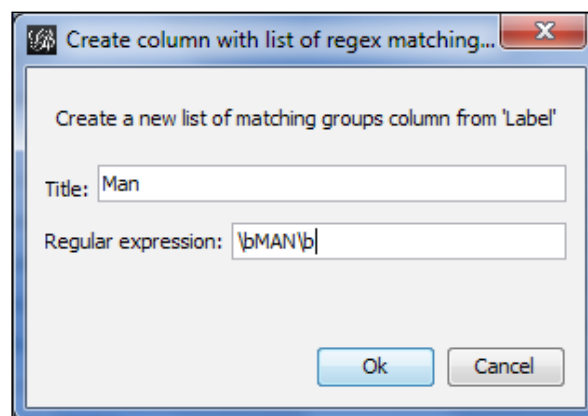
A bit overwhelming at first, isn't it? We do have a few visual cues to begin working with – there are clearly some high degree nodes sized accordingly. There also appear to be some outlying clusters at the extreme edges of the graph, and we do have the ability to understand the structure a bit by hovering over selected nodes. Nonetheless, the graph suffers from the infamous hairball effect that we generally seek to avoid in network visualization. So where to begin?

Some of the more obvious opportunities to improve the graph are not available in the data. For instance, there are no categorical variables informing us when or where a character originated. Nor are there any fields indicating in which of the many Marvel comic books the characters appeared. Our best (and only?) option would seem to be to improve the graph by creating our own variables and filters using either statistical measures or created variables. We can do this – it isn't as hopeless as we might have originally feared. Gephi provides some powerful tools to enhance even the most challenging networks.

I'm going to start with a tool that we haven't previously explored in the book, but one that's especially useful when confronted by a dataset with few variables to work from. At the bottom of your **Data Laboratory** window are a series of icons, including the **Merge columns** function we previously used and will encounter again later in this chapter. However, for this example, we'll slide to the right of the window and employ one of the useful regex commands to create some new variables for our graph. You might recall the **regex** (short for **Regular Expression**) functionality from *Chapter 5, Working with Filters*, where we used it to create wildcard filters. This time we're going to use it to create some new field values to make network navigation far easier for the end user.

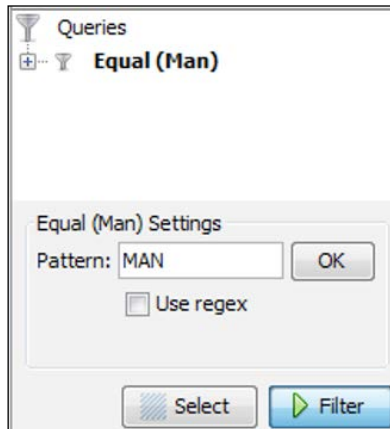
We'll use the second of the two regex functions (Create a boolean column from regex match, Create column with a list of regex matching groups) – the one with the lengthy title of **Create column with list of regex matching groups**. Click on the **Create column** button to open the dialog box, where we'll create a new field titled **Man**. Why Man, you ask? Because our quick perusal of the dataset and slight familiarity with Marvel characters (and comics in general) suggests that there will be many characters with man in their title. For starters, we have Iron Man, Spiderman, and likely dozens of others. We'll have to be careful with the filter so that we don't identify all the characters with Woman in their title as well, since man is also represented within those five letters. For that matter, we don't want portions of a character named Norman (for example). So this can get a little complex, but we'll get there using regex.

For our example, we can use the `\b` modifier to find only those instances where the term MAN is found in the dataset (this data is in uppercase). This allows us to create a new column using the dialog screen:



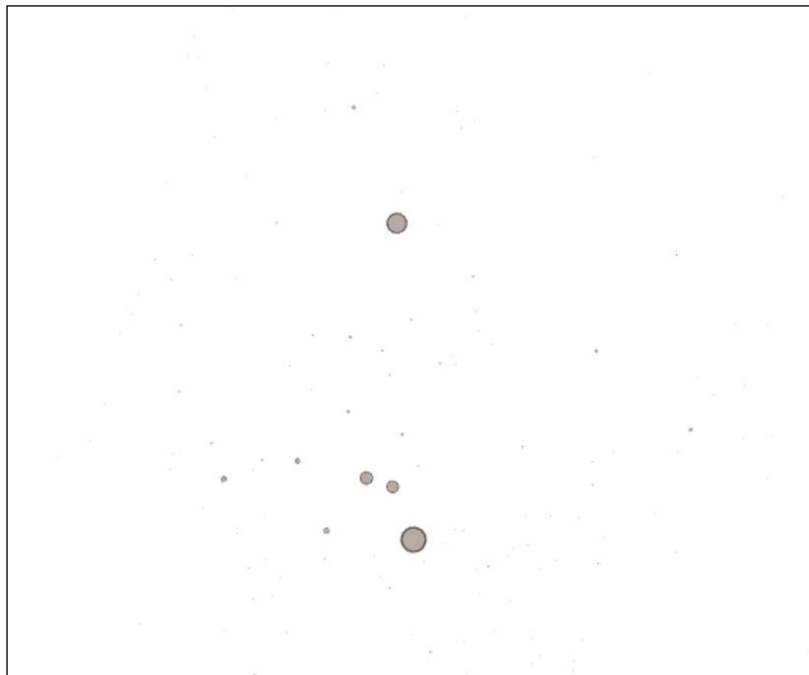
Using regex to create a new column

Now that we've created the column, let's use it to filter the graph by working with the **Equal** filter and dragging the **Man** field down to the **Queries** window. We then need to enter the matching value for the filter to work properly, as follows:



Setting the query pattern with equal filter

Clicking on the **Select** and **Filter** buttons results in a network graph that displays only the records that match our criteria, as shown in the following diagram:



The graph results from applying the MAN filter

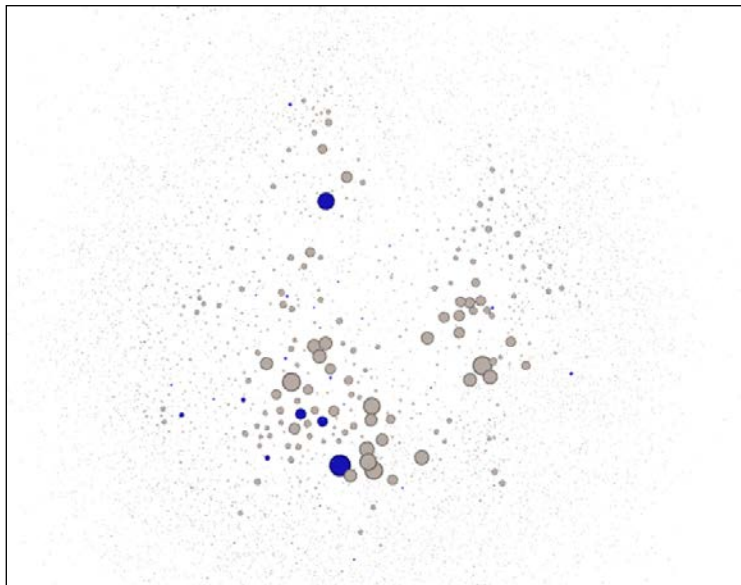


We can verify the efficacy of the filter by perusing our results in the data laboratory:

Label	Man
IRON MAN/TONY STARK	MAN
IRON MAN IV/JAMES R.	MAN
MAN-THING/THEODORE T	MAN
SEAWEED MAN	MAN
SPIDER-MAN/PETER PAR	MAN
WONDER MAN/SIMON WIL	MAN
, "WONDER MAN/SIMON WIL	MAN
, "IRON MAN/TONY STARK	MAN
, "IRON MAN IV/JAMES R.	MAN
COBALT MAN	MAN
, "COBALT MAN	MAN

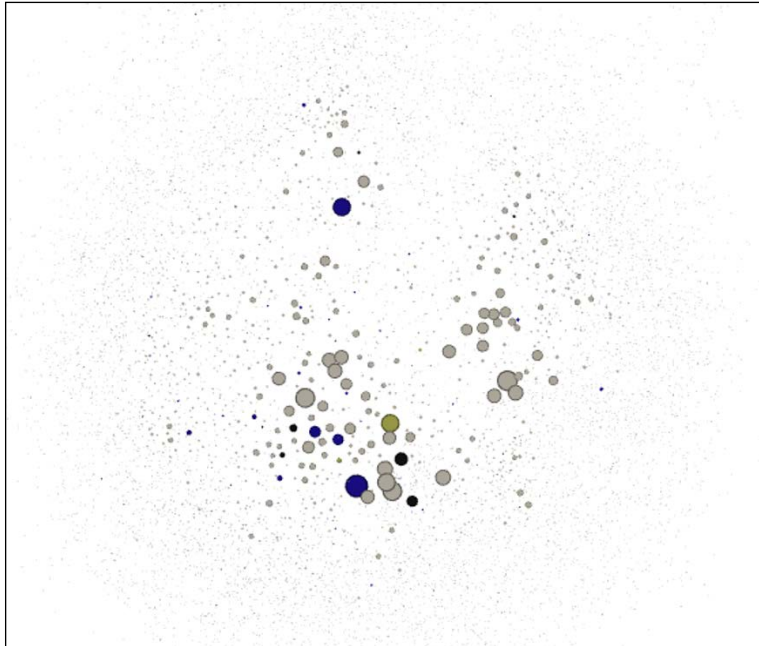
The partial results from application of MAN filter

So we've accomplished something that's potentially very useful as we attempt to create a more easily navigated network. There's clearly more that can be done, including coloring the graph nodes based on our filter. All we have to do here is right-click on the **Reset colors** icon and select a color that will be applied only to our filtered members (make sure the filter is applied – otherwise the entire graph will be re-colored). We'll choose a dark blue shade to identify the *Man* characters, and then remove the filter to see the following result:



The updated graph with new color for all MAN characters

We can then repeat this series of steps to create other new fields. For example, I created a field titled `woman` and another titled `black` in recognition of the many cartoon characters (often villains) with black as part of their name. By the way, you might need to hit the **Reset** button to see your new values in the **Filter** window. Performing the filtering and coloring on each in sequence results in the following graph where we now have three distinct colors beyond the gray for the remaining members of the network:

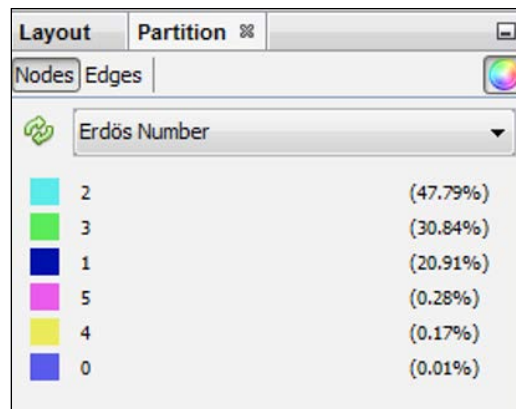


The updated network with distinct colors for Man, Woman, and Black columns

We can also use graph statistics to filter the graph more effectively. Bear in mind that this might take a while given the size and complexity of this graph (it might even fail, depending on your available memory). Memory settings can be adjusted by editing the `gephi.conf` file found in the `Gephi-0.8.2\etc` folder, or refer to <http://gephi.github.io/users/install/>; however, you might be able to calculate some centrality measures, or at the very least we can work with degrees to segment the graph. We could even elect to use an **Erdos Number** (for more details on the Erdos Number Project, refer to <http://www.oakland.edu/enp/>) to see how close each graph member is to a specific character (say Iron Man). In fact, let's try the Erdos Number approach to partition the graph.

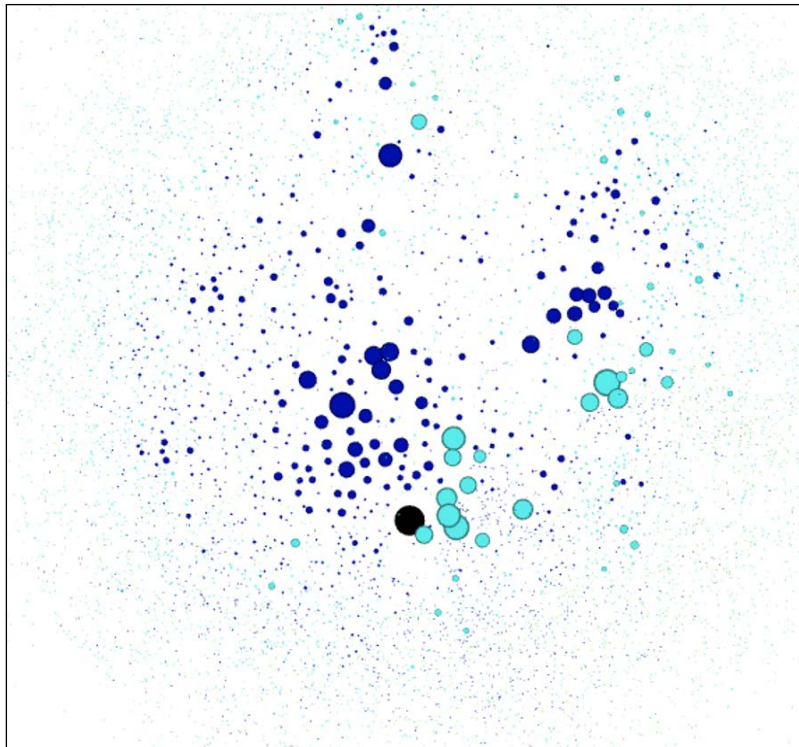
After calculating the Erdos Number for the Tony Stark Iron Man character (as opposed to the various other Iron Man incarnations) we can begin to segment the graph based on how each character relates to Iron Man. The average number turns out to be 2.103, indicating that a typical character is slightly more than two edges away — direct connections would have an Erdos Number of one. We would like to see the distribution of these visually to better understand the role of Iron Man in the Marvel pantheon, and who is most closely identified with him.

To do this, we will simply select **Erdos Number** as our partition criteria, which will show us the numbers ranging from 0 (for the central character) through 6, as follows:



Setting an Erdos Number partition

We can then apply the colors (modify these first if you wish to create a personalized color scheme) to see the final result:



Marvel hero graph partitioned by Iron Man Erdos Number

Looks a lot better than where we started, doesn't it? There are obviously many other ways we could have modified the graph—different field creation, alternative partitioning strategies, new color schemes, or filtering approaches. The point is that Gephi provides a powerful array of tools to help you move in almost any direction when working with an existing network graph.

We'll now move on to create our own projects from start to finish.

## Creating new Gephi projects

Now that we've dissected and improved upon some existing network graphs, the time has come to develop our own projects from scratch. This will involve a number of steps, and will allow us to fold in much of what has been covered thus far in this book. Here's a summary level synopsis for everything we'll do to create our projects—it's a long list, but you should be well prepared to handle each of these steps based on what we've covered to this point.

Here are the steps, arranged in a typical order, although some of these steps could be swapped (such as the steps involving statistics and filtering):

1. First, we'll locate a dataset that is suitable for network analysis. The data might or might not be in a suitable format when we find it, which will dictate whether the next step is necessary.
2. We might need to format the data so that Gephi can read it without issues. Remember that Gephi is able to handle a variety of input sources created in various graph file formats. If we are sourcing data that isn't already in one of these formats, then we'll have to create either a pair of `.csv` files (one each for nodes and edges) or have the data available via some MySQL (or other database) tables.
3. When the data has been imported by Gephi, the first thing we see is a random layout. After a cursory inspection of the data in the data laboratory, we'll want to apply one of the many layout options made available in Gephi. This is typically an iterative process, as your initial selection might not deliver the results you are seeking. Stay with this process until you get something that makes you comfortable.
4. After finding a suitable layout, it's time to do a visual inspection of the graph. Are there obvious patterns such as **homophily**, or does the network take on a more random appearance? Do we have a single **giant component**, or are there multiple disconnected subnetworks? Do we see obvious **hubs**, or are there many alternative ways to traverse the graph? Does the graph have a large or small **diameter**? These are just a few of the questions we should ask ourselves as we inspect the graph.
5. Now we can further understand the network by employing any of the many filters provided within Gephi. These can be especially helpful when faced with a dense network, but are certainly not limited to just that condition. We can also gain insight by filtering the data based on degree levels, by classifications or partitions, or based on a combination of criteria using an intersect query.

6. It's time to apply some statistical measures to the graph to help confirm our initial impressions. The centrality measures are especially important, as they will apply real numbers that can help identify the network influencers, regardless of their placement within the network. Other statistics will help confirm initial impressions about diameter, clustering, and homophily, among other measures.
7. From here, we can begin segmenting and partitioning the network using a variety of tools in Gephi. This step will often highlight graph patterns through visual means such as sizing and coloring, making it easier for the graph viewer to understand the message.
8. In cases where the network has some sort of time element, we can create dynamic graphs that call out temporal changes in the network. This might come in the form of a node entry and exit as time passes, or it could reflect time-driven changes in status for individual nodes. In either case, dynamic networks can convey a very powerful story to viewers.
9. Our final step in most cases will be to make the graph available for external users, often through deployment to the Web. This then makes the network interactive for all users without requiring any knowledge of Gephi. In other cases, we might elect to simply share an image of the network via a `.png` file, or we could choose to tell a story using the `.svg` or `.pdf` output formats.

Sounds like a lot, but as you become more comfortable with Gephi, much of this will become second nature. We're going to put this into practice by creating a pair of projects, the first a dynamic network that remains within Gephi, and the second a project that we will push to the Web for user interaction.

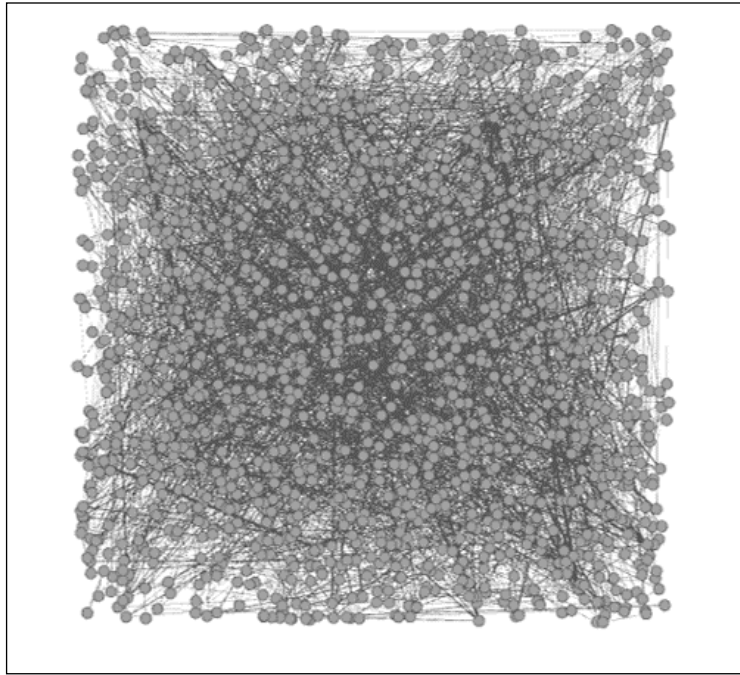
## Project 1 – Newman NetScience dataset

Our first example will use the NetScience dataset created by noted network scientist Mark Newman. Newman's data examines the working relationships between hundreds of academic network science practitioners through a co-authorship network. Nodes represent authors, and edges the connections between co-authors. This data can be found in multiple places on the Web, including Newman's own site. We'll begin with a `.gml` version of the data, which you can find at <https://app.box.com/s/177yit0fdovz1czgcecp>.

All that exists at the start of the project are the respective nodes and edges tables, which will give us full control over the ensuing steps. From this raw data, we will create a project that incorporates a wide range of Gephi techniques and methods, resulting in a finished network graph that tells a compelling story. We'll follow the steps outlined a moment ago, although we might make a slight detour here and there.

## Exploring the network in Gephi

Once the data has been loaded in Gephi, we'll see the following network in a random layout, as provided in detail in our chapter on selecting a layout algorithm. This won't look like much, but it does provide us with enough to get underway:

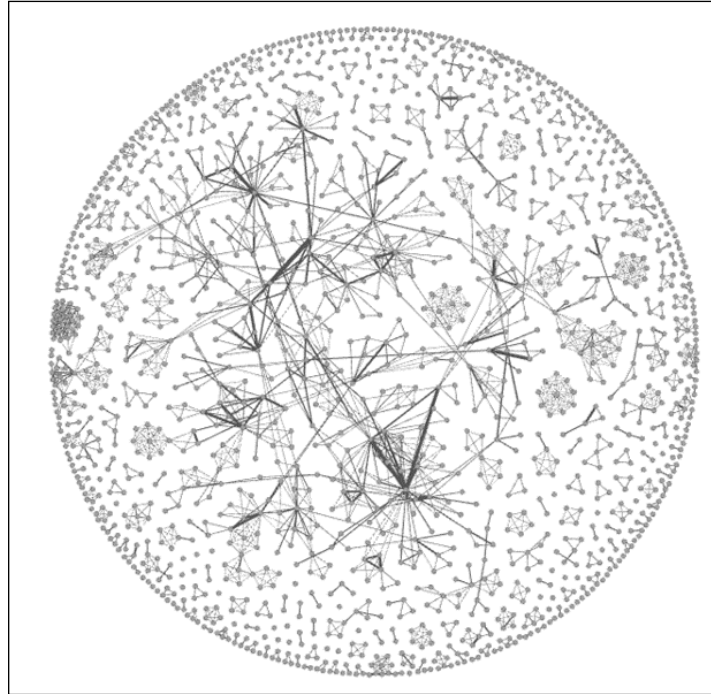


The Newman network science collaboration network

We need to get the data into some sort of layout that will help us to understand the network structure. The context window tells us that the network has 1,589 nodes and 2,742 edges; fewer than two edges per node on average. Thus, our network is not very dense, which might help point us to a specific sort of layout. We also know that the network has enough nodes to perhaps eliminate some other algorithms from consideration—a circular layout might not work effectively for displaying this network.

Given this information, I am going to begin with the **ARF** algorithm, which I find to be useful for small- to medium-sized networks of this sort. We'll see whether ARF effectively displays the network; if not, then another option will be considered. For instance, if our network turns out to be highly clustered, the ARF might not distinguish the clusters as well as something like Force Atlas (remember that ARF creates largely circular layouts). We'll need to make that determination after seeing the results.

After running the ARF algorithm for nearly 10 minutes (your time might vary) using the default settings, we will be able to see the following output:



Newman network after applying the ARF layout

Based on these results, I believe we can move forward. The graph quite clearly displays a number of clusters, an indication of network scientists collaborating on projects. We also see some instances where nodes are linked to more distant members of the graph, an indication of some cross-cluster collaborations.

Something else is very clear — this is not a single connected network. Instead, there are multiple cases where smaller subnetworks exist. This is going to influence some of our statistical measures, as we'll see momentarily. For instance, there is no way to calculate a single diameter measure, as it is impossible to traverse the entire graph.

Next, let's begin using some filters to better understand the network. Here are a few questions we can attempt to answer:

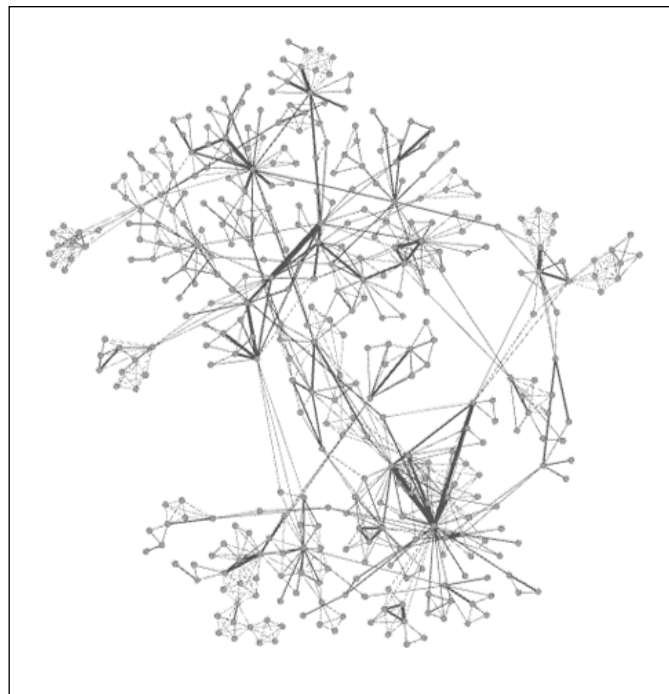
- Which nodes are the most influential, as measured by degrees?
- Where are the heaviest edges an indication of frequent collaborations?
- How large is our largest connected component, and who belongs to it?



We will quickly discover an issue — there is no explicit degree measurement in our nodes table, as we had in some prior datasets. Fortunately, we have several easy ways to measure degrees. We could take the data outside of Gephi and calculate a degree value for each node, we can use the Gephi ranking function to size all nodes based on their individual degrees, or we could simply use the filters within the **Topology** folder to look at **Degree Range**. So even though we don't have an explicit field for degrees, Gephi recognizes the network structure and lets us filter using the degree attributes. We can note that the degree range runs between 0 and 34, so let's examine all the values of 15 or greater. The graph now shows just 34 nodes, roughly 2 percent of the network, clearly concentrated in three distinct areas of the network.

Now let's look at edge weights to see where the most frequent collaborations occur. In this case, the source file does provide the edge weights, making it very easy for us to filter on. We have multiple ways we can go about this, but using a range would be a sensible approach. Let's set the range to 2.0 or greater, which leaves us just 14 edges from our initial total of close to 3,000. We can easily note who the collaborators are by navigating to the edges table in **Data Laboratory**.

Finally, let's apply the **Giant Component** filter to the network to understand what proportion of the network is connected in the largest area of the graph and we will get the following result:



Giant component of Newman network

As you can see, there is a large connected component, but it represents just 24 percent of the entire network, suggesting a very fragmented graph with multiple pockets of isolated activity. So, fully three of every four researchers are not a part of the largest network. This could change significantly if just a few of the external nodes were to collaborate with something in the giant component, and would make for an interesting temporal study to check whether the network evolves significantly over time.

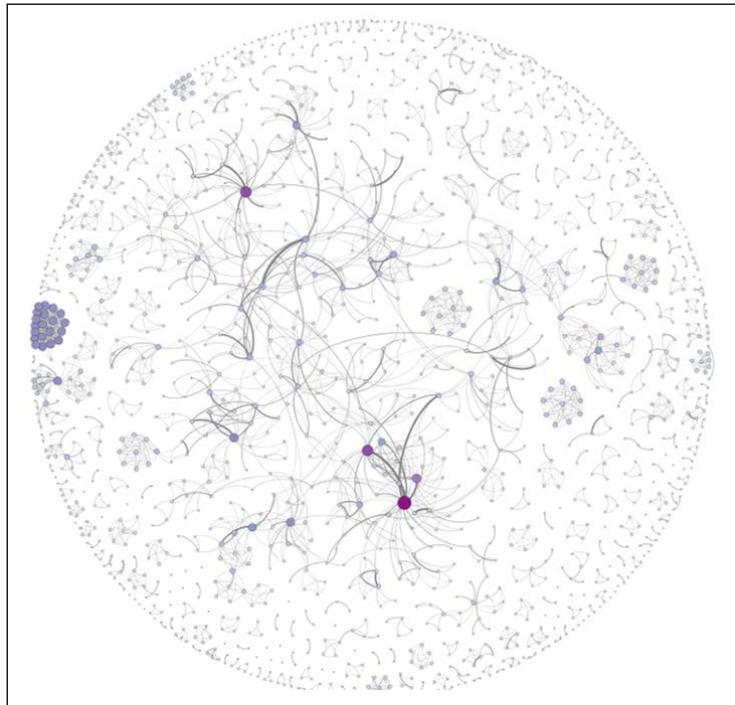
Let's apply some statistical measures to the graph to understand patterns even better. Remember our earlier mention about the difficulty of calculating diameter across the network, due to the many component groups. However, we still have the ability to run many statistical functions, but must recognize their limited meaning in certain contexts (such as a component with just five members). So our primary goal should be to examine the giant component and its member nodes, as this is where the most significant interactions are taking place.

After running a battery of statistical measures, we have the official validation for what we already suspected. Here are a few examples of the official validations:

- Graph density is 0.002, an exceptionally low figure, which confirms what we knew about the low number of connections relative to the node count
- The average clustering coefficient is 0.878, which confirms that the graph is highly clustered, something that is visually apparent
- There are 396 components in the network, which gives further evidence of the fragmented nature of the network
- The average degree is 3.45, which means that a typical member of the network has between three and four collaborations

Everything we note confirms our expectations, with no hidden surprises. Now that we've done our due diligence on the filtering and statistical fronts, it's time to make the graph more accessible and informative for users. We'll do this through the use of partitioning and segmentation, which will bring to our attention the most critical elements and patterns within the network.

For starters, we'll rank the nodes based on their degree level, which will quickly highlight some of the more evident patterns in the network. While we're at it, let's apply color using the same criteria. Set the size range between 2 and 25 and use one of the built-in color themes, and then open the **Preview** window for improved aesthetics. Let's increase the edge thickness to 2.0 and we see the following result:



Newman network with node size and color customization

These edits call out the high degree nodes in the graph, and also help to highlight unusual patterns like the tight cluster at the left of the graph. So performing the simple ranking exercise has helped make the graph more understandable at first glance. We could have employed some other approaches to segment the graph, such as using one of the available clustering algorithms, or perhaps through the use of a partition. The latter is a bit problematic with this dataset, as we don't have any information about institutional affiliations, professional credentials, or some other unifying characteristic.

## Deploying the project to the Web

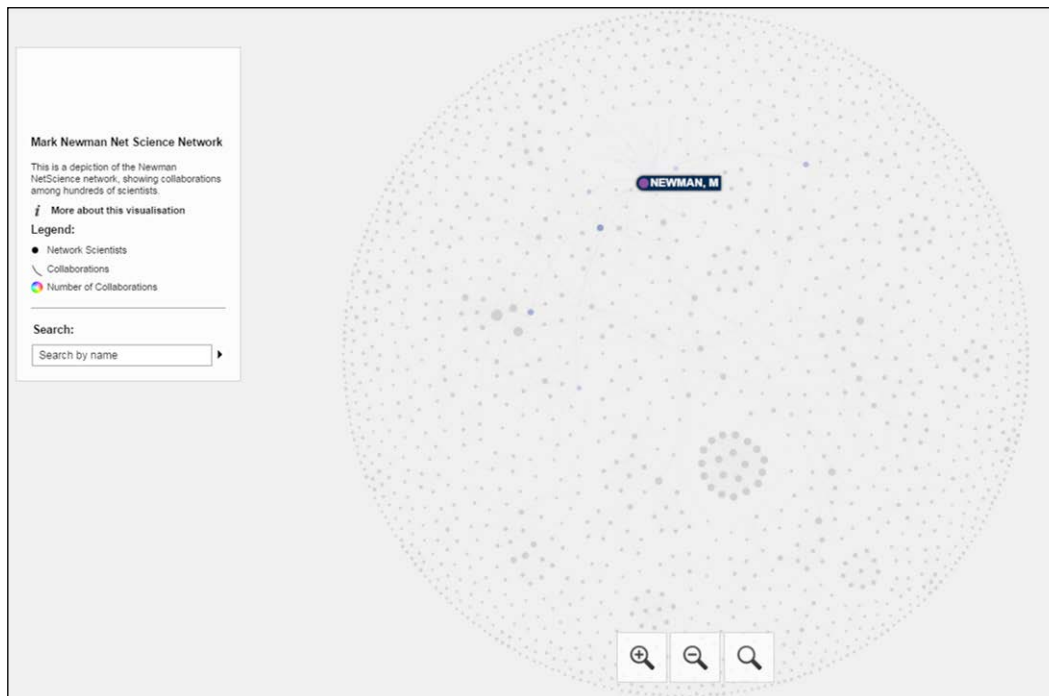
Our next step at this point is to take the graph beyond Gephi, perhaps as an image or a PDF file, or even an interactive web project. Let's make this one interactive, as it will give end users an opportunity to explore the graph and find answers to many potential questions. For this example, we'll proceed using the Sigma.js Exporter to create a straightforward interactive network for the Web.

If you are unfamiliar with the SigmaExporter process, refer to *Chapter 9, Taking Your Graph Beyond Gephi*, where we employed it to build our Miles Davis example for the Web. As you can recall, we used the template process to fashion an informative network about the 48 studio albums created by the pioneering musician. For our new instance, we'll need to edit the text to tell a relevant story about the network science collaboration network assembled by Newman.

I've gone ahead and edited the template settings to reflect the data in this particular network and published the graph to the Web. You can find it at:

<http://visual-baseball.com/gephi/NetScience/network/>

Here's a screenshot that shows the familiar layout using ARF and Sigma.js, hovering over the Mark Newman node:



Newman network on the Web using Sigma.js

If we select the same node, we'll see all of Newman's collaborators and then have the ability to view their respective connections through a simple click. This is the same approach we shared with the Miles Davis network in *Chapter 9, Taking Your Graph Beyond Gephi*. Here's what we get after selecting the Newman node:



List of collaborators for M. Newman

To recap, we started with a raw graph file and wound up with an engaging, interactive graph out on the Web. Now users have the ability to easily navigate through this network, to learn more about the various collaborations as they go. Had the dataset provided the dates and titles of each of the collaborations, we could have turned this into an incredibly rich experience, but we'll have to be satisfied for the moment with this still powerful visualization.

The output files can be found at <https://app.box.com/s/177yit0fdovz1czgcecp>.

This gives you the ability to begin playing with the CSS styling and other settings that allow you to personalize the network.

## Project 2 – high school network with dynamic edges

For our second project, we're going to have some fun working with a dataset that shows interactions between students at a high school in France over the course of seven school days. The original dataset can be found and downloaded from <http://www.sociopatterns.org/datasets/high-school-dynamic-contact-networks/>.

The data provides a history of active contacts between individual students within a single high school, measured in 20 second intervals. As you can imagine, mapping the edges in this fashion could lead to a lot of connections appearing, disappearing, and then reappearing, making for some light entertainment without adding a lot of insight into behavioral patterns. Feel free to create your own time intervals in this fashion if you want to explore this; our finished project will take a slightly different approach.

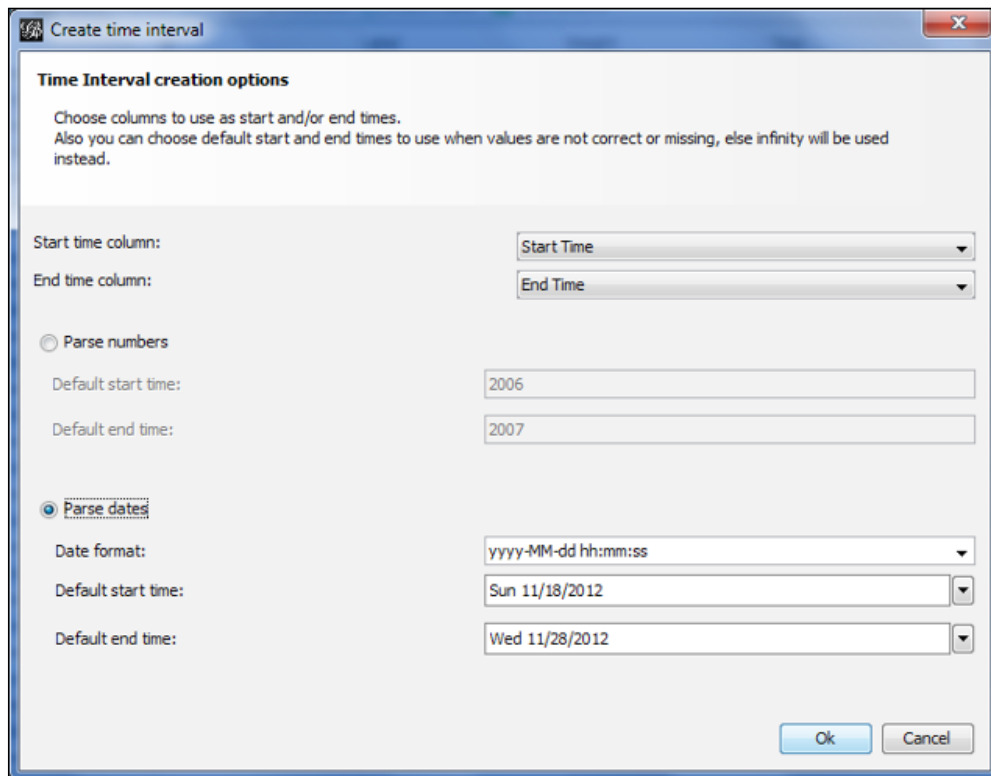
Rather than mapping the slightly spurious 20 second connections, I wanted to view the patterns that build over the duration of the study. To do this, we will add new connections as time elapses while still retaining previously existing ones. This will give us a better idea of how patterns evolve over the course of a day or a week, while simultaneously drawing attention to frequent connections that might highlight a variety of relationships in the network.

We're going to step through the project using a number of Gephi methods to set up our working project. Once everything is in place, we're going to capture images of the network at the end of each school day to note what changes have taken place. We'll also capture some fundamental graph statistics at a macro level (feel free to explore individual students on your own) to confirm our visual impressions. Finally, we'll pull everything together in an Adobe PDF file that tells a useful story for viewers.

If you wish to start from scratch, the source node and edge files can be found at `thiers_nodes.csv` and `thiers_edges.csv` at <https://app.box.com/s/177yit0fdovz1czgcecp>.

## Using Gephi to explore the network

Let's begin by opening the node and then the edge file using the Gephi data laboratory import spreadsheet functionality. Once the files have been imported, we need to create a time interval so Gephi can display a dynamic graph. Remember that we covered this process in *Chapter 8, Dynamic Networks*. In **Data Laboratory**, select the **Merge columns** icon, and then select the **Start Time** and **End Time** columns, followed by the **Create time interval** option from the drop-down menu. Click on **OK** to load this dialog screen:



Setting time intervals by merging columns

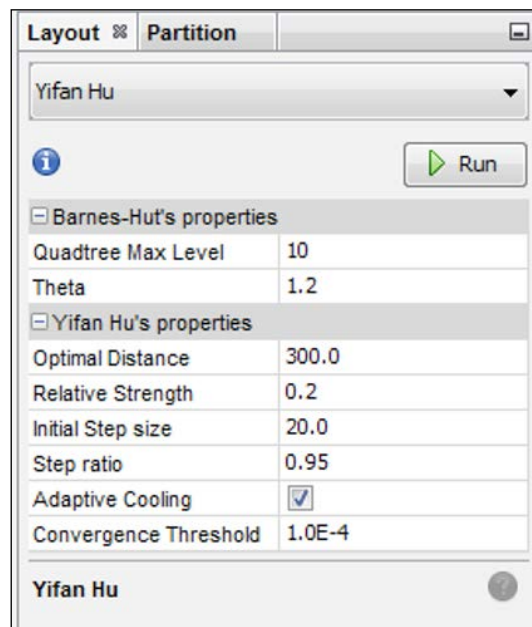
Select the **Parse dates** option, and make sure that the format is entered in the same form as in the image. After clicking on the **OK** button, you should see an **Enable Timeline** icon at the bottom of the Gephi window. Selecting this will load a timeline that looks like this:



Timeline after creating time intervals

As you can see, Gephi has identified the day of the month element in the data, giving us the range of days from the 19th through the 27th of the month. You'll also see the typical random layout in the **Preview** window, waiting anxiously for you to provide a proper layout option. To make the project a bit more fun, hide the edges using the **Show edges** icon (it toggles the edges on and off) in the preview window. This way, we won't spoil the fun for the dynamic edges we'll see in a moment.

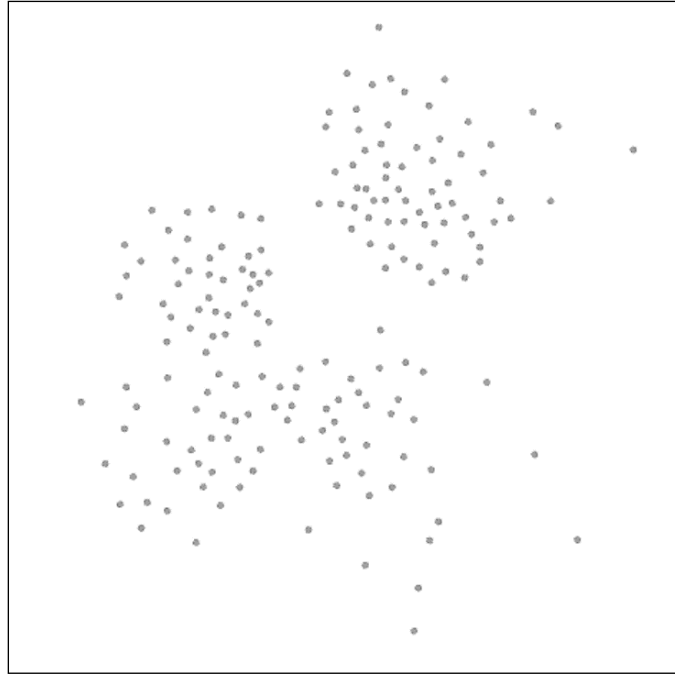
Now move to the **Layout** tab and select the **Yifan Hu** option if you wish to follow this example to the letter. Feel free to play with other algorithms if you want to create something a little different from this example – the underlying statistics and network structure will remain the same regardless of your selection. I tinkered slightly with the default settings to spread the network out just a bit:



Yifan Hu settings for a dynamic high school network



Here's what the Yifan Hu yields (edges still hidden) using the settings in the prior screenshot:

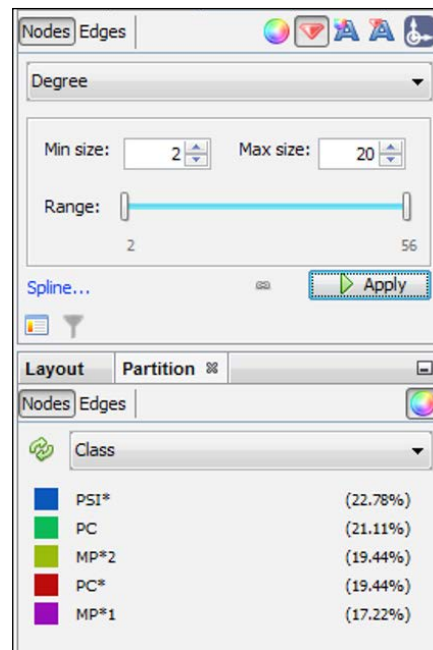


Network graph after applying the Yifan Hu layout

Our next step will be to size the nodes by degree and then to color them according to the **Class** element in the data. These steps will serve two purposes:

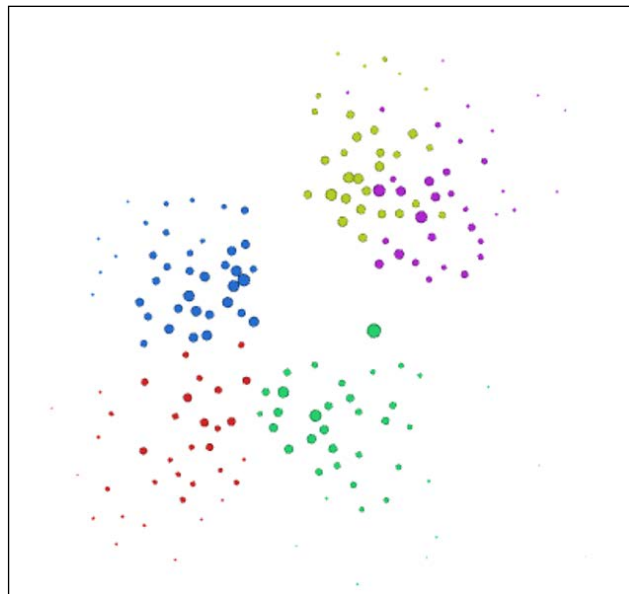
- Sizing will give us an indication for which nodes have the highest contact levels in the network.
- Partitioning by class will provide the formal structure of the network as defined by the school authorities. In this case, we have five classes, so the number of colors in the graph will be quite manageable.

For this example, the nodes have been given a size range of 2 to 20, as shown in the following screenshot, (actual values range from 2 to 56 degrees), which should allow us to see the higher influence members without distorting the graph or obscuring the smaller nodes:



Sizing and partitioning nodes in the high school network

After running each of these steps, we now have a graph that effectively illustrates the formal class structures while also showing the range of influence across all nodes:



High school network colored by class partitions

It's now time to show the edges between nodes — toggle the **Show edges** icon to make the connections visible again. Once that's complete, move to the timeline and drag the bar to the very start of the timeline. Shrink it as far as possible, by grabbing the right edge of the bar with your mouse. This puts us at the start of the first day (the 19th). Notice that there are just a few connections showing in the overview window, that reflect the handful of students already connected early in the school day. Now let the timeline run by clicking on the arrow to the left. Watch how connections build as we move through the school week.

If you find the graph moving a little quicker than you prefer, adjust the timeline settings (covered in *Chapter 8, Dynamic Networks*) to the left of the arrow. Watching the network build over the course of the study period reveals patterns of how members connect within their own class as well as with nodes in other classes. In some cases, there is little interaction between members of two classes, perhaps due to the physical structure of the school building, or perhaps related to the curriculum constraints. In any case, we have a potentially interesting story we can turn into a sort of time-lapse static presentation.

## Creating the project as a PDF

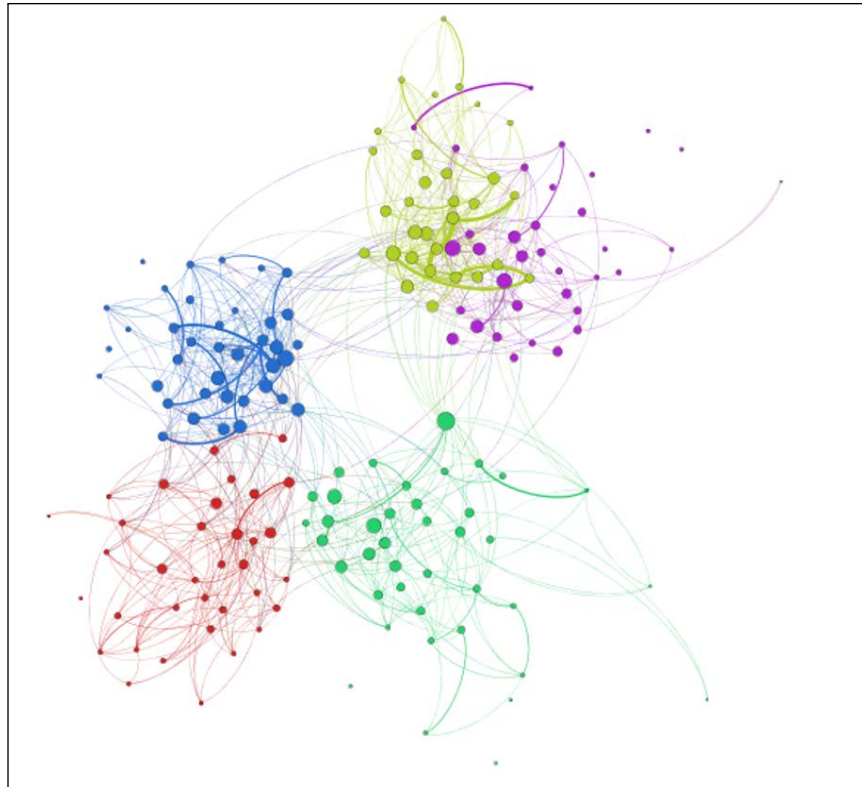
In order to tell a compelling story about a dynamic network in a static format, we're going to have to provide a little more background information than in our interactive web example earlier in the chapter. We cannot afford the luxury of networks where users can zoom and pan for more information. So our approach must be slightly different, in that we have to provide users with enough information to tell the story. Everything is in our hands in this case — the user cannot craft their own story via interaction.

So what will we need to craft a compelling story? Here are a few ideas:

1. Our first step will be to create snapshots of the network at the end of each school day to show how the network evolves over the study period.
2. We'll also want some essential network statistics to help support the graphs, preferably in a visual format that shows the network trends.
3. We might also need to provide more of a narrative than we would in an interactive network situation. As I noted a moment ago, we need to tell the story of this dynamic network.

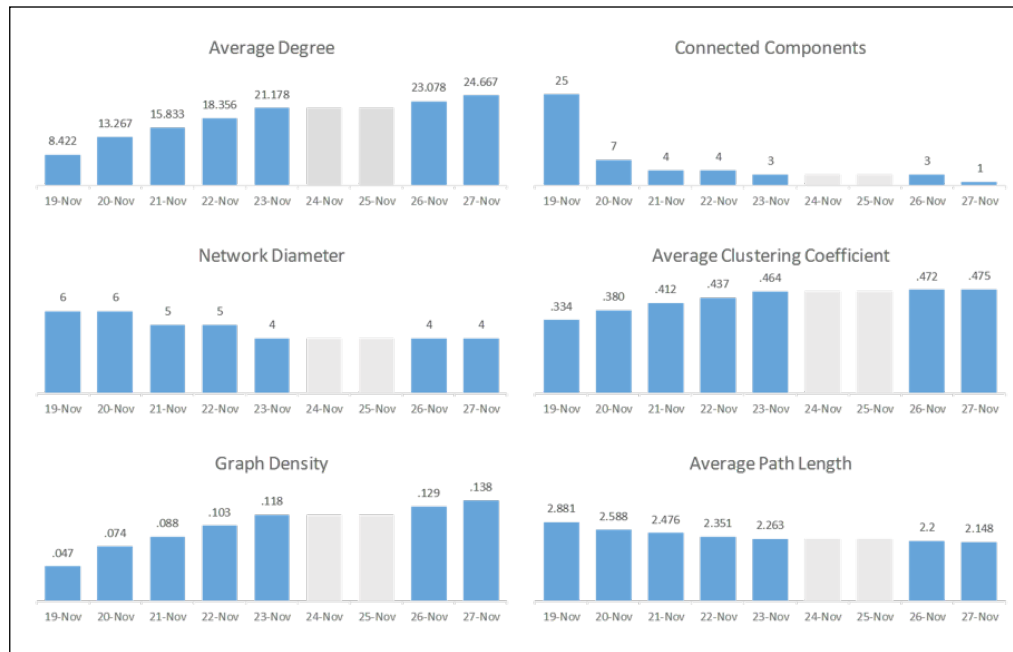
I'm going to provide a glimpse into what the final visualization will look like. The entire PDF is available at <https://app.box.com/s/177yit0fdovz1czgcecp>.

Here's one of our visuals that shows the network at the end of day 1 (November 19, 2012):



Dynamic high school network at the end of day 1, November 19, 2012

There will be a series of these visuals (one each for the 19th, 21st, 23rd, and 27th), showing the evolution of the network over the course of the study period. Viewers will be able to detect some of the changes, especially when comparing the first and last day. Yet the story would be incomplete without sharing some of the critical graph statistics that fill in the gaps. So with each end of the day snapshot, I also captured several critical network structure measures and pulled them together in a single page. This allows users to see how the network evolved, and when major changes (if any) took place:



Statistical measures tracking network evolution

Notice that the two weekend days have been grayed out so as not to lead the user to the wrong conclusions. Placing all of these graphs on a single page lets the viewer see the evolution of the network in statistical terms, perhaps confirming their initial visual impression. Key changes in the network are also called out in the final visualization, completing the entire story.

## Anticipating the future of network analysis

So we've seen in the course of this book the many wondrous things that can be done using Gephi, and we've still really just scratched the surface. There are now so many opportunities to process, analyze, and visualize network datasets that it can be overwhelming at times. Countless examples of Twitter, Facebook, and other social networks proliferate across the Web, some of them thoughtful and powerful, some others far less so. Likewise, we have all seen many instances of collaboration networks, protein networks, cell structure networks, citation networks, and so on. Where does it all go from here? Will the future look like the present, simply with more examples?

One outcome that seems a near certainty will be the increasing involvement of end users in interacting and perhaps even participating in the creation of the eventual network display. Imagine a case where users can adjust and adapt to the graph on a real-time basis, affecting the outcome of the network. Look no further than many of the massive multiplayer online games for proof of what can be done at the user level. Perhaps game theory examples will involve multiple end users acting according to their own preferences of the moment, with the network adapting and evolving to reflect the instantaneous input of hundreds or even thousands of users. These real-time examples could change much of today's analysis from a dependence on static historical datasets to an environment that is plugged into the ongoing changes in an evolving network.

Imagine also the possibilities of delivering additional insight by providing relevant information behind every node and edge in a network. In today's world, we have the ability to craft templates and provide generally static information to accompany our network graphs. In some cases, this is abetted by external links that provide additional information. This approach, while useful, puts the burden on the user and provides something less than a seamless process. How can we improve this process?

In the near term, are there untapped opportunities where network analysis can work more closely with **ontological** resources to deliver richer information and interactivity to the end user? In the slightly more distant future could all relevant information be accessed within the bounds of a single network page, enabling users to travel through the network as if they were in a gaming environment? Consider the possibilities of adding relevant information that activates multiple senses, and how users could leverage the power of sight, sound, and touch to examine and understand the network. Users could perhaps access a sort of virtual world, but one that is based totally on the reality of moment to moment interactions, not on some fictional construct created by game designers.

Beyond the evolution of network graph analysis through technological advances there is the question of additional use cases. We have seen the many instances of social media, citation, collaboration, biological, and infrastructure networks, but are there other opportunities to leverage network analysis to understand systems? Could network graph analysis be a powerful tool for exploring topics such as chaos theory, tracking the impact of specific stimuli on a surrounding network? Might we also be able to explore connections within the body that could help combat disease or infection? Are there greater opportunities to use these tools to better predict long-term effects of specific short-term decisions, in a modeling sort of environment?

These are merely a few examples for where network graph analysis could be heading. You no doubt have additional thoughts about the future and where the opportunities will emerge. Regardless of our individual ideas, I believe we can all agree that the future will be filled with exciting opportunities to employ these methods to better understand our world and all of its interactions. The future of network graph analysis and all its manifestations promises to be both challenging and rewarding, and will evolve and grow through the efforts of readers like yourself.

I also invite you to view my personal explorations and discussions at <http://visual-baseball.com/wordpress/?gallery=network-graphs> as well as the great work of users in the Gephi communities on Facebook and LinkedIn. Some truly exceptional work is being shared in these forums.

## Summary

In this final chapter, we covered a few key areas that focused on Gephi and network graph analysis at a more holistic level.

We learned how to understand and ultimately enhance existing network graphs through the use of many of Gephi's capabilities. We learned how to use regex to provide additional insight into an existing network, particularly when the network data provides no obvious starting points for analysis. From there, we employed our more familiar skills of filtering, sizing, and coloring graph elements to create a more accessible user experience.

We then created two projects from scratch using publicly available datasets. In the first example, we took a static network, enhanced its appearance, and made it interactive on the Web. In our second case, we created a dynamic network by building time intervals that documented member patterns, and ultimately created a time lapse project as a PDF output.

In our final section, we took a look at the future of network graph analysis and where it might be heading. We looked at the possibilities enabled by advanced technology as well as opportunities for horizontal expansion through the application of network analysis methods to additional disciplines.

I hope this book has helped provide you with opportunities to leverage Gephi more effectively to address your own network analysis needs. In addition, my hope is that I have triggered some additional ideas that you might pursue as you continue to investigate this fascinating yet largely untapped field.





# Data Sources and Other Web Resources

This appendix lists various data sources, web resources, and references which you can use as supplementary material for the best use of this book.

## Data sources

Here are a number of data sources that provide network data as well as links to other data sources:

- **Stanford Network Analysis Project (SNAP)** has a website <http://snap.stanford.edu/data/links.html> with links to many network datasets (large and small).
- Mark Newman is one of the most recognized network scientists who maintains a number of network datasets at his site <http://www-personal.umich.edu/~mejn/netdata/>.
- Albert-László Barabási is one of the foremost network scientists and runs the Barabasi Lab at Northeastern University. More information about him can be found at <http://barabasilab.com/rs-netdb.php>.
- The Gephi wiki has many network datasets as well as a number of files that are already in various proprietary formats. These can be found at <https://wiki.gephi.org/index.php/Datasets>.
- **KONECT** is the **Koblenz Network Collection** available at the University of Koblenz-Landau. The University's site houses one of the best collections of network data, and provides easily accessible symbols that identify specific network attributes, which can be found at <http://konect.uni-koblenz.de/>.

## Web resources

Here are just a few websites that provide practical or theoretical insights into network analysis. There are, of course, dozens more that can be discovered by a simple web search:

- The **Center for Network Science (CNS)** at Central European University hosts a variety of resources of interest for network scientists and practitioners at <https://cns.ceu.hu/>.
- The Carnegie-Mellon University houses the **Center for Computational Analysis of Social and Organizational Systems (CASOS)**. A variety of informational and educational resources are provided that will be of interest to network analysts. More information can be found at <http://www.casos.cs.cmu.edu/index.php>.
- NetLogo is a site where you can run model simulations online or using the downloaded NetLogo software. Many interesting examples are available at <http://ccl.northwestern.edu/netlogo/index.shtml> to help you understand network growth and evolution patterns.

## Import processes

For additional details on how to import datasets, refer to *Chapter 4, Network Patterns*, from *Network Graph Analysis and Visualization with Gephi*, available at <https://www.packtpub.com/big-data-and-business-intelligence/network-graph-analysis-and-visualization-gephi>.

## Bibliography

There are hundreds of papers available on the Web dealing with various aspects of network graph analysis, as well as multiple books available in PDF versions. This list provides some of the resources I have found most useful while preparing for and writing this book:

- Barabási, A. *Linked: The New Science of Networks*. 2014.
- Bearman, P., Moody, J., Stovel, K. *Chains Of Affection: The Structure of Adolescent Romantic and Sexual Networks*. *American Journal of Sociology*. 2004. 44-91. Retrieved April 19, 2014.
- Easley, D., Kleinberg, J. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press. 2010.

- Kourtellis, N., Alahakoon, T., Simha, R., Iamnitchi, A., Tripathi, R. *Identifying high betweenness centrality nodes in large social networks: Social Network Analysis and Mining*. 899-914. Retrieved November 2, 2014.
- Lattanzi, S., Sivakumar, D. *Affiliation networks*. 2009.
- Newman, M. *The Structure and Function of Complex Networks*, SIAM Review. 2003. 167-167. Retrieved March 27, 2014.
- Steen, M. *Graph Theory and Complex Networks: An Introduction*. 2010.
- Tang, J., Musolesi, M., Mascolo, C., Latora, V. *Temporal distance metrics for social network analysis*. 2009. Retrieved March 21, 2014.
- Wimmer, A., Lewis, K. *Beyond and below racial homophily: ERG models of a friendship network documented on Facebook*. American Journal of Sociology. 2010. 583-642. Retrieved June 22, 2014.



# Index

## A

### **Attractive and Repulsive Forces (ARF) layout**

about 36, 61, 71, 93  
General attraction force 72  
Neighbor attraction force 71  
Precision force 72  
Repulsive force 72  
testing 97-100

### **attribute-based DNA**

about 262  
data, preparing 262, 263  
dynamic attribute networks,  
implementing 264-273  
dynamic attribute networks,  
viewing 264-273

### **attributes filter**

about 152  
Equal function 152  
Inter Edges 152  
Intra Edges 152  
Non-null condition 153  
Partition Count filter 153  
Partition filter 153  
Range filter condition 153

### **average path length, network measures 183**

### **average path length, network statistics 200**

## B

### **Barabasi-Albert scale free model 128**

### **betweenness centrality 15, 188, 195, 204, 205**

### **bibliography 346, 347**

### **bipartite graph 85**

### **brush function 49**

## C

### **Center for Computational Analysis of Social and Organizational Systems (CASOS)**

URL 346

### **Center for Network Science (CNS)**

URL 346

### **centrality 14, 15**

### **centrality measures, graph statistics**

about 185, 186  
betweenness centrality 188  
closeness centrality 187  
degree centrality (undirected graphs) 186  
eigenvector centrality 187  
in-degree centrality (directed graphs) 187  
out-degree centrality (directed graphs) 187

### **centrality statistics**

betweenness centrality 204, 205  
closeness centrality 202, 203  
degree centrality 201, 202  
eigenvector centrality 203, 204

### **centrality statistics, interpreting**

about 193  
betweenness centrality 195  
closeness centrality 194  
degree centrality 193  
eigenvector centrality 195  
in-degree centrality 194  
out-degree centrality 194

### **Chinese Whispers algorithm 221**

### **Chinese Whispers plugin**

using 233-237

### **Circular layout**

about 35, 78, 79  
Concentric layout 80  
Dual Circle layout 80, 81

- closeness centrality** 187, 194, 202, 203
- cluster analysis** 123
- clustering** 29, 122-124
- clustering and neighborhood measures,**
  - graph statistics**
    - about 188
    - clustering coefficient 188
    - Link Communities 189
    - modularity 189
    - neighborhood overlap and embeddedness 189
    - number of triangles 189
- clustering statistics**
  - about 205
  - clustering coefficient 205
  - embeddedness 207
  - Link Communities 206
  - modularity 206
  - number of triangles 206
- clustering statistics, interpreting**
  - about 196
  - clustering coefficients, interpreting 196
  - embeddedness 197
  - Link Communities statistic 197
  - modularity statistic 197
  - number of triangles 197
- CmapTools**
  - URL 116
- community detection** 219
- complex filters**
  - about 168
  - INTERSECTION filter 175-177
  - Mask, working with 175-177
  - multiple filter conditions, applying 168, 169
  - subfilters, using 169-174
  - UNION operator 178, 179
- Concentric layout**
  - about 35, 80, 93
  - testing 101, 102
- connected components, network**
  - measures** 184
- connected graph** 17
- connections** 10
- connectivity** 13

- contagion**
  - about 11, 114-117
  - SIR model 117
  - network, viewing 131-135
- CSV files** 281
- CSV import, data laboratory** 24
- cycles** 12

## D

- data**
  - formatting, for Gephi 43
  - importing, into Gephi 44
- data laboratory**
  - about 23, 29
  - CSV import 24
  - Excel import 25
  - Graph file import 25
  - helper 30
  - manual entry 24
  - MySQL import 25
- data sources**
  - about 345
  - identifying 43
- degree centrality** 16, 193, 201
- degree centrality (undirected graphs)** 186
- Degree Range filter** 155
- density** 19
- diameter, network measures** 182
- diffusion**
  - about 119-122
  - network, viewing 136-139
- Directed Acyclic Graphs (DAG)**
  - layout** 78, 94
- DL files** 282
- DNA**
  - about 243
  - topology-based DNA 245
  - types 244
  - uses 244
- Dual Circle layout** 80, 81, 94
- dynamic attribute networks**
  - implementing 264-273
  - viewing 264-273

**dynamic filtering** 152

**dynamic GEXF files**

creating 274-277

URL 274

**Dynamic Network Analysis.** *See* DNA

**dynamic network, topology-based DNA**

generating 245, 246

implementing 251

viewing 251

## E

**eccentricity, network measures** 183

**eccentricity, network statistics** 200

**edge betweenness, network**

statistics 200, 201

**edge betweenness statistic,**

network measures 185

**Edge Cut** 75

**edge pencil tool** 49

**edges**

about 153

Edge Weight filter 153

filtering 160-162

Self-Loop filter 153

**Ego Network function** 155

**eigenvector centrality** 15, 187, 195, 203, 204

**embeddedness, clustering statistics** 197

**Equal filter**

regex function, applying 159, 160

using 157-159

**Equal function** 152

**Erdos number, network measures**

about 184

URL 321

**Excel import, data laboratory** 25

**expansion** 221

**exports**

about 30

ExportToEarth 31

Graph Streaming plugin 30

Seadragon Web Export 30

Sigmajs Exporter 30

## F

**filtering**

about 150

complex filters 168

Equal filter 157, 158

functions 151, 152

primary filtering 151

simple filters 156

tab 27

**Force Atlas 2 layout** 73, 74, 94

**Force Atlas 3D layout** 74

**Force Atlas layout** 56, 57, 72, 73, 94

**force-based layouts**

attraction 70

gravity 71

repulsion 71

**friends of friends' network** 166

**Fruchterman-Reingold algorithm** 74, 94

**Fruchterman-Reingold layout** 58

## G

**GDF files** 282, 283

**generators**

about 31-33

using 126-131

**geodesic path** 11

**geographic layouts**

about 83

Geo layout 83

Maps of Countries layout 84

**Geo layout** 83, 94

**Gephi**

about 7, 8

existing networks 316-323

overview 22

projects, creating 324, 325

toolkit, URL 8

wiki, URL 316, 345

**Giant Component filter** 155

**Give color to nodes (Tools)** 36



## **graph**

- analyzing 48, 62-64
- applications 8
- collaboration 8
- exporting 68
- information linkages 9
- modifying 49, 65-67
- natural-world networks 9
- technological networks 9
- Who-Talks-to-Whom graphs 9

## **graph aesthetics**

- about 107
- example 108-111

## **graph density, network measures 183**

## **graph density, network statistics 200**

## **Graph Exchange XML Format (GEXF) files**

- about 283
- URL 283

## **graph file exporters**

- about 280, 281
- CSV files 281
- DL files 282
- GDF files 282, 283
- Graph Exchange XML Format (GEXF) files 283
- GraphML files 284, 285
- Graph Modeling Language (GML) files 284
- NET files 285
- VNA files 285, 286

## **graphing needs, Miles Davis network**

### **example**

- about 87-89
- analysis goal 89
- dataset parameters 89, 90
- interactivity 92, 93
- network behaviors 91
- network density 91
- network display 91
- temporal elements 92

## **GraphML files**

- about 284
- URL 285

## **Graph Modeling Language (GML) 284**

## **graph statistics**

- about 182

- centrality measures 185, 186
- centrality statistics, interpreting 193
- clustering and neighborhood measures 188
- clustering statistics, interpreting 196
- interpreting 190
- network measures 182
- network measures, interpreting 190-193
- used, for filtering 207-215

## **graph window 25**

## **GUESS**

- URL 282, 283

## **H**

## **hairball effect 34**

## **Hiveplot layout 34, 82, 95**

## **homophily**

- about 19, 123, 124
- identifying 144-147

## **Hyperlink-Induced Topic Search (HITS)**

- function, network measures 184

## **I**

## **image exporters**

- about 280, 286
- PNG export 286, 287
- SVG export 288

## **Image Preview tool**

- URL 292

## **import processes 346**

## **in-degree centrality 194**

## **in-degree centrality (directed graphs) 187**

## **In Degree Range filter 155**

## **inflation 221**

## **initial graph layout**

- viewing 45, 46

## **Inkscape**

- PDF file, editing 293, 294

## **Inter Edges 152**

## **INTERSECTION filter 175, 177**

## **INTERSECTION operator 154**

## **intervals, topology-based DNA**

- creating, in existing project 251-253

## **Intra Edges 152**

## **Isometric layout 84, 95**

## K

**KONECT**  
URL 345

## L

**Layered layout** 35, 85, 95  
**layout**  
about 34  
additional 84  
additional layout tools 86  
ARF layout 36, 71, 72  
ARF layout, testing 97-100  
Circular layout 35, 78, 79  
Concentric layout 35, 101, 102  
Force Atlas 2 layout 73, 74  
Force Atlas 3D layout 74  
Force Atlas layout 72, 73  
Force-based layouts 70, 71  
Fruchterman-Reingold algorithm 75  
geographic layouts 83  
Hiveplot layout 34  
Isometric layout 85  
Layered layout 35, 85  
Multipartite layout 34, 85  
Network Splitter 3D 86  
OpenOrd algorithm 75  
OpenOrd layout 35  
Radial Axis layout, testing 103-105  
radial layouts 81  
selecting 47  
selection, criteria 106  
strengths 93-96  
testing 97  
tree layouts 77, 78  
types 69, 70  
weaknesses 93-96  
Yifan Hu algorithm 76  
Yifan Hu multilevel approach 77  
Yifan Hu Proportional layout 76  
**layouts tab** 28  
**Link Communities, clustering and neighborhood measures** 189  
**Link Communities statistic, clustering statistics** 197  
**links** 10  
**Loxa Web Site Export** 298, 300, 308-313

## M

**Maps of Countries layout** 84, 95  
**Markov Clustering plugin**  
about 221  
using 237-241  
**Marvel Social network** 316  
**MASK (Edges) operator** 154  
**modularity, clustering and neighborhood measures** 189  
**modularity statistic, clustering statistics** 197  
**Multipartite layout** 34, 85, 95  
**MySQL import, data laboratory** 25

## N

**neighborhood overlap and embeddedness, clustering and neighborhood measures** 189  
**Neighbors Network filter** 155  
**NET files** 285  
**NetLogo**  
URL 346  
**network**  
clustering 140-144  
contagion 20  
contagion network, viewing 131-135  
diffusion 20  
diffusion, viewing 136-139  
growth 21, 22  
growth, patterns 125, 126  
**network analysis**  
future 341, 342  
**network diameter, network statistics** 200  
**network graph**  
analysis primer 10  
analyzing 48  
connectivity 13  
cycles 12  
data, formatting for Gephi 43  
data, importing into Gephi 44  
data sources, identifying 43  
example graph, creating 51  
exporting 50, 51  
final output, determining 42  
idea or topic, identifying 40, 41  
initial graph layout, viewing 45, 46  
layout, selecting 47

- modifying 49
- paths 11
- paths and connectivity 11
- process flow, proposed 40
- structure 13
- network graph, example graph**
  - appropriate layout, selecting 56
  - data, formatting for Gephi 52-54
  - data, importing 54
  - data source, finding 52
  - Force Atlas layout 56, 57
  - Fruchterman-Reingold layout 58
  - initial network, viewing 55
  - Radial Axis layout 59
  - topic, identifying 51, 52
  - Yifan Hu layout 60
- network graph, structure**
  - about 13, 14
  - behaviors 20
  - centrality 14-17
  - clustering 19
  - components 17, 18
  - density 19
  - giant component 18
  - homophily 19
- network measures, graph statistics**
  - about 182
  - average path length 183
  - connected components 184
  - diameter 182
  - eccentricity 183
  - edge betweenness 185
  - Erdos number 184
  - graph density 183
  - Hyperlink-Induced Topic Search (HITS)
    - function 184
- network measures, interpreting**
  - about 190-192
  - average path length 192
  - diameter 191
  - eccentricity 191
  - graph density statistic 192
  - HITS 192
- Network Splitter 3D layout 86, 96**
- network statistics**
  - about 199
  - average path length 200
  - centrality statistics 201
  - eccentricity 200
  - edge betweenness 200
  - graph density 200
  - network diameter 200
- node pencil tool 49**
- Non-null condition 153**
- NOT (Edges) operator 154**
- NOT (Nodes) operator 154**
- number of triangles, clustering and neighborhood measures 189**
- number of triangles, clustering statistics 197**
- Num Iterations 76**
- Num Threads 76**

## O

- OpenOrd algorithm 75**
- OpenOrd layout 35, 96**
- operator**
  - about 154
  - INTERSECTION operator 154
  - MASK (Edges) operator 154
  - NOT (Edges) operator 154
  - NOT (Nodes) operator 154
  - UNION operator 154
- out-degree centrality 194**
- out-degree centrality (directed graphs) 187**
- Out Degree Range filter 155**

## P

- painter function 49**
- Pajek program**
  - URL 285
- Partition Count filter 153**
- Partition filter**
  - about 153
  - using 162, 163
- partitioning 222-227**
- partitioning and clustering, examples**
  - about 222-227

- Chinese Whispers plugin, using 233-237
- color and size options, using 228-230
- manual graph segmentation 231-233
- Markov Clustering plugin, using 237-241
- Ranking tab 228
- partitioning and clustering, options**
  - about 218, 219
  - Chinese Whispers algorithm 221
  - manual settings 220
  - Markov clustering 221
  - Partition tab 219
  - Ranking tab 220
- Partition tab** 219
- paths** 11
- PDF export**
  - about 292
  - PDF file, editing in Inkscape 293, 294
- plugins**
  - about 28, 29
  - additional 36
  - clustering 29
  - data laboratory 29
  - data laboratory, helper 30
  - exports 30
  - generators 31-33
  - Give color to nodes (tools) 36
  - layout 34
  - Link Communities (metrics) 36
- PNG export** 286, 287
- preview window**
  - about 26
  - PDF option 26
  - PNG option 26
  - SVG option 26
- primary filtering**
  - about 151
  - attributes 151-153
  - edges 151-153
  - operator 151-154
  - topology 151-155
- primary windows**
  - about 23
  - data laboratory 23
  - graph window 25
  - preview window 26

- project, high school network**
  - about 333
  - creating, as PDF 338-340
  - network in Gephi, exploring 334-338
- project, Newman NetScience dataset**
  - about 325
  - network in Gephi, exploring 326-330
  - project, deploying to Web 331, 332

## R

- Radial Axis layout**
  - about 59, 82, 83, 96
  - testing 103-105
- radial layout**
  - about 81
  - Hiveplot layout 82
  - Radial Axis layout 82, 83
- Range filter** 153
- Ranking tab**
  - about 220
  - working with 228
- RDF**
  - URL 152
- Regular Expression (regex)**
  - about 318
  - applying 159, 160
- repulsion strength** 72

## S

- Scalable Vector Graphics.** *See* SVG export
- Seadragon** 300, 302
- Seadragon Web Export** 30, 294, 295
- secondary windows (tabs)**
  - about 27
  - filtering tab 27
  - layouts tab 28
  - statistics tab 27, 28
- Self-Loop filter** 153
- semantic filtering** 152
- Sigma.js Exporter** 30, 296-308
- simple filters**
  - about 156, 157
  - edges, filtering 160-162

- Equal filter, using 157-159
- Partition filter, using 162, 163
- regex function, applying 159, 160
- topology filters 164-167
- SIR model (Susceptible, Infectious, and Removed) 117**
- SIRS model 118**
- SIS model 118**
- six degrees of separation 63**
- sizer function 49**
- small world networks 129**
- Social Network Analysis (SNA) 8**
- Stanford Large Network Dataset Collection**
  - URL 43
- Stanford Network Analysis Project (SNAP)**
  - URL 345
- statistical measures, application**
  - about 198
  - basic statistical applications 198
  - graph statistics, used, for filtering 207-215
- statistics tab 27, 28**
- subfilters, complex filters**
  - using 169-174
- SVG export**
  - about 288
  - SVG file, editing with Inkscape 288-292

## T

- ties 10**
- time intervals, topology-based DNA**
  - about 246, 247
  - adding, to new project 254
- timelines, topology-based DNA**
  - about 248, 249
  - applying 259, 260
  - as filters 260, 261
  - working with 258
- topology**
  - Degree Range filter 155
  - Ego Network function 155
  - Giant Component filter 155
  - In Degree Range filter 155
  - Neighbors Network filter 155
  - Out Degree Range filter 155

- topology-based DNA**
  - about 245
  - data, importing 249, 250
  - data, preparing 249, 250
  - dynamic network, generating 245, 246
  - dynamic network, implementing 251
  - dynamic network, viewing 251
  - existing GEXF file, using 254, 255
  - multiple timeframes, adding 255-257
  - time intervals 246, 247
  - time intervals, adding to new project 254
  - time intervals, creating in existing project 251-253
  - timelines 248, 249
  - timelines, applying 259, 260
  - timelines, as filters 260, 261
  - timelines, working with 258
- topology filters, simple filters 164-167**
- topology, primary filtering 154, 155**
- tree layouts**
  - about 77, 78
  - Directed Acyclic Graphs (DAG) layout 78

## U

- UCINET program**
  - URL 282
- UNION operator 154, 178, 179**

## V

- VNA files 285, 286**

## W

- Watts-Strogatz small world Alpha model 130**
- web exporters**
  - about 280, 294
  - Loxa Web Site Export 294, 298-300
  - Seadragon Web Export 294, 295
  - Sigma.js Exporter 294, 296-298
- web graph**
  - exporting 300
  - Loxa Web Site Export 308-313
  - Seadragon 300-302
  - Sigma.js Exporter 302-308

web sources 346  
Who-talks-to-Whom graphs 9

## Y

Yifan Hu algorithm 76, 96  
Yifan Hu layout 60  
Yifan Hu multilevel approach 77, 96  
Yifan Hu Proportional layout 76, 96





## Thank you for buying Mastering Gephi Network Visualization

### About Packt Publishing

Packt, pronounced 'packed', published its first book, *Mastering phpMyAdmin for Effective MySQL Management*, in April 2004, and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution-based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern yet unique publishing company that focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website at [www.packtpub.com](http://www.packtpub.com).

### About Packt Open Source

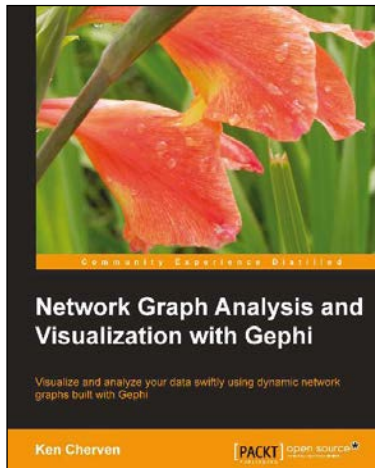
In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around open source licenses, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each open source project about whose software a book is sold.

### Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to [author@packtpub.com](mailto:author@packtpub.com). If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, then please contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.





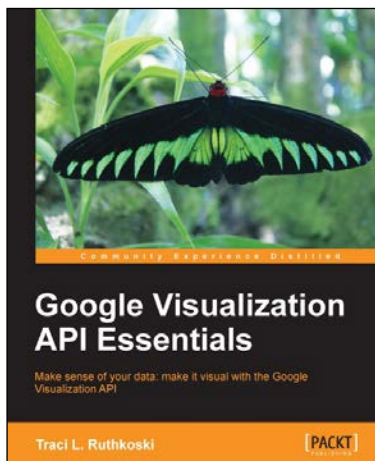
## Network Graph Analysis and Visualization with Gephi

ISBN: 978-1-78328-013-1

Paperback: 116 pages

Visualize and analyze your data swiftly using dynamic network graphs built with Gephi

1. Use your own data to create network graphs displaying complex relationships between several types of data elements.
2. Learn about nodes and edges, and customize your graphs using size, color, and weight attributes.
3. Filter your graphs to focus on the key information you need to see and publish your network graphs to the Web.



## Google Visualization API Essentials

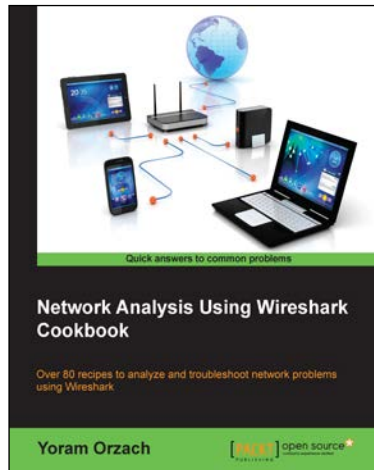
ISBN: 978-1-84969-436-0

Paperback: 252 pages

Make sense of your data: make it visual with the Google Visualization API

1. Wrangle all sorts of data into a visual format, without being an expert programmer.
2. Visualize new or existing spreadsheet data through charts, graphs, and maps.
3. Full of diagrams, core concept explanations, best practice tips, and links to working book examples.

Please check [www.PacktPub.com](http://www.PacktPub.com) for information on our titles



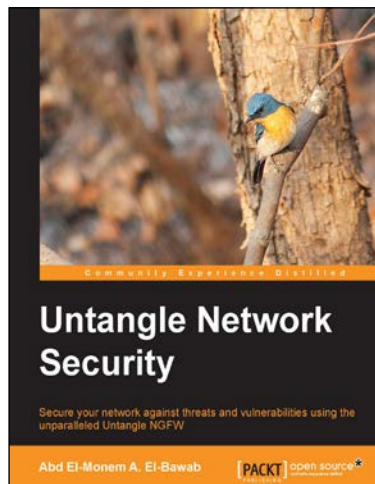
## Network Analysis using Wireshark Cookbook

ISBN: 978-1-84951-764-5

Paperback: 452 pages

Over 80 recipes to analyze and troubleshoot network problems using Wireshark

1. Place Wireshark in the network and configure it for effective network analysis.
2. Use Wireshark's powerful statistical tools and expert system for pinpointing network problems.
3. Use Wireshark for troubleshooting network performance, applications, and security problems in the network.



## Untangle Network Security

ISBN: 978-1-84951-772-0

Paperback: 368 pages

Secure your network against threats and vulnerabilities using the unparalleled Untangle NGFW

1. Learn how to install, deploy, and configure Untangle NG Firewall.
2. Understand network security fundamentals and how to protect your network using Untangle NG Firewall.
3. Step-by-step tutorial supported by many examples and screenshots.

Please check [www.PacktPub.com](http://www.PacktPub.com) for information on our titles