Mastering Gephi Network Visualization

Produce advanced network graphs in Gephi and gain valuable insights into your network datasets

Ken Cherven
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 “Fundamentals of Complex Networks and Gephi”
- A synopsis of the book’s content
- More information on Mastering Gephi Network Visualization

About the Author

Ken Cherven is a Detroit-based data visualization and open source enthusiast, with 20 years of experience working with data and visualization tools. In addition to Gephi, he has worked with a variety of open source tools, including MySQL, SpagoBI, JasperServer, D3, Protovis, Omeka, QGIS, Leaflet, and Exhibit. He also has considerable experience using corporate software tools from Microsoft, Cognos, Tableau, and Oracle.

Acknowledgments

I would like to thank the members of my family for their patience and understanding over the course of several months spent working on this book. This always starts with my wife, Karen, and extends to my children, Kellen, Kristopher, and Katie, as well as my always helpful mother-in-law, Carole Young.

This book would not have been possible without the considerable efforts of a group of thorough technical and content editors. I would like to sincerely thank Mollie Taylor, Ladan Doroud, Miro Marchi, Ted Polley, George Vega Yon, Marta Castellani, and Manasi Pandire for their considerable efforts to make this the best possible book. All of your input has been noted, and many improvements have been incorporated.

A special thanks also to Amey Varangaonkar at Packt Publishing for managing the entire process while also making recommendations that will result in a more enjoyable reading experience. Thanks also to others who helped in the early stages by providing useful feedback to get the book started. This list includes Joanne Fitzpatrick and Richard Gall at Packt Publishing, plus Gephi community members, Randy Novak, Mike Hughes, Matthieu Totet, Marco Valli, Gerry Wilson, and Carlos Benito Amat.

Finally, I would like to thank the creators and maintainers of Gephi for providing such a powerful tool that allows users to explore the fascinating world of network science. Thanks also to the growing community of enthusiasts who use Gephi to create some remarkable visualizations. My hope is that this book will make it easier for you to tap into the power of Gephi and, perhaps, even provide a few new approaches to leverage this powerful tool.
Mastering Gephi
Network Visualization

Gephi has rapidly become one of the most utilized tools for the exploration and analysis of network data, as users seek to understand the relationships between groups of people, institutions, events, and other connected phenomena. At the same time, Gephi can help us understand serious topics such as disease transmission, the diffusion of ideas and innovation, and changes over time to community structures. With the ability to import a wide variety of data formats both open and proprietary, Gephi is truly moving toward democratizing network information.

This book aims to assist both new and experienced users in fully leveraging the immense potential of Gephi, regardless of whether the end goal is exploration, analysis, visualization, or some combination of each. While not every nuance of Gephi is covered in this volume, the topics in the book should go a long way toward improving your capabilities for effectively using Gephi.

What This Book Covers

Chapter 1, Fundamentals of Complex Networks and Gephi, provides background into the world of complex networks and how we can use Gephi to explore and analyze network patterns.

Chapter 2, A Network Graph Framework, provides a process for creating and developing network visualizations using Gephi.

Chapter 3, Selecting the Layout, will introduce many available layout algorithms in Gephi, and help you to select the most appropriate types based on the characteristics of your network data.

Chapter 4, Network Patterns, examines several critical network patterns, including contagion, diffusion, and homophily. We then use Gephi to explore and understand these behaviors.

Chapter 5, Working with Filters, provides multiple examples for how and when to use the powerful filtering capabilities provided within Gephi.

Chapter 6, Graph Statistics, provides readers with background on some key statistical network measures, followed by examples for how to effectively apply these methods in Gephi.
Chapter 7, Segmenting and Partitioning a Graph, provides insight into the multiple approaches that can be used to effectively segment a network, based on categorical or behavioral attributes. The use of size and color to partition a graph will be thoroughly explored.

Chapter 8, Dynamic Networks, will introduce the concept of Dynamic Network Analysis (DNA) and how time-based networks can be explored and understood in Gephi.

Chapter 9, Taking Your Graph Beyond Gephi, gives an overview of available export options, followed by several examples for creating visualizations by combining Gephi with external tools.

Chapter 10, Putting It All Together, incorporates many of the methods covered earlier in the book to create both revised and brand new visualizations. We'll also introduce some new methods that will allow for further network customization.

Appendix, Data Sources and Other Web Resources, lists out all the important references that you can use to understand the book in much more detail.
Welcome to the world of Gephi, the open source network graph and analysis tool. If you aren’t already familiar with Gephi, this chapter will provide you a brief primer on both Gephi and the many ways in which it can be used to address real-world network analysis. For those who are already familiar with the software, this section will inspire you to take advantage of the many opportunities provided by this powerful tool.

Advances in computing power have helped to revolutionize the science of network graph analysis by putting complex datasets into the hands of thousands of users, and thus leading to the evolution of tools that help practitioners make sense out of previously unmanageably large datasets. Gephi is one of the tools at the forefront of this revolution, with a growing user community that produces some exceptional visualization. Gephi is far more than just a tool for producing attractive graphs; it is also a powerful instrument for exploratory data analysis that enables users to learn more about a given network.

This chapter will cover the following topics:

- First, we will examine a range of graph applications that represent some of the many ways we might choose to use Gephi in order to address real-world data
- The next section will provide an overview of the network graph vernacular, which will cover the primary topics of connectivity, network structure, and network behaviors
Finally, we will begin to explore how we can use Gephi as the primary toolkit to move from the conceptual realm into a very hands-on, real-world application.

Note that this chapter and the remainder of the book are focused solely on the Gephi desktop software, and not the Gephi toolkit. The toolkit provides Java library modules that can be plugged into new Java applications to deliver core Gephi functionality. More information on the toolkit can be found online at http://gephi.github.io/toolkit/.

Let's get started with some potential applications that will begin to illustrate the number of ways in which we can use Gephi to create compelling network analysis and visualization.

**Graph applications**

One doesn't have to look far to recognize the enormous growth of network graphs as a means to explore and explain networks. *Social Network Analysis (SNA)* has certainly been the most visible subset of network graph analysis, with thousands of cases where users have mapped Facebook, Twitter, and LinkedIn peer networks. While this has been, and continues to be, a viable use of the approach, there are many lesser known, but frequently more compelling, datasets with highly interesting networks that are available for our exploration. In the next few sections, we will walk through some of the primary categories where we can access data and use Gephi to construct highly informative graphs by employing definitions laid out in the book *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*, by David Easley and Jon Kleinberg.

**Collaboration graphs**

Collaboration graphs represent one of the more frequently encountered categories in the world of network analysis. These graphs include networks where individual nodes are connected based on having some sort of collaborative relationship. The nodes might represent individuals or institutions; these graphs often depict collaborative research between universities and their staff within a specific discipline.
Who-talks-to-whom graphs
Another popular utilization of network analysis has been through the viewing of network graphs based on a variety of communication methods between the actors in a network, often within the confines of a single corporation, organization, or educational institution. These graphs can be constructed using e-mail or phone communications, and will often focus on the frequency of contact, thus exposing the true information flows and power structures within the organization.

Information linkages
Graphs that examine the flow of information across the Web are typical of this category of network analysis. These linkages can reference anything from connections between bloggers, pages on Wikipedia, or among scientific paper citation networks. This is a very popular type of graph, given the accessibility of information via the Web and its various applications.

Technological networks
This category often manifests itself through physical structures but, as David Easley and Jon Kleinberg note, there are underlying economic structures here as well, in the form of companies, regulatory bodies, and other organizations. In these sorts of networks, connections between nodes are likely to refer to a literal physical linkage, as in connections between routers or computers.

Natural-world networks
Another discipline that has received much attention through the use of networks is in the world of biology, where graphs are used to show relationships between predators and prey, neural networks within the brain, and a number of other science-based scenarios. Other network types are likely to emerge as well, and hybrids of these networks is also a possibility.

The idea here is to help stimulate thought processes about what sort of graphs you might be most interested in creating, and begin working toward their creation using Gephi. Specific examples of each of these network types are provided in the Appendix, Data Sources and Other Web Resources.
A network graph analysis primer

Network graphs are an element of what is termed as graph theory, defined by Merriam-Webster (http://www.merriam-webster.com/dictionary/graph%20theory) as:

"Mathematical theory of networks. A graph consists of vertices (also called points or nodes) and edges (lines) connecting certain pairs of vertices"

Stated simply, network graphs are collections of nodes (often called vertices) that are connected by edges (sometimes called connections, links, or ties) to form a graph. Nodes can be thought of as individual elements in a network that might represent persons, places, or objects that collectively define a network.

As you might have guessed, this definition could apply to an unlimited number of datasets, ranging from the obvious (Facebook or Twitter networks) to less explored examples, such as connections between staff, students, and facilities at a University; teammates in a baseball team; or musicians within a specific genre of music. In these cases, the nodes will represent the individual entities (people or structures), while the edges act as the connections that link them to one another. Each of your Twitter followers, for instance, represents individual nodes. They are then connected to you (a node) through a series of connections (edges). There are literally hundreds of thousands of opportunities to apply the principles of graph theory to interesting datasets spanning thousands of fields of interest.

So what can we learn from the principles of graph theory beyond creating a compelling visualization of a complex network? By the way, I will never downplay the visual output aspect of network graphs, as I am a strong believer in the power of aesthetics to enhance the story we are telling through our data. For example, refer to Beautiful Visualization: Looking at Data through the Eyes of Experts, edited by Julie Steele and Noah Ilinsky, O’Reilly Media. Still, there is much more to the basics of graph theory than a pretty picture. Let's examine a few of the most prominent principles in the next several sections.
Chapter 1

Paths and connectivity

We can learn a great deal about how a network functions by looking at all of the connections within the graph and by understanding how they are structured. Perhaps the graph is loosely structured, with few connections between nodes, or it could be densely connected, or a combination of both with dense clusters interspersed with gaps or structural holes in the network. Why is this important? One example would be to understand the potential for the spread of an infectious disease (known as a contagion in network terms) across a large network. A network with many structural holes will not support the rapid spread of a disease, while a densely connected network can facilitate this spread. There are many such cases where network structure is critical to understanding the behavior of elements within the network. The following sections will touch on some of the key concepts employed to help us better comprehend the structure of a network, and how its member nodes interact with one another.

Paths

A path is quite simply defined as the set of connections required for one node to interact with another node. We can use paths to understand the shortest distance between nodes, or perhaps to determine the shortest route to reach a distinct cluster of nodes. The following figure illustrates an example of a path for Node A to reach Node E, using the shortest path, shown by the bold edges. The shortest path is also known as the geodesic path. Note that the path could also have passed through Node D on the way to Node E, but this is not the most efficient path, unless the direct connection from C to E is severed at some point.

In some cases, a path might involve passing through a specific node more than once, but more often, this will not be true, especially if we are attempting to minimize the path distance. If our path does not repeat any nodes, it can be termed a simple path, as illustrated in the preceding figure.
Cycles

A cycle is an important variant of a non simple path, where there are a minimum of three edges and the first and last nodes are the same. All other nodes must be distinct; the cycle cannot traverse any of these nodes more than once. Cycles are critical to understand shortest paths through the network. Here is a simple cycle diagram—let's suppose we start at Node A:

![A cycle](image1)

This cycle is easy to follow, as we would simply move from A to B, C, D, E, and return to A in order to complete the cycle. We could also move in the reverse order, moving in a clockwise direction starting at Node E. This would become a little more complex, when there were additional nodes that do not flow around the perimeter as we have shown in the following figure. For example, a cycle could not move through the newly added Node F, since it would need to pass through Node C a second time to complete the entire path:

![A cycle with extra node F](image2)
Connectivity

Connectivity is defined as the degree of connectedness of a graph, and can be measured using several formulas. At its essence, connectivity is a measurement of the robustness of a graph, as defined by the relative number of connections within a network. Networks with low connectivity are inherently fragile, as the removal of a small number of edges serves to weaken the network, and can actually disconnect some members from the components of the graph.

A few of the more common forms of connectivity measures are the beta, alpha, and gamma indexes. The beta index is a simple measure that looks at the number of edges divided by the number of nodes in a graph. Very simple networks will have a score of less than one, while more complex, densely connected graphs will exceed a value of one, and might go much higher in many instances. This is a very simple equation: $\frac{e}{v}$, where $e$ equals the number of edges, and $v$ represents the number of nodes.

An alpha index evaluates the number of cycles in a graph relative to the possible number of cycles. At one extreme, a simple tree network would have an alpha value of zero, as there is no way to cycle through the network without repeating nodes. A perfectly connected network would have a score of one, but this is both rare and impractical, as this would indicate an inefficient network.

Finally, a gamma index measures the number of actual or observed links relative to the number of possible links, which gives us a value between zero and one. Scores closer to one indicate a more densely connected graph, although it is unusual to find a network that approaches that level. The gamma index is particularly useful to assess temporal (time-based) changes in a network. There are two equations, one for a planar graph with no crossing edges, and a second for non-planar graphs.

More details on connectivity and its various measures will be provided to you in Chapter 4, Network Patterns.

Network structure

We have thus far focused on the concepts of paths, cycles, and connectivity, all of which help us to understand the interactions within the graph and even provide us with some statistical measures of the composition of the network. Yet these approaches fall short to convey all the information about the overall structure of the graph, such as how influential individual nodes or clusters are within the network. Fortunately, there are many ways in which we can statistically measure the structure of a network. If our graph is limited to a small number of nodes and edges, it is not difficult to see connectivity patterns, node groupings, and the overall topology of the graph, and we might not be terribly concerned with the statistical output.
However, when the network grows to more than a few dozen nodes, simple visual assessment will not provide us with all of the information within the graph, so we need to rely on more sophisticated formulas that provide us with detailed insights into the data and its structure.

The following sections will provide some details on many of the primary measures that tell us more about our graph. Further details are provided in Chapter 6, *Graph Statistics*. There are also many additional sources that provide further details on these, as well as more advanced network concepts, statistics, and theory. If you have not already consulted some of these resources, I would encourage you to do so. A listing of many available sources is provided in the *Bibliography* section of Appendix, *Data Sources and Other Web Resources*.

**Centrality**

One of the key constructs within network graph analysis is the idea of centrality, where we make an attempt to understand the relative influence of individual nodes within the network. As one might anticipate, there are several ways we can measure centrality, with each method providing a different definition, and often, very different results. Let's assume in each of the following cases that we are examining a subset of a network, rather than its entirety. Each centrality measure will be measured across an entire network, but we will use the following measures for illustrative purposes. The general principles do not differ.

There are four primary centrality measures to explore, which we will look at in no particular order. The first measure of centrality is *closeness centrality*, a measure of the proximity of a selected node to all other nodes within the graph. A node with strong closeness centrality would typically have very short paths to all other nodes within the network. Note that the result will be a lower average number, as we are talking about how many steps it takes to reach all other nodes. Here is a simple example:

![Diagram of a network with nodes A, B, C, D, E, F, and G, illustrating closeness centrality.](image)
Note that Node D, despite having direct connections to just three of the six remaining nodes, has a maximum distance of two to reach any other point in the graph, while all other nodes have paths that might require three or even four steps. The central location of D makes traversing the graph very simple. Generally speaking, we would expect this type of node to lie at or near the physical center of a graph, although this is not always the case. In any event, this category of node is very prominent within the graph, and is also likely to have strong degree centrality, which we will discuss in a moment.

Another key measure is betweenness centrality, which often returns a very different result than the other centrality measures. In this case, we will find nodes that are highly influential in connecting otherwise remote regions of a graph, even though these nodes might have low influence as measured by other centrality measures. These nodes form a bridge between parts of the graph and thus play a key role in reducing path distances when traversing the graph. An example of this might be a jazz musician of relatively limited stature who managed to perform with both Duke Ellington in the 1940s and Wynton Marsalis in the 1990s, thus forming a bridge between musicians of different eras.

Here is a simple illustration of betweenness centrality, again using the Node D:

In this case, D has just two first degree connections yet it plays a pivotal role in the network structure by being the bridge between the BAF and CEG clusters, which otherwise would be unable to connect.

Another critical centrality measure is eigenvector centrality, where the influence of a particular node is defined by the connectedness of its closest neighbors. This can be thought of as the who you know type of centrality, wherein an individual node might not be thought of as important on its own, but its relationship to other highly connected nodes indicates a high level of influence. In our modern society, you might view this type of node as being a confidant of a popular celebrity, athlete, actress, and so on (perhaps a gatekeeper for some highly influential neighbors).
Let's examine a basic illustration of eigenvector centrality at work:

![Eigenvector centrality](image)

In this instance, D has only two first degree connections, but is surrounded by a host of very well connected nodes. Node D would thus score highly on the eigenvector centrality measure due to the relative importance of its first and second degree neighbors.

Finally, we consider the degree centrality measure mentioned earlier, which examines the number (or proportion) of other nodes linked to a specific node, either through inbound, outbound, or undirected connections. This type of node might act as a sort of hub for information flow—it might not be the source of direct information, but plays a critical role in dispersing this information to others.

The following example illustrates a node with a high degree centrality (once again using Node D):

![Degree centrality](image)

In this case, D has direct connections to five of the six remaining nodes, while no other node has more than three such edges, making D a hub within the network. Based on this structure, information is likely to flow though D, particularly as nodes communicate across the network (say from B to G).
To summarize, centrality is an essential measure for how information flows within a network, and should be assessed using a combination of the above measures to achieve a complete understanding of the network. We will go into greater detail on this critical topic in Chapter 4, Network Patterns, where a number of more detailed visual examples will be presented, so that we can begin to apply these concepts to real-world graphs.

Components
Graphs can be termed as connected, where all nodes are joined through a fully linked network, and disconnected, in which there are separate groups of nodes with no relationship between the groups. It is in this latter instance where components take root with multiple groups of nodes standing alone with no linkages to other portions of the graph. Let’s look at a hypothetical group of friends’ network, first in a connected state, and then in a disconnected state with two components.

First, in the connected state, (as shown in the following figure) all member nodes can reach one another, with a maximum path distance of four Nodes (from G to both C and E). This is a rather loose set of connections yet it remains fully connected:

Next, let us suppose that Node F has done something to alienate both nodes A and D, forcing them to break off their connection to F. Now not only is F cut off from the remainder of the network, but so is G, who was previously connected through F. We now have two distinct components to the graph, as shown here:
Many friends' networks exhibit this type of behavior, albeit on a larger scale than we have shown here. One of the more notable examples in the literature comes from Berman, Moody, and Stovel published in 2004 (http://www.soc.duke.edu/~jmoody77/chains.pdf) and shows the romantic connections in a selected high school, with a graph composed of nearly 20 distinct components.

Let's take a brief look at a specific type of component—the giant component.

**Giant components and clustering**

A giant component (http://en.wikipedia.org/wiki/Giant_component) might be thought of as the largest cluster in a network assuming it follows a specific mathematical formulation. For simplicity, we might refer to this as large components or largest connected components. These might also qualify as giant components, but will not require the same level of qualification. In our prior example where we saw a split in our friends' network, nodes A through E all remained connected within one cluster, while nodes F and G formed their own smaller cluster. In this case, the first cluster becomes the large component, by virtue of its larger size relative to the two node cluster represented by F and G. Now let's consider a case where we have more than two distinct clusters.

Take a look at the following network:

![A large component composed of nodes H-R](image)

We started with the same two clusters from our disconnected network example, but added nodes H through R, which have formed a new cluster that is clearly the dominant grouping in the graph. This is now the giant component, as it encompasses 11 nodes, compared to the other clusters with five and two, respectively. In Chapter 4, Network Patterns, we will take a look at how giant components form using various connectivity thresholds and assumptions.
Even when graphs are connected within a single component and we no longer have multiple components, clustering still plays a critical role in helping us to understand relationships, information flow, the spread of disease, and other relevant topics. We can assess the degree of clustering through statistics such as the clustering coefficient, applied at both global and local (that is, neighborhood) levels. Many clustering applications will be covered in greater detail in Chapter 4, Network Patterns, and the statistical measures will be covered in Chapter 6, Graph Statistics.

Homophily

Homophily is one of the key concepts that we need to understand when we examine network graphs, and is critical in helping us to assess networks with significant clustering. The term refers to what is often characterized as "birds of a feather stick together", wherein individuals tend to congregate with other like-minded individuals to form tightly knit clusters. Homophily might be driven by gender, race, age, occupation, education level, social status, or some other salient characteristic possessed by individuals within the network. These attributes might act individually, but will often interact with one another to define subgroups within a graph. Here are a few simple examples of groups we might find within a network graph:

- Women with a postgraduate degree
- Electricians belonging to the same union
- Executives serving on overlapping boards of directors

I think you get the idea, and could no doubt come up with many other relevant combinations. Once our graph is created, we can test for homophily and begin to examine its causes by exploring the tightly knit clusters that characterize its presence. In Chapter 4, Network Patterns, we will learn more about the critical role of homophily as it relates to the spread of information and innovation, while some of the statistical measures covered in Chapter 6, Graph Statistics will help us understand the presence of homophily in a network.

Density

Graph density is a measure of how tightly interconnected a network is calculated by examining the proportion of edges relative to the possible number of connections. A network with a high degree of homophily will tend to have a low density (due to the lack of connections beyond the local clusters), while networks that show a high degree of interaction across the network will have higher density levels. This will depend to some degree on whether we are viewing the entire network or a more localized sample. Two networks with identical numbers of nodes might have very different density levels; even the same network measured at different time intervals is likely to have differing density measures as links are formed or broken over time. A more detailed exploration of this measurement will be included in Chapter 6, Graph Statistics.
Network behaviors

We have just completed a brief overview on network structure and how it helps us to explain the patterns we see within an already existing network. Now it is time to touch on some of the behaviors that can take place within those structures and how they might develop or fail to develop according to features of the graph.

There are two terms used almost interchangeably to explain the evolution of a process within a network—contagion and diffusion. We will provide a basic overview in this section and then explore this concept in much greater detail in Chapter 4, Network Patterns and Chapter 8, Dynamic Networks.

Contagion and diffusion

The concept of contagion is typically associated with the spread of disease, but it can be used to describe a variety of phenomena in the marketing and social spaces. At its essence, contagion refers to the ability of something—a disease, an idea, or a book—to spread rapidly based on its network structure. Given the typical verbal association of contagion and the spread of disease, I will use that term in cases where we examine the progression of a disease. In all other contexts, the book will default to the use of diffusion, as that seems to be the predominant term when we look at how ideas propagate through a network.

To understand the potential of diffusion, we need to comprehend both, the structure of the network and the influence of various actors (nodes) within the network. If our subject (idea, book, information, and so on) flows through highly influential nodes within the network, we would then anticipate a rapid spread throughout the graph. To the contrary, if an idea is launched from a distant, poorly connected node, it is highly likely that there will be a very low level of diffusion, and only a small portion of the network will be exposed to the idea.

The following figure shows a hypothetical diffusion of a new product through a small network, using the simple assumption that each node requires 33 percent of its neighbors to adopt the product before it tries the product. Note that whenever this criterion is not met, the diffusion will cease in that part of the network. This figure shows the first four rounds of diffusion, beginning with the purchase of the new product by Node A:
In addition to the influence factor just mentioned, the basic structure of a network will help to determine the rapidity with which information spreads. Networks with lots of localized clusters tend to limit or even terminate the flow of information, while densely connected networks with few gaps promote the transmission of information. As noted earlier, we will explore these concepts in greater detail in Chapter 4, Network Patterns and Chapter 8, Dynamic Networks, with visual examples that illustrate information flows through the existing networks.

**Network growth**

Much of what we have considered until this point has been by looking at an existing set of networks without regard for additional growth or expansion of the network. In reality, we know that networks do grow, often very quickly and unpredictably. There exist a number of models in the literature that attempt to predict network growth using a variety of assumptions. These include the Erdos-Renyi random growth model as well as the Barabási–Albert model of preferential attachment, plus a host of variations on these themes.

Barabási offers a concise definition of a random network model (http://barabasilab.neu.edu/networksciencebook/downlPDF.html#):

"A random network consists of N labeled nodes where each node pair is connected with the same probability p."

- Barabási, L. (2012), Network Science
Given this definition, we can quickly determine that random graphs could take on a variety of structures, especially as we increase or decrease the $p$ value. If $p$ is high, say 0.8, then our graph will tend toward a dense structure, while a $p$ value of 0.2 would lead to a very sparse graph with few connections between nodes. Random networks have been criticized for being unrealistic in modeling network growth (refer to Social and Economic Networks (2010), P. 78, Matthew Jackson, which can be retrieved from http://books.google.com), yet they continue to serve as a useful starting point in understanding how networks evolve.

Preferential attachment, on the other hand, has been shown to correlate much more closely, if imperfectly, to real-world situations, such as the structure of connections on the Web. Barabási notes the basic premise behind preferential attachment. The following is retrieved from http://barabasilab.neu.edu/networksciencebook/downlPDF.html#:

"Preferential attachment is a probabilistic rule: a new node is free to connect to any node in the network, whether it is a hub or has a single link... if a new node has a choice between a degree-two and a degree-four node, it is twice as likely that it connects to the degree-four node."

– Barabási, L. (2012), Network Science

In the preferential attachment model, new nodes are more likely to connect to nodes that have higher degrees, a pattern often referred to as the rich get richer. Thus, our network winds up with a number of hubs with many connections, surrounded by a greater number of nodes with fewer edges. Many real-life examples have shown this sort of behavior, from mathematical processes to citation networks and on to the Web.

We will defer further discussion on this subject until Chapter 4, Network Patterns and especially in Chapter 8, Dynamic Networks, where we will discuss the potential ways a network might grow, how quickly it will evolve, and what it might look like at some future point in time. Gephi provides the capability to perform this type of analysis, which can lead to powerful illustrations for how a network has changed over time.

**Overviewing Gephi**

Before moving into a discussion of the components of the Gephi interface, it will be helpful to provide you with a more holistic view for you to understand Gephi. Many of you might already be quite familiar with the general philosophy behind Gephi and how it is laid out. If so, feel free to skip ahead. For everyone else, let's step back for a moment and take a view at the big picture. You might think of this as viewing the entire building before visiting each of the individual rooms.
If we carry our building analogy a step further, Gephi has three primary sections, surrounded by a host of smaller rooms. The three main sections are as follows:

- **The data laboratory**: This houses all of our original data, plus additional calculated values created when we apply statistical or partitioning methods.
- **The overview window**: Most of the actions here will take place as we test layouts, set filters, and perform many more operations on our network.
- **The preview window**: This is where graph window output is refined, typically starting with the original graph and then using an array of tools to add aesthetic appeal. This is also where we can choose to export the graph to a new output format, such as a PDF or SVG file.

Beyond these three main sections lie a host of tabs, or smaller rooms, that allow us to perform many functions that will primarily be carried out inside the graph window. You might wish to rearrange these tabs to suit your work style, but I find the default setup quite intuitive and easy to work in.

So now that we've taken a very simple look at how Gephi is laid out, let's examine each of the primary and secondary windows in slightly greater detail.

**Primary windows**

The three main operating windows discussed earlier are covered in the following sections. While there will be some details provided within each of these sections, this book will not provide a comprehensive guide to the functionality of each and every option. Additional information is available via the Gephi documentation and forums, as well as through my introductory book on Gephi—*Network Analysis and Visualization with Gephi*, Packt Publishing.

Let's begin with the data laboratory, which will be the repository for our graph data.

**Data laboratory**

All data that feeds our network graphs will reside in the data laboratory. The laboratory is built around the concepts of nodes and edges, which we covered extensively earlier in this chapter. While the data laboratory might have a spreadsheet-like appearance, do not confuse it with the likes of Excel, Calc, or Google Spreadsheet. Certain aspects of data manipulation can be done here, but it is best to have your base data largely prepared prior to importing it into Gephi. For instance, I find it much easier to utilize a spreadsheet tool when there is a need to create distinct values within a categorical field. Likewise, any field values that are based on a specific sorting scheme might be best created outside of Gephi.
This is not to say that data held within the laboratory is fully static. For example, all statistical and clustering calculations will automatically append new values to each node when a process is run. You can also add columns, copy data from one column into another, and so on. Still, making individual node or edge-level changes here can become tedious (and very time consuming), especially if your dataset consists of hundreds or thousands of values.

There are several ways we can add data to the laboratory, from the very basic to more complex (albeit, more powerful). Here are a few ways:

- Manual entry
- CSV import
- Excel import
- MySQL import
- Graph file imports

Let’s briefly discuss each of these options. I will not attempt to go into each and every use case, as that alone could fill an entire book. Instead, we’ll look at some generic examples, and I recommend you the Gephi forums for cases that are beyond the scope of this book. Example processes for each of these processes will be provided in the Appendix, Data Sources and Other Web Resources.

**Manual entry**

If you are working with a very small dataset, or are very skilled at data entry, there is a manual option to create a Gephi dataset. This approach can be useful for those who wish to experiment; this is discouraged for all, but the smallest networks. Importing data from a .csv format is so easy that it makes little sense to choose the manual option beyond the simplest of scenarios.

**CSV import**

One of the simplest ways to move data into Gephi is through the use of comma separated values (.csv) files. Users can start saving and exporting data in Excel, Calc, Google Spreadsheet, or any application that allows the files to be saved and exported in the .csv format. To make the data transfer even simpler, only an edge file is actually needed by Gephi, as it will create a node file automatically. However, if you wish to add more detail to describe your nodes, I recommend that you create separate node and edge worksheets.
Excel import
Excel users have the ability to easily load data into Gephi using the Excel/csv converter to network plugin from the Gephi Marketplace. This plugin uses a more flexible approach as compared to the data laboratory import spreadsheet process. More information on this approach can be found at the Gephi Marketplace at https://marketplace.gephi.org/.

MySQL import
Gephi users that have data housed in the open source MySQL database are also able to directly import data by creating specific tables for nodes and edges, and then pointing Gephi to the database using connection parameters.

Graph file import
Gephi also enables the use of multiple graph file formats, making it very easy for the users of other software (UCINET, Pajek, GUESS, and so on) to import existing files into Gephi.

As a final note, we also have the ability to merge files in Gephi using either the data laboratory or simply through opening a second (or greater) file.

Graph window
All visual output is initially viewed using the graph window, with Gephi providing a somewhat crude initial view of your network. The initial view is very simple given that we have not selected any sort of layout at this stage; this is an issue that will soon be rectified. It is highly likely that the majority of your time will be spent working within the graph window, observing the patterns within your network. All applications of filtering, partitioning, sizing, coloring, and any layout adjustments will be seen here first, so it is wise to become very familiar with this space, if you haven’t already done so.

You will observe that the graph window is adjacent to multiple toolbars, each with an array of functions. The functionality behind each of these options is generally intuitive, and should be explored for further understanding. This book will not spend considerable time with each of these functions. For a primer on these, my introductory book on Gephi provides greater detail, or, alternatively, takes some time to play with each option and represents what happens to your graph.
Preview window

The Gephi preview window allows the user to adjust a variety of graph attributes that have been created in the original graph window. Here, we can customize node labels by adjusting font size, font color, outlines, specifying whether to use boxes for the labels, and electing whether to display the labels at all. These decisions can be made based on the density and complexity of our graph; dense graphs might benefit from labeling only the critical nodes. Using Inkscape or Adobe Illustrator to create labels after exporting the graph is another option that allows the greater customization.

Node appearance is also addressed by providing border width, border color, and opacity options. As with the node labels, you can elect to use external tools to provide a greater degree of customization, where you have the ability to color individual nodes and edges rather than applying a one size fits all approach. Remember that you can always toggle to the overview window to do many of these customizations in Gephi, and then simply refresh the preview window.

Additional options are provided for adjusting the appearance of graph edges. Edge thickness, color, opacity, radius, and curved edges are all available options. Likewise, edge arrows (for directed graphs) and edge labels are customizable.

The preview window is also where some of Gephi’s built-in export options reside, specifically, in the SVG, PDF, and PNG formats. Let’s briefly consider each of these options, and why you might select one over another:

- **PNG**: This represents the simplest choice. It creates an image of your network, making it easy to share it online or elsewhere, provided you have no desire to further enhance the graph. This option is ideal for sharing a quick snapshot of your work on the Web or via e-mail, but is obviously limited from an editing standpoint.
- **SVG**: The SVG export creates a scalable vector graphic that can be edited in other programs such as Inkscape, although the large file size of this format might be most suitable for graphs without a high degree of complexity.
- **PDF**: The PDF export offers some of the advantages of SVG minus the large footprint. This format is also editable in Inkscape and Illustrator, and will ultimately allow you to customize every aspect of your graph, as well as to add titles or other notations describing the graph.
Secondary windows – tabs
As we noted earlier, Gephi provides an array of secondary tabs that surround the main workspace, permitting the user to execute actions on the graph without the need to toggle between multiple windows. With this approach, the impact of filters, partitions, color and size adjustments, and much more can be seen instantly, making it easy for the user to take an iterative approach to manage and analyze the graph.

The next several sections will provide an overview of how to use each of these tabs, without going into greater detail at this point. Some of these options, such as filtering and statistics, will be covered in much greater detail later in the book, while further information on other functions can always be found on the Gephi wiki or in the user forums.

The filtering tab
The filtering tab is where we will eventually examine our graph output using a range of criteria, so that we gain a better understanding of our network. Many times, our network will be very large, dense, and thus difficult to navigate in its entirety. In these cases, filters provide us with the necessary tools to begin probing the graph systematically, searching for specific attributes or graph features. Note that Gephi gives us the ability to create individual or compound filters, where multiple conditions are nested.

The application of filters in Gephi is not always easy, so we will devote an entire chapter (Chapter 5, Working with Filters) of the book to them, as they can and should become a very powerful component of our Gephi skillset.

The statistics tab
Not all graphs are created with the end goal of analysis or measurement, but for those who wish to understand network interactions and patterns, the statistics tab can provide a wealth of information. Gephi provides an array of statistical graph measures that can be employed to better understand the structure of a network, and ultimately can be used to compare networks to one another.
Chief among these statistics are a variety of centrality measures to be applied at the node level, including betweenness centrality, eigenvector centrality, closeness centrality, and eccentricity. Other measures include graph diameter, clustering coefficients, edge betweenness, and average degree measures. Many of these are included with the base Gephi installation, while others are available through selected plugins. Chapter 6, Graph Statistics, will be devoted to discuss these statistics using individual graph examples to drive a greater understanding on how to measure the network, and what the numbers mean.

The layouts tab

The selection of an appropriate layout can make the difference between creating an impenetrable graph that fails to communicate a story versus an easily accessible visualization that not only communicates, but also has aesthetic appeal. In perusing the network graph literature, one is bound to come across the term hairball, a description for a very dense network with many connections that is all but undecipherable using standard graph algorithms.

By using Gephi, we have the ability to test many layout algorithms before settling on a final choice. This provides us with an opportunity to not only avoid the hairball issue, but also to find a layout that is most complimentary to the underlying network. Many of the layout algorithms provide options that allow the user to determine the ideal spacing within a graph by tinkering with attraction, repulsion, gravity, and other available settings.

We can also determine whether to employ a force-directed graph that displays a network based on the aforementioned attraction and repulsion settings, or to select a predetermined layout that arranges the network in a circle or set of concentric circles ordered by some sort of categorical ranking. In other words, Gephi makes it possible to explore network data using a wide variety of layouts, making it possible for us, the users, to select the best possible option for our graph.

We will explore many of these layout options in greater detail in Chapter 4, Network Patterns, by comparing and contrasting outputs using a variety of methods.

Essential plugins

Plugins used in Gephi do not abide by the primary and secondary workspaces model that we just covered. Instead, they logically wind up where they are designed to be used; layout plugins are placed in the layout tab, formatting plugins can be found at the perimeter of the graph window, and so on.
The basic idea with Gephi plugins, as with plugins for other software, is to add features that are not readily available in the core software. In some cases, this will be in the form of functions that help users to better format their graphs, while in other cases the plugins represent full fledged layout algorithms or graph generators that provide users with additional choices for graph creation and analysis.

There are a number of Gephi plugins which we will use later in this text, so it might be the best to download and install them early on so that you can follow along with some of the examples. While the Gephi Marketplace plays host to some excellent plugins that extend the core Gephi functionality, the number is very manageable. If you wish, download and install them all as the installation process is very simple, and the space requirement is minimal.

Here are some of the most essential plugins I refer to, and often use within the course of this book, along with a brief description of their category and functionality. If you need more detailed information, please navigate to the Gephi Marketplace site to learn more. Many of these plugins will be used within the book, so it might be a good idea to download them all up front so that you will have a fully capable Gephi installation to work with as you follow examples in the book.

I'm going to walk you through these plugins by category, providing brief descriptions of each. We will go into greater detail as each plugin is used within subsequent chapters.

**Clustering – Chinese Whispers**

Gephi provides several options for partitioning and/or clustering graph data, including this useful plugin. The goal of this clustering approach is to partition your network data into individual clusters, which can then be used to color or size the graph nodes, creating a more intuitive and easily interpreted visualization. While it is possible to color nodes in Gephi manually or through partitioning, the Chinese Whispers clustering provides another option that is based on an analysis of network patterns.

**Data laboratory**

The data laboratory is where all the data manipulation takes place. While the base installation provides several helpful functions, others can be added using plugins.
Data laboratory helper
For users frustrated with the limited ability to edit data in Gephi, this plugin provides the ability to recast the column type (from string to numeric, for example) as well as to create new columns based on existing values.

Exports
The ability to export data and graphs from Gephi to other formats is highly useful, as it makes Gephi a very flexible tool for further interaction with or deployment of network data. We'll provide a brief overview of a few plugins that can be used to display network graphs beyond Gephi.

Sigma.js Exporter
One of the best ways to make your network graph even more powerful is to deploy it on the Web and make it interactive using Sigma.js, with the zooming, grouping, and filtering capabilities. This is an easy to use tool that provides a template approach to publish interactive graphs, making it especially easy to replicate a series of graphs using a consistent approach. Once the graph has been exported to Sigma, additional customization is possible using the CSS, JavaScript, and HTML methods. We will learn more about Sigma.js and explore actual examples in Chapter 9, Taking Your Graph Beyond Gephi.

Seadragon Web Export
Another option for deploying a graph to the Web is through Seadragon, which permits graph users to zoom in and out of your graph, which can be especially useful in the case of large or very dense networks. While this option does not provide the full range of capabilities found with Sigma.js, it does provide a quick solution to make your graph accessible through the Web.

Graph Streaming
One of the most powerful aspects of network analysis is its ability to see how a network evolves over time, rather than viewing a static graph. There are a couple of ways we might approach this, the Graph Streaming plugin providing perhaps the most powerful approach. All that is required to use this tool is a JSON dataset with time elements.
ExportToEarth

Users with geography-based datasets are able to use Gephi to create their initial graph before exporting the network in the .kmz format used by Google Earth and other GIS programs. All that is required to leverage this tool is some geocoded information in your data file, such as latitude and longitude data.

Generator – the Complex Generators plugin

A wide range of fundamental network graph types can be generated using this tool, including Erdos-Renyi (random graphs), Barabási-Albert (preferential attachment), and Watts-Strogatz (small world) graphs. These generators help to provide a quick visual understanding of several classic network growth theories, and can ultimately help us to comprehend network behavior while viewing existing graphs.

Here are three simple examples created in Gephi using the Random Graph, Barabási-Albert scale-free model, and the Watts-Strogatz small world model Alpha generators, all using a 20 node specification.

First, let's take a look at the random graph example, which is as follows:
Next, a scale-free model is as follows:

![Barabási-Albert scale-free model](image)

Lastly, a small world graph is as shown here:
Note the dramatic differences in network structures between the three models, all based on underlying assumptions of network growth. As mentioned earlier, the generators are very useful to understand and visualize network structures using different assumptions, which can then provide insight when we create graphs from our real-world datasets.
Layout

One of the most powerful ways to extend Gephi is through the use of a wide array of layout algorithms available through plugins. These layouts, when paired with the multiple layout options already available in base Gephi, will provide you with a wealth of choices to map your networks. Some of these choices will be useful for very specific use cases, while others are much more generally used for a variety of networks. Let's take a quick look at a handful of highly useful layout algorithms, and the situations where we might find them most appropriate.

The Multipartite layout

A multipartite graph is a network with multiple nodes (vertices) that belong to different groups, where all edges are between members of different partitions, and no edges can be found between members of the same partition. One can think of this in terms of members of a category (a sports team, for instance) that are connected to the top level of the category, but not to one another. If the team and team members represent the only two partitions, we have a bipartite graph; but we could also have many cases with more than two partitions. This becomes especially useful in cases where we have a temporal network, where players are associated with Team A initially, but are later traded to Team B or Team C.

The primary purpose of the multipartite layout in Gephi is to minimize edge crossings, thus making it easier to view and interpret the graph.

The Hiveplot layout

A Hiveplot is a graph layout that attempts to overcome the so-called hairball effect produced by large, highly connected networks. The hiveplot addresses this by placing nodes along multiple radial axes based on network structure. This approach is particularly appealing in cases where there are three or more definable levels, as it will position nodes in an effort to avoid some of the unintended or misleading effects that might appear while using other algorithms. We'll examine this approach further in Chapter 4, Network Patterns.
The Concentric layout
Networks are often most easily viewed using familiar visual forms such as circles. Concentric layouts allow us to take advantage of this, particularly while working with small to medium datasets. Nodes are arranged in a series of concentric circles based on the distance from our central node. Thus, nodes with direct connections are arranged in the first circle followed by nodes that are at a distance of two nodes away from the center, and so on. By arranging nodes in this concentric fashion, viewers are able to more easily navigate small network structures and see the closeness of relationships to a single node, and to each other.

The OpenOrd layout
The OpenOrd plugin helps to generate network graphs very rapidly, and is best suited to very large networks due to a loss of precision in the interest of greater speed. This approach is based on the classic, but much slower, Fruchterman-Reingold algorithm provided with Gephi. In cases where you are dealing with hundreds of thousands of rows of data, this algorithm enables a rapid look at the network structure.

The Circular layout
The circular layout plugin actually provides three distinct layout types—the circular layout, dual circle layout, and radial axis layout. A variety of options allow users to order nodes by degree, ID, attribute sort, or randomly. This can be especially useful to arrange a network based on predefined characteristics, as opposed to calculated relationships within the network.

The Layered layout
The layered layout is a useful layout for cases where we wish to visualize a small world phenomenon using numerical values to assign the layer, or orbit, that each node resides in. Stronger relationships to a key node will occupy inner orbits, with more distant connections occupying the perimeter of the graph. This approach is similar to the one used by the concentric layout.
The ARF layout

The Attractive and Repulsive forces (ARF) layout provides a useful layout tool that affords considerable flexibility through attraction and repulsion settings. ARF outputs tend toward a more circular appearance than many of the other spring-based algorithms such as the Fruchterman-Reingold and the Force Atlas models.

Additional plugins

A number of additional plugins are provided for Gephi, with new ones being added on a regular basis. Here are a couple tools that provide even more utility as you create and analyze your network graphs.

Link Communities – metrics

Link Communities is a clustering approach that assesses links in undirected and unweighted networks and then classifies nodes into communities based on their similarity. Nodes can be placed into multiple communities, making this approach differ from other clustering approaches. Once the metric has been computed, users can then select a layout algorithm of their choice to display the network.

Give color to nodes – tools

One of the most effective methods to convey network information is through the use of color. Gephi provides the ability to color individual nodes within the graph window, but this useful plugin lets you provide colors within your dataset that can be used to color the entire graph, versus making ad hoc changes using the Gephi toolbars.
Summary
As you might have gathered from this introductory chapter, there are an almost infinite number of graphs that can be created by pairing network data with a wide variety of algorithms provided either in base Gephi or through one of the many available plugins. Whether your end goal is simply to create a compelling visual image that communicates a specific story or you intend to perform some thorough network analysis using filters, graph statistics, and other tools, Gephi provides a robust framework for your explorations.

Before moving on, you might wish to learn more about the specific plugins—most of them provide some level of detail about their purpose and methodology. There are additional plugins not covered here which you might also find highly useful to create your own graphs.

In the next chapter, we’ll walk through a process that will help you to scope your graph needs, and then provide an overview on how to create or access a dataset and import it into Gephi. Finally, we’ll provide a brief overview on how to manage your data in the Gephi data laboratory, before moving on to creating and testing some actual graphs in Chapter 3, Selecting the Layout.
Where to buy this book

You can buy Mastering Gephi Network Visualization from the Packt Publishing website. Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers. Click here for ordering and shipping details.