A Geometry for Network Visualization in `ggplot2`

Sam Tyner and Heike Hofmann

Abstract will be here

1. INTRODUCTION

At its core, a network is simply a set of points connected in pairs by a set of lines (Newman, 2010). Here, we refer to the lines as edges and the points as vertices, although these are also called nodes. These two seemingly simple sets of graphical objects, points and segments, are used to encode a huge variety and quantity of information across many fields of study. For instance, networks of scientific collaboration, a food web of marine animals, and American college football games are all covered in a paper on community detection in networks by Girvan and Newman (2002). Additionally, Buldyrev et al. (2010) examine node failure in interdependent networks like power grids. Social networks, such as links between actors found on `www.imdb.com`, and neural networks, like the completely mapped neural network of the *C. elegans* worm are also etensively studied (Watts and Strogatz, 1998). Networks vary widely in scope and complexity: the smallest network is simply an edge between two vertices, while one of the most commonly used and most complex networks, the world wide web, has billions of vertices (webpages) and billions of edges (hyperlinks) connecting them. The edges in a network can be directed or undirected: directed edges represent information travelling from one vertex to another, and switching the direction would change the structure of the network. The world wide web is an example of a directed network because one webpage may link to another, and not necessarily the other way around. Undirected edges, however, are simply connections between vertices. Coauthorship networks that encode information about academic publications are examples of undirected networks because if two people author a paper together, that creates a connection between them that is bidirectional.

A social network is a network that everyone is a part of in one way or another. We do not necessarily refer here to social media like Facebook or LinkedIn, but rather to the connections we form with other people. To demonstrate the functionality of our geometry for plotting networks, we have chosen an example of a social network from the popular television show Mad Men. This network was compiled in Chang (2013). In this example

network, there are 52 vertices and 87 edges. Each vertex is a character on the show, and there is an edge between every two characters who have had a romantic relationship.

This network is shown in figure 1.

In the plot, we can see one central character who has many more relationships than any other character. This vertex represents the main character of the show, Don Draper.

There are many kinds of networks, and networks are extensively studied across many disciplines. Many sociologists study social networks, and many biologists study protein networks. As different as these and the many other disciplines that study networks are, they all need the ability to quickly and effectively visualize networks. We found the current tools to be lacking in this ability, so we chose to fill this gap by adding network plotting capabilities to the popular and widely used `R` package `ggplot2`. From January 1, 2015 to March 21, 2015, `ggplot2` was downloaded over 270,000 times, or approximately 3,454 downloads per day. It has also been downloaded in 189 countries at least once, and in 31 of those countries, including China, Israel, and Colombia, it has been downloaded over 1,000 times.

The three necessary elements of any network visualization are the vertices, the edges, and the layout. But once those three items are visualized, we usually find our visualization to be lacking. We may want to color the vertices (or edges) by some sort of grouping variable, or we may want to make vertices of degree ten twice the size of vertices of degree five. Many `R` packages already exist for network analysis and visualization such as `igraph` by Csardi and Nepusz (2006), `sna` by Butts (2014), and `network` by Butts (2008); Butts et al. (2014) but we have found these packages to have unintuitive or burdensome methods for customizing the colors, sizes, etc of the vertices and edges of the network. For instance, the `igraph` package allows for coloring vertices by groups but the user must assign the colors to each vertex individually as opposed to assigning color by a grouping factor variable.

The `GGally` package by Schloerke et al. (2014) contains a very useful function written by François Briatte and Moritz Marbach called `ggnet` that does allow for fairly straightforward customization of these three necessary graph attributes. Coloring the vertices or edges in a graph is a quick and easy way to visualize grouping and can help with pattern or cluster detection. The vertices in a network and the edges between them compose the structure of a network, and being able to discover patterns among them visually is a key part of network analysis. Viewing multiple layouts of the same network can also help reveal patterns or clusters that would not be discovered when only viewing one layout or analyzing only an adjacency matrix. The `ggnet` function, however, requires the graph input to be a `network` object according to the `network` package. Our work builds off of the `ggnet` function and presents an intuitive `geom` for network visualization within the `ggplot2` framework, without the need for objects other than simple data frames.

1.1 Data Structure

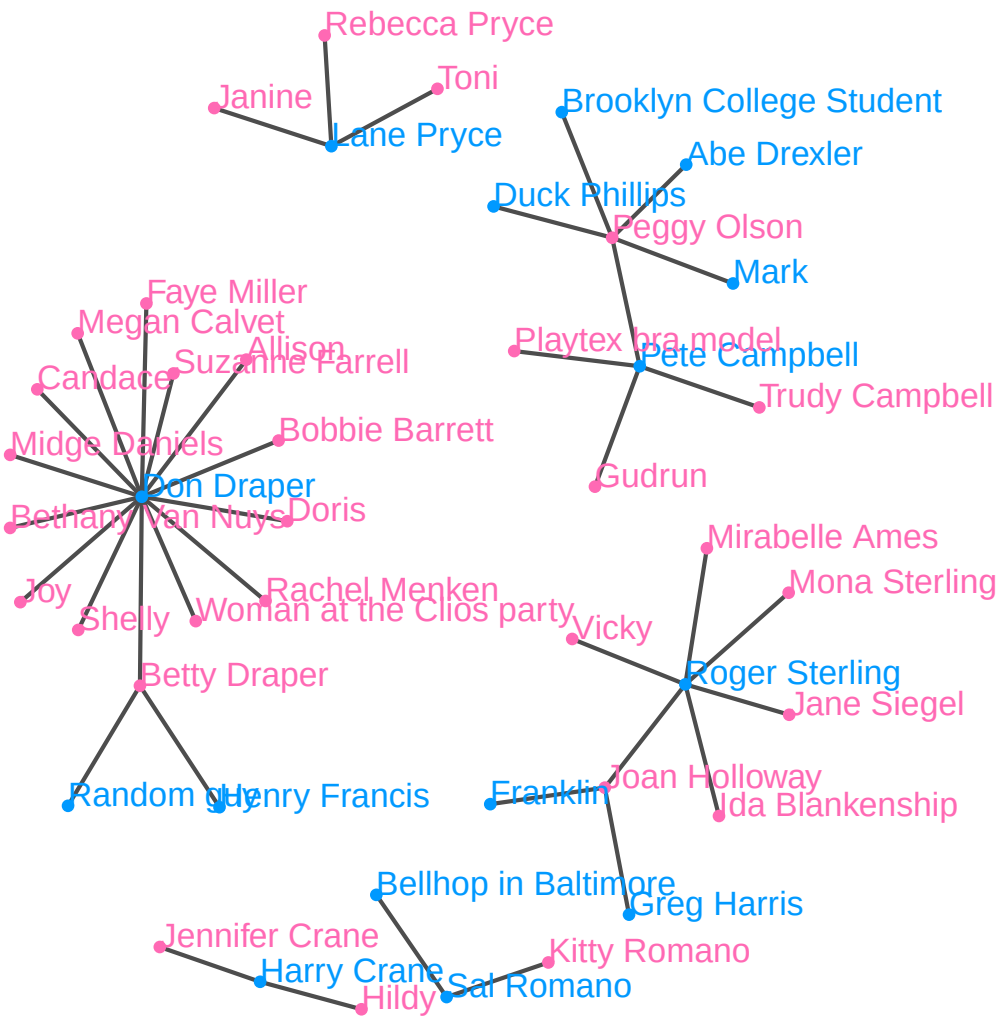

Figure 1. Graph of the characters in the show Mad Men who are linked by a romantic relationship.

XXX we need a link between the two data frames: we have from_id and to_id in the edge data frame, but we also need an 'id' variable in the vertex data. We don't officially have that at the moment - in the code there is a soft requirement (new as of Mar 20) that one of the columns in the edge data set has to be called 'label' ... that's a bit stoopid, but I couldn't think of a better name. It would make more sense to call this id or edgeid or something along those lines - let's talk about it, and make it a formal requirement. In order for this geometry to work using data frames, there need to be two separate data frames given to the `geom_net` function: one for the edge information and one for the vertex information. The vertex data frame should contain all the relevant vertex information. *The only necessary variables are the vertex names/labels and the vertex degree (if undirected) or the vertex in degree and out degree (if directed). The names and degrees are stored in columns and the rows are the vertices. Other values of interest, such as grouping variables, for each vertex should be stored as columns in this dataframe, with an observation for each vertex.*

The edge data frame should contain all the relevant edge information. The only necessary variables are the "from vertex" and "to vertex" for each edge in the network. The From and To vertices should be listed according to the names of the vertices in the vertex information data frame. Other variables may also be included for each edge, such as the edge weight or grouping variable. As before, the variables of interest are columns in the data frame and the rows are each edge in the network.

1.2 Vertex Aesthetics

In our geometry, the possible vertex aesthetics mimic the usual aesthetics in `ggplot2` for points. The `vlabel` aesthetic is a logical value that prints the name or number assigned to each vertex in order to uniquely identify it on the plot. If this value is set to TRUE, the vertex data frame is required to have a column titled label. This aesthetic is most useful for smaller network objects where all vertex names can be printed on or near their corresponding vertices and still be read clearly. The `vcolor` aesthetic can be an identity color (e.g. `color = "red"`) to change the color of all vertices, XXX shouldn't we take this out? or it can take a factor variable which identifies each vertex as belonging to one of several groups, and it colors the vertices of the graph according to this grouping. Right now, the `vcolor` aesthetic does not work like a typical color aesthetic in `ggplot2`. It can, however, take HEX color characters and assign them to different levels of a factor variable. XXX We should probably take this out too. It can also color the vertices according to the different values of some numeric variable, such as degree. The colors are set to the default color palette in `ggplot2`, and the palette used can be changed in the usual way within the `ggplot2` framework, by using `scale_color_brewer`. The `vsize` aesthetic can also be set to an identity value to increase or decrease the size of all vertices in the graph, or it can be set to the degree of the vertices in undirected graphs or the in-degree (or out-degree) of directed graphs. These must be integer-valued columns in the vertex data frame. It can

also be set to any other numeric variable associated with the vertices in the network. The `vshape` aesthetic is for changing the shape of the vertices from the default circle to other shapes, like square or triangle. It can also change the shape of the vertices according to some grouping variable, which is currently required to consist of integer values from 0 to 25.. Finally, the `valpha` aesthetic will change the vertex transparency to a set value, like 1/10, or to some numeric variable of interest that takes on the values $[0, 1]$.

Evidently, we created all of these aesthetics to fit in exactly to the `ggplot2` grammar of graphics. We also created the edge aesthetics in a similar fashion.

1.3 Edge Aesthetics

Again, our edge aesthetics mimic the familiar `ggplot2` aesthetics, this time for lines. XXX take out this??? The `elabel` aesthetic is the name or number assigned to each edge. This aesthetic is most useful for visualizing random graphs or Markov processes, where each edge probability is of interest. The `ecolor` aesthetic changes the color of all of the edges according to the identity function or can change the color according to HEX colors assigned to levels of a grouping variable. The `elinetype` aesthetic can also be used with a grouping variable to change the line type of each edge in the graph from solid to one of the different types of lines in `ggplot2`. The column associated with this aesthetic must be integer valued and take on values from one to six. The `esize` aesthetic can also be set to the weight or probability of each edge, or it can be changed for all edges with the identity function. The associated column can take on any numerical values. The `ealpha` aesthetic can change the edge transparency to a set value, or to some numeric variable of interest that takes on the values $[0, 1]$, such as edge weight or probability. This property is especially useul for networks with a lot of connections and with vertices of relatively high degree. XXX should we take this out? Finally, the `directed` aesthetic is a logical value that identifies whether or not the network is directed. When this value is true, the arrows change size, color, etc. with the previous aesthetics.

1.4 Layout Argument

There are many layouts to choose from, but the default layout is the Kamada-Kawai layout. This is a force-directed layout for undirected networks (Kamada and Kawai, 1989). The `layout` argument in the `geom_net` function can be changed to any of the layouts also found in the `sna` package, such as circular, eigen, or Fruchterman-Reingold. Should I insert a table of layouts here? And I took out the thing about the background color because I figure it can just be changed with `theme()`

2. MANY MANY EXAMPLES

Format-wise, is it ok if each example is its own subsection? Also, can you check the references I added to the .bib file to make sure I'm creating them correctly? I've just been

appending them.
In this section, we demonstrate the capabilities of `geom_net` in a series of examples.

2.1 Blood Donation

Note: would be best with labels and direction
In this directed network, there are eight vertices and 27 edges. The vertices represent the eight different blood types in humans that are most important for donation: the ABO blood types A, B, AB, and O, combined with the RhD positive and negative types. Can I cite wikipedia here? Should I find a more academic source? The edges are directed: a person whose blood type is that of a *from* vertex can to donate blood to a person whose blood type is that of a corresponding *to* vertex. In the example below, loops are removed. Loops exist on every vertex in this example, as blood of matching ABO and RhD type can always be received.

```
ggplot(data = blood$edges, aes(from_id = from, to_id = to)) +
  geom_net(vertices = blood$vertices, layout='circle', vlabel = TRUE, vsize = I(2), directe
  expand_limits(x = c(0,1), y = c(0,1)) + theme_net #add direction later
```

This network is shown in figure 2.

2.2 Email Network

Note: would be best with coloring and direction This email network comes from the 2014 VAST Challenge (Cook et al., 2014). It is a directed network of emails between company employees with 55 vertices and 9,063 edges. Each vertex is an employee of the company, and each edge is an email sent from one employee to one or more other employees. The arrow of the directed edge points to the recipient(s) of the email. The network contains two business weeks of emails across the entire company.

This network is plotted in figure 3. This plot is very crowded, and it is difficult to distinguish structure in the network when it there is so much over-plotting. To remedy this, we facet the network by day to examine how the company's communications change throughout the week.

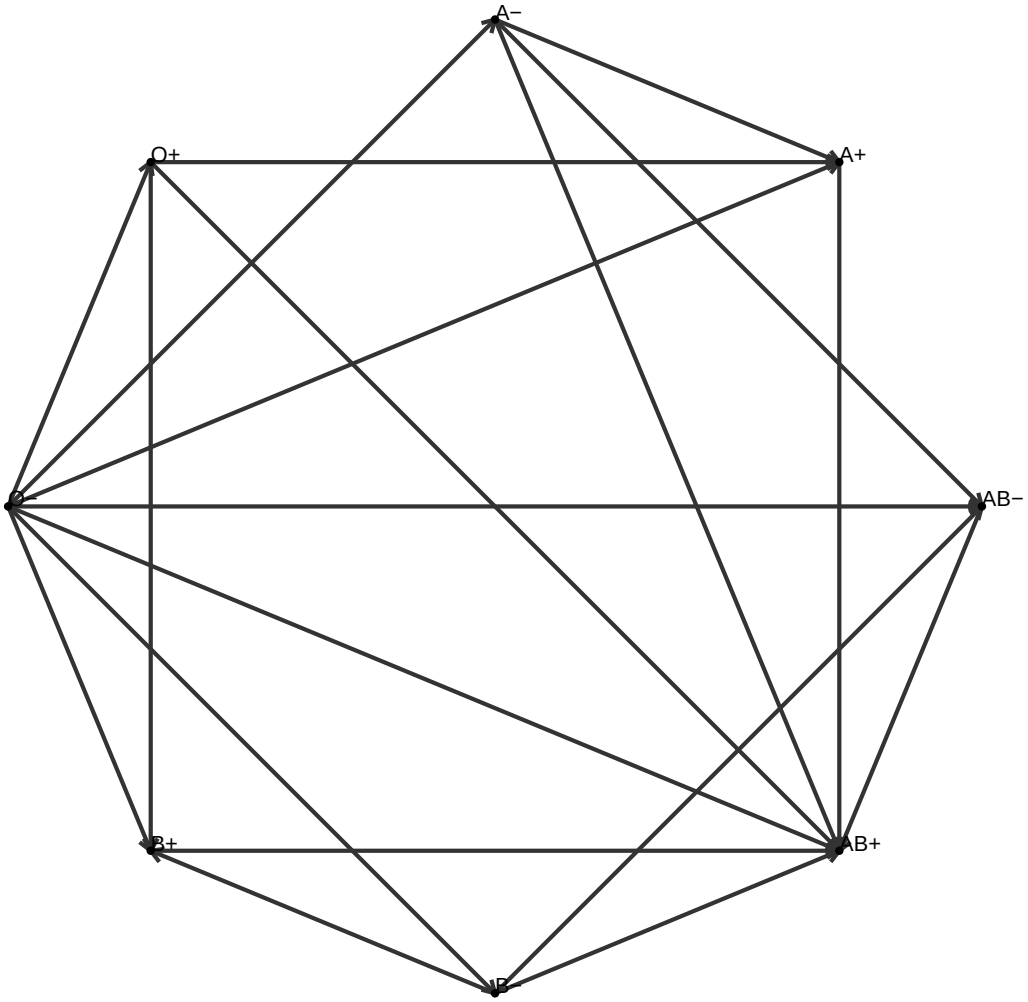

Figure 2. Network of blood donation possibilities in humans by ABO and RhD blood types.

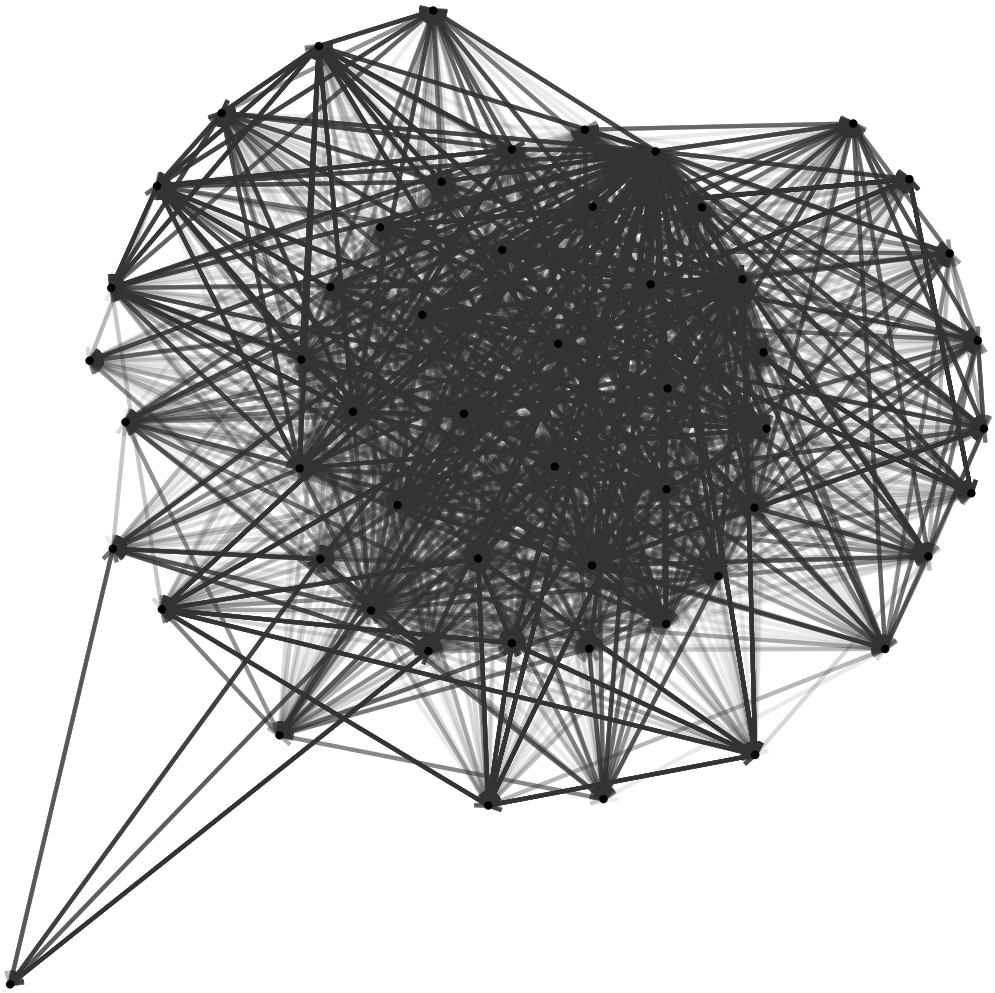

Figure 3. Email network within a company over a two week period.

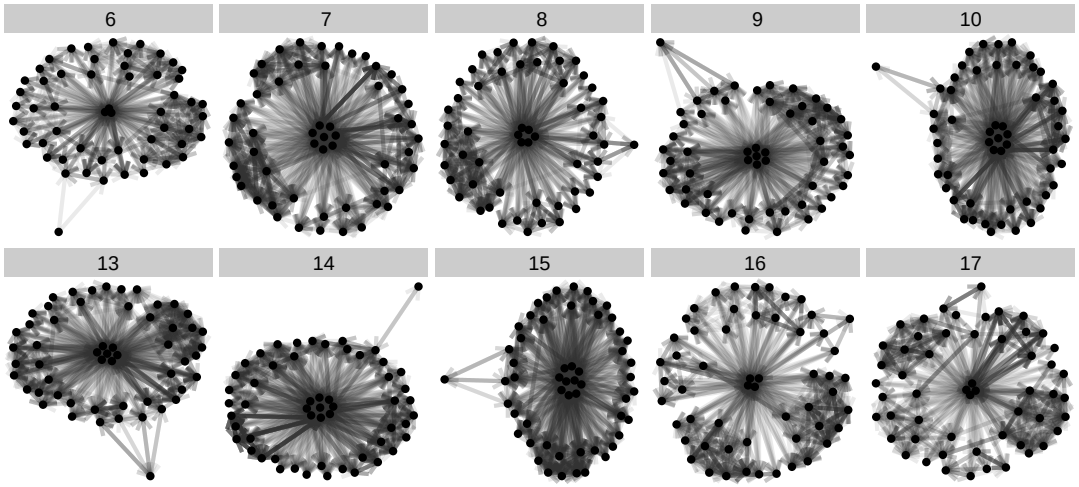

Figure 4. The same email network as in figure 3 facetted by day of the week.

This plot is shown in figure ??. XXX note for later: remove all the emails that go to everyone in the network. With the facetting, we can see that there are between four and seven employees sending or receiving? will know with direction option added later several emails each day to a majority of the company. We can also see clusters that have formed on the perimeter of each day's network where the employees are communicating heavily with each other, but not much outside of their cluster. We can also see that there is heavier inter-cluster communication on different days. For example, day 17 has more concentrated in-group communication than day 15. Additionally, we can better distinguish the communication of the outlying employee, who only corresponds with up to four other complany employees on a given day. talk about clusters being different departments within the company?

2.3 `ggplot2` Theme Elements

Note: would be best with labels and direction
This example comes from the `theme()` help page in the `ggplot2` documentation (Wickham, 2009). It is a directed network which shows the structure of the inheritance of theme options in the construction of a `ggplot2` plot. There are 53 vertices and 36 edges in this network. There is an arrow from one vertex to another if the element represented by the *to* vertex inherits its values from the *from* vertex. For example, the `axis.ticks.x` option inherits its value from the `axis.ticks` value, which inherits its value from the `line` option. Thus, setting the `line` option to a value such as `element_blank()` sets the entire inheritance tree to `element_blank()`, and thus no lines appear anywhere on the plot background. Finally, we note that the vertices with no edges were incorporated into the plot by adding their labels to the edges data frame in both the 'from_id' and 'to_id' columns before passing the edges data frame to `ggplot`.

```
ggplot(data = theme_elements$edges, aes(from_id = parent, to_id = child)) +
  geom_net(vertices = theme_elements$vertices, directed = TRUE, vlabel = TRUE) +
  expand_limits(x = c(0,1.2), y = c(0,1)) +
  theme_net
```

The inheritance structure is plotted in figure 5.

2.4 Madmen Network (again?? or no?)

leaving blank for now.

2.5 College Football

Note: would be best with coloring vertices by conference This next example comes from M.E.J. Newman's network data web page (Girvan and Newman, 2002). It is an undirected

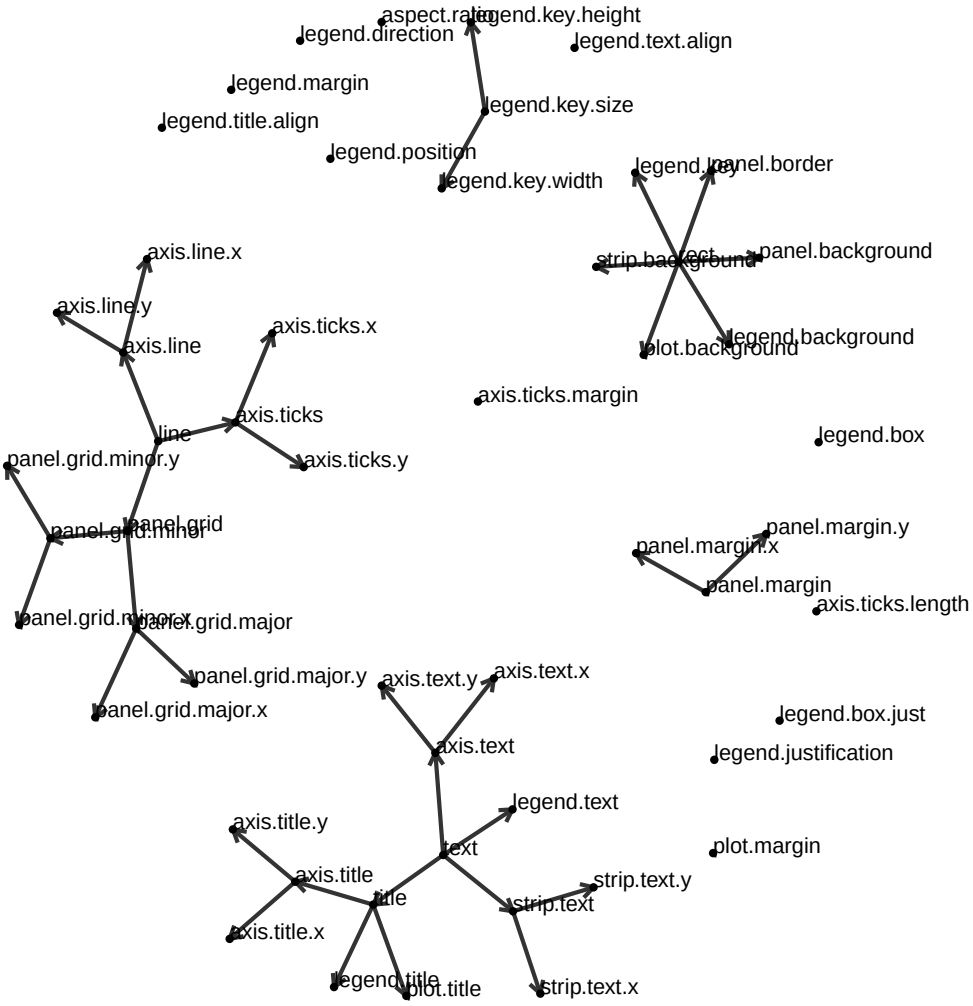

Figure 5. Inheritance structure of `ggplot2` *theme elements. This is a recreation of the graph found at* `http://docs.ggplot2.org/current/theme.html`.

network consisting of all regular season college football games played between Division I schools in Fall of 2000. There are 115 vertices and 613 edges: each vertex represents a school, and an edge represents a game played between two schools.

```
ggplot(data = football$edges, aes(from_id = from, to_id = to, elinetype = in.conf + 1)) +
  geom_net(vertices = football$vertices, ealpha = 0.2, vsize = 2, vlabel = TRUE, aes(vcolou
  expand_limits(x = c(0,1), y = c(0,1)) +
  theme_net

## Error in eval(expr, envir, enclos):  object 'in.conf' not found
```

The network of football games is given in figure 6.

2.6 Les Misérables

Note: data set has edge weightings for number of co-occurences of characters. esize would be useful here!
This next network comes from Knuth (1993). It is an undirected network of coappearances of characters in Victor Hugo's *Les Misérables*. There are 77 vertices representing each of the 77 characters in the book, and an edge connects two vertices if those two characters appear in the same chapter of the book. There are 254 edges in this network. The edges are also weighted by the number of coappearances. The largest weighting is 31, between the characters Jean Valjean and Cosette.

```
ggplot(data = lesmis$edges, aes(from_id = from, to_id = to, esize = value/mean(value))) + g
```

2.7 Protein Interaction Network in Yeast

This example of a protein interaction network comes from Jeong et al. (2001). It is the complete protein-protein interaction network in the yeast species *S. cerevisiae*. There are 1,870 proteins that make up the vertices of this network, and there are 2,240 edges between them. These edges represent "direct physical interactions" between any 2 proteins (Jeong et al., 2001, p. 42).

```
ggplot(data = yeast$edges, aes(from_id = V1, to_id = V2)) +
  geom_net(vertices = yeast$vertices, ealpha = 0.1, valpha = .5) +
  expand_limits(x = c(0,1), y = c(0,1)) + theme_net

## [1] "new net geom"
## [1] "it would be nice at this point to check, whether layout is one of the supported fun
## [1] "draw net"
```

Figure 6. The network of regular season Division I college football games in the season of fall 2000. The vertices and their labels are colored by conference.

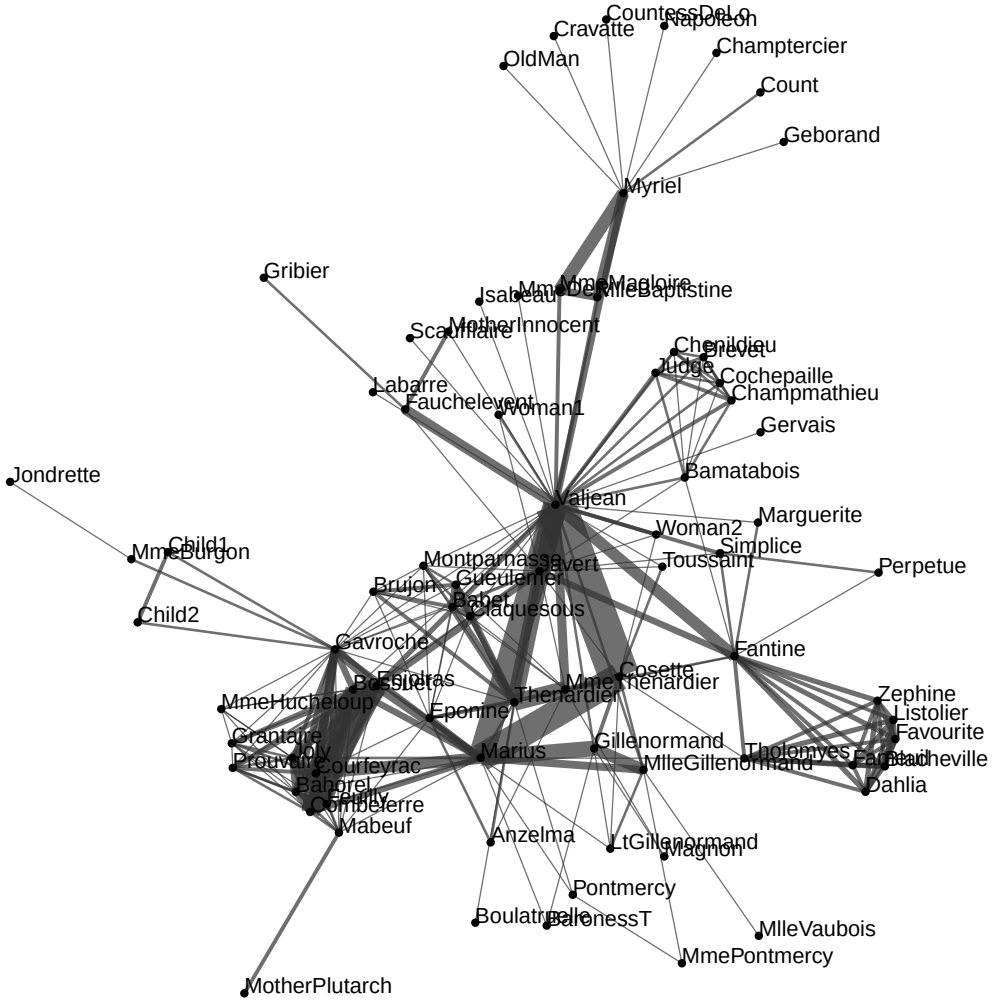

Figure 7. Coappearance network of characters in Victor Hugo's Les Misérables.

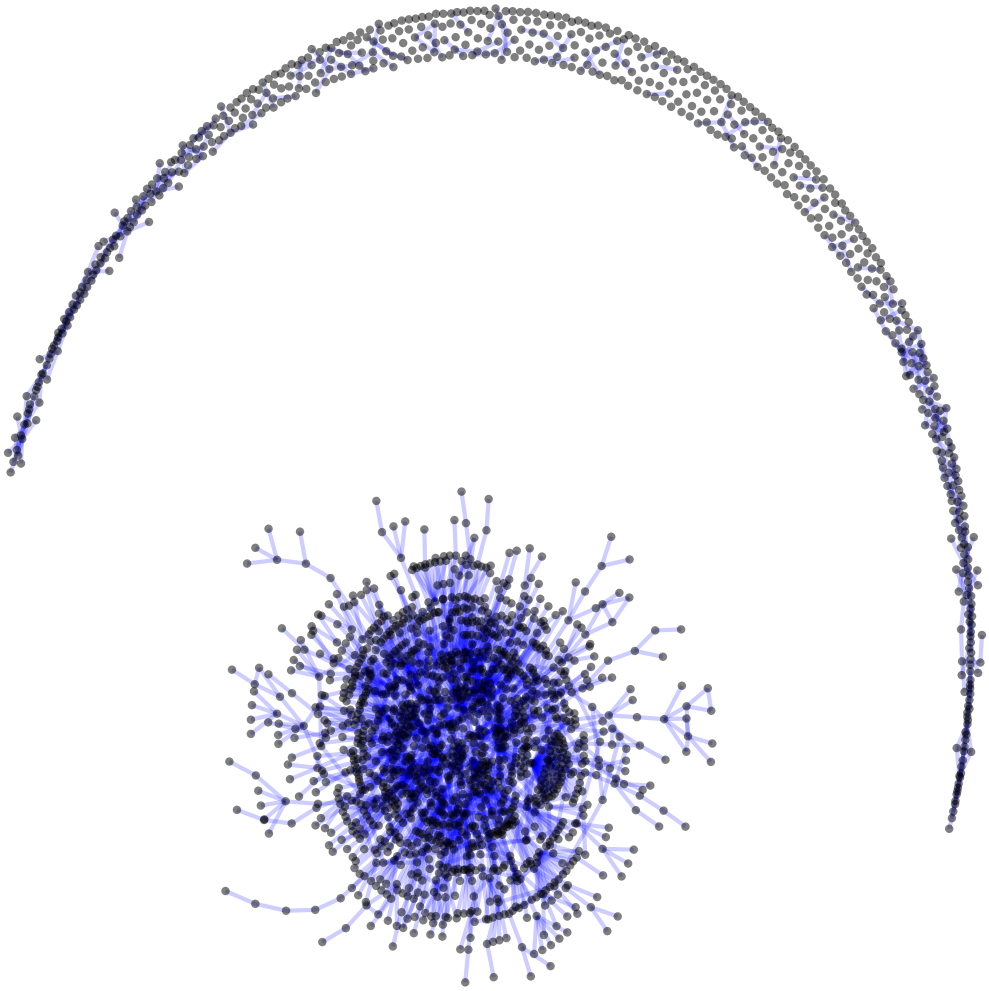

Figure 8. Complete protein-protein interaction network in S. cerevisiae.

3. TECHNICAL DETAILS

I have no idea exactly how detailed I should be. I'm going to attempt to be detailed without parsing each line in the code, but I do want to describe how everything works in detailed enough descriptions. I know you'll tell me if I'm too detailed or not detailed enough! In order to construct the `geom` in `ggplot2`, we needed two `R` script files: a 'stat' file and a 'geom' file. The stat file performs all the necessary calculations on the two input data frames, and the geom file performs the drawing of the network.

3.1 Creating `stat_net`

In order to plot the network with only the edge connections and vertex names, we relied on the `sna` and `network` packages (Butts, 2014, 2008). First, we used the edges data frame to construct a network object with the `as.network()` function, then constructed an adjacency matrix from that network using the `as.matrix.network.adjacency()` function. This adjacency matrix is then passed to the chosen `gplot.layout.*()` function from the `sna` package. *When the layout argument is changed in the `geom_net()` function, the corresponding `gplot.layout.*()` function in `sna` is called with a `do.call()` statement.* This layout function produces a matrix of coordinates of the vertices, which we then transform into a data frame containing the coordinates of the edges using the `as.matrix.network.edgelist()` function in `network`. We are grateful to Moritz Marbach for his `gplot()` function which we used when creating this process (Marbach, 2011). At the end of the previous routine, we have a data frame with `x, y, xend,` and `yend` columns describing the edges and another data frame with `x,y` columns describing the vertices. These two data frames, along with the many `aes()` and other argument options get passed on to the actual network geometry for drawing.

3.2 Creating `geom_net`

We constructed `geom_net` using the drawing capabilities of the `geom_point`, `geom_line`, *and `geom_text`* functions that were already in `ggplot2`. Using these capabilities, we were able to seamlessly incorporate our network geometry into the `ggplot2` structure. We took our edges data frame and used that as our input to the `GeomSegment` draw function. Then, on top of that, we plot the vertices by passing our vertex data frame to the `GeomPoint` draw function. *Finally, we use the `GeomText` draw function to add the vertex labels when the `vlabel` argument is TRUE. We added an if/else statement in the `draw` function which created a labeling layer which is `NULL` if `vlabel = FALSE`, which is the default. We then created a new data frame for the label layer. This allows the vertex properties to be propogated through to the labels, including size and color.* This doesn't seem very long, but I already feel like there's too much detail! Should I add more?

References

Buldyrev, S. V., Parshani, R., Paul, G., Stanley, H. E., and Havlin, S. (2010), "Catastrophic cascade of failures in interdependent networks," *Nature*, 464, 1025–1028.

Butts, C. T. (2008), "network: a Package for Managing Relational Data in R." *Journal of Statistical Software*, 24.

— (2014), *sna: Tools for Social Network Analysis*, r package version 2.3-2.

Butts, C. T., Handcock, M. S., and Hunter, D. R. (2014), *network: Classes for Relational Data*, Irvine, CA, r package version 1.10.2.

Chang, W. (2013), *R graphics cookbook*, Sebastopol, CA: O'Reilly.

Cook, K., Grinstein, G., and Whiting, M. (2014), "Vast Challenge 2014," `http://vacommunity.org/VAST+Challenge+2014%3A+Mini-Challenge+1#Download_the_Datasets_Entry_Forms_and_Documentation`.

Csardi, G. and Nepusz, T. (2006), "The igraph software package for complex network research," *InterJournal*, Complex Systems, 1695.

Girvan, M. and Newman, M. E. J. (2002), "Community structure in social and biological networks," *Proc. Natl. Acad. Sci. USA*, 99, 7821–7826.

Jeong, H., Barabsi, S. P. M. A.-L., and Oltvai, Z. N. (2001), "Lethality and centrality in protein networks," *Nature*, 411, 41–42.

Kamada, T. and Kawai, S. (1989), "An Algorithm for Drawing General Undirected Graphs," *Information Processing Letters*, 31, 7–15.

Knuth, D. (1993), *The Stanford GraphBase : a platform for combinatorial computing*, New York, N.Y. Reading, Mass: ACM Press Addison-Wesley.

Marbach, M. (2011), "Visualizing networks with ggplot2 in R," `https://sumtxt.wordpress.com/2011/07/02/visualizing-networks-with-ggplot2-in-r/`.

Newman, M. E. J. (2010), *Networks : an introduction*, Oxford New York: Oxford University Press.

Schloerke, B., Crowley, J., Cook, D., Hofmann, H., Wickham, H., Briatte, F., Marbach, M., and Thoen, E. (2014), *GGally: Extension to ggplot2.*, r package version 0.4.7.

Watts, D. and Strogatz, S. (1998), "Collective dynamics of 'small-world' networks," *Nature*, 393, 440–442.

Wickham, H. (2009), *ggplot2: elegant graphics for data analysis*, Springer New York.