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, who is quite the "ladies' man." This fun example shows just how ubiquitous networks are.

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`. Just to give an idea of the popularity and the wide-spread use of `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. This is the user base we are aiming at by making network visualizations a part of `ggplot2`.

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.

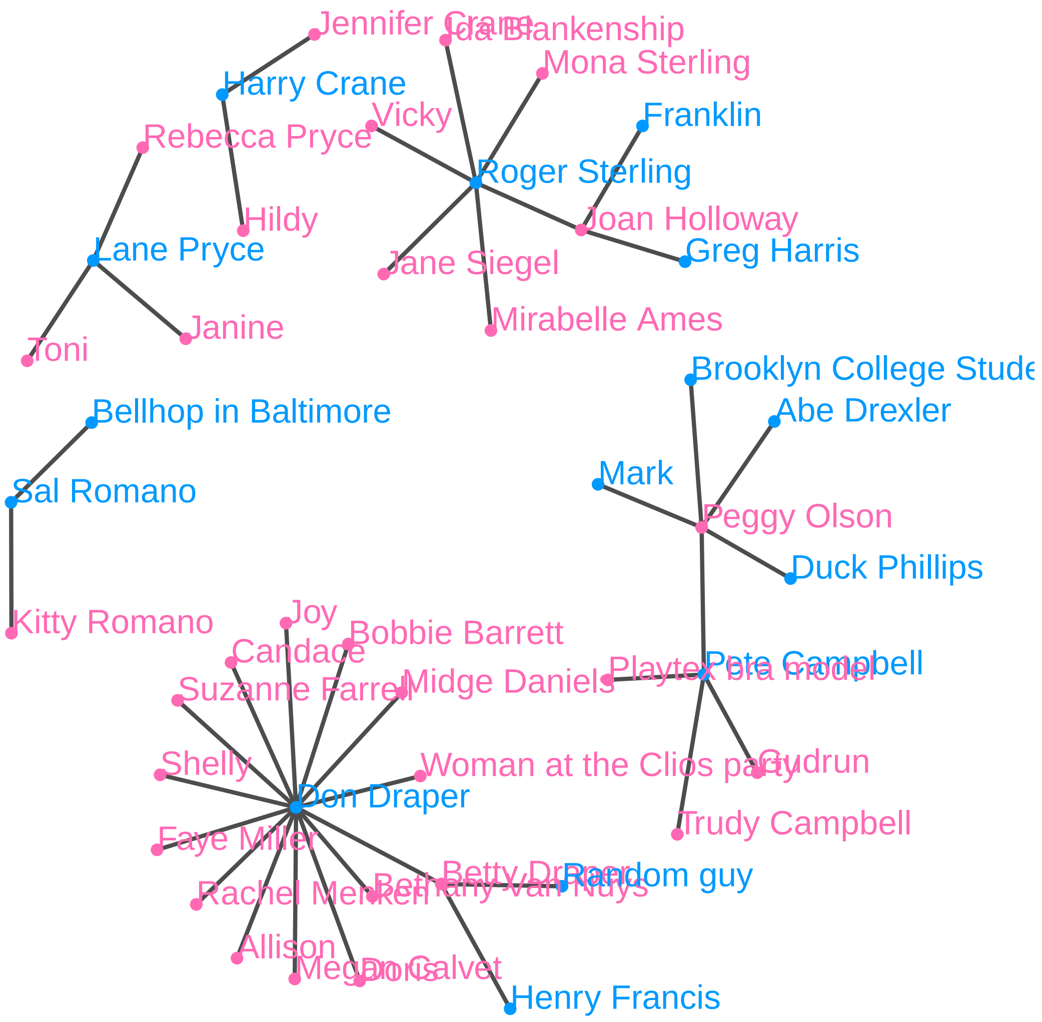

Figure 1. Graph of the characters in the show Mad Men who are linked by a romantic relationship.

1.1 Data Structure

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 variable is one called `label` which contains the vertex labels as they are encoded in the edges data set. This will be a formal requirement in the final version of the `geom_net` function. Each row represents one vertex. Other values of interest, such as grouping variables or degree of 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 match the names of the vertices in the vertex information data frame. The formal names for these columns are `from_id` and `to_id`, respectively. 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, we want to create all possible vertex aesthetic to mimic the usual aesthetics in `ggplot2` for points. The `vcolor` aesthetic can be an identity color (e.g. `color = "red"`) to change the color of all vertices, or, in the final version, it will 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. In the current version, 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. We use the `RColorBrewer` package to choose the colors for us in this paper (Neuwirth, 2014). It will also color the vertices along a color gradient according to the different values of some numeric variable such as degree. The vertex color defaults to black. 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 one of any numerical columns in the vertex data frame. Right now, the numerical values are strict point values: if the size of a vertex is two, then it will be two points in diameter. A point is the size of the default fontsize in the `grid` package. In our final version, the `vsize` will scale to minimum and maximum values as all other size aesthestics in `ggplot2` do. The `vshape` aesthetic is for changing the shape of the vertices from the default circle to other shapes, like square or triangle. It also changes the shape of the vertices according to some grouping variable, which is currently required to consist of integer values from 0 to 25. In the final version, it will be able to change shape based on different levels of a factor variable. Finally, the `valpha` aesthetic will change the vertex transparency to a set value, like 1/10 or 0.5. It operates in exactly the same way as the `alpha` aesthetic in `geom_point`. This aesthetic is very useful for networks with hundreds or thousands of vertices, which can easily crowd the static visualization. Making the vertices more transparent will better show the underlying structure in the network.

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 segments. 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. In the final version, the `ecolor` aesthetic will automatically color according to levels of a grouping variable or according to a gradient scale associated with a numeric 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`. Currently, the column associated with this aesthetic must be integer valued and take on values from zero to six. In the final version, it will change the edge linetypes according to levels of a grouping variable. 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, and currently maps the values directly to points, just like the `vsize` aesthetic. In the final version, this aesthetic will work exactly like the sizing in `ggplot2`, by scaling sizes to the minimum and maximum values. The `ealpha` aesthetic can change the edge transparency to a set value on the interval $[0, 1]$. This property is especially useul for large networks with a lot of connections or with vertices of relatively high degree. Finally, we will add an `elabel` aesthetic to print the name or number assigned to each edge on or next to it. This aesthetic will be very useful for visualizing random graphs or Markov processes where each edge probability is of interest.

1.4 Other Arguments

Similar to the other geometries and functions in `ggplot2`, our geometry takes several arguments outside of the aesthetic mapping parameters. One such argument is the `vlabel` argument. The `vlabel` aesthetic is a logical value that, if set to `TRUE` prints the name or number associated with each vertex in the `label` column in order to identify it on the plot. In the final version of this geometry, we will add capabilities to adjust the size and positioning of the labels. For the moment, the color is set to the vertex color, the size is set to twice the vertex size, and all other possible parameters are set to their defaults in `geom_text()`. The `vlabel` argument is most useful for smaller network objects where all vertex names can be printed on their corresponding vertices and still be read clearly. Next, the `directed` argument is a logical value that identifies whether or not the network is directed. When this value is true, arrows are created on the end of the line segment that

Layout Name	Description
`"adj"`	a version of `"mds"` which scales the raw adjacency matrix
`"circle"`*	places vertices uniformly in a circle
`"circrand"`	places vertices randomly in a circle
`"eigen"`	places vertices based on the eigenstructure of the adjacency matrix
`"fruchermanreingold"`	uses a variant of Fruchterman and Reingold's force-directed placement algorithm
`"geodist"`	a version of `"mds"` which scales the matrix of geodesic distances
`"hall"`*	places vertices based on the last two eigenvectors of the Laplacian of the input matrix
`"kamadakawai"`	generates a vertex layout using a version of the Kamada-Kawai force-directed placement algorithm
`"mds"`	places vertices based on a metric multidimensional scaling of a specified distance matrix
`"random"`	places vertices randomly
`"segeo"`	a version of `"mds"` which scales the squared euclidean distances between row-wise geodesic distances
`"seham"`	a version of `"mds"` which scales the Hamming distance between rows/columns of the adjacency matrix
`"target"`	produces a "target diagram" or "bullseye" layout using a force-directed placement algorithm.

Table 1. All possible layouts to pass to `geom_net`. All descriptions are from Butts (2014). Note that the layouts marked with an asterisk (*) are the only layouts that do not take layout parameters. All others use the default layout parameters in `sna` for now.

corresponds to the `to_id` value for each edge. The arrows match the edges in coloring, linetype, and size, and the default length of the arrow is set using the `unit` function in the `grid` package to 0.015 of the normalised parent coordinates of the plot. Finally, the `layout` argument takes a character value corresponding to the possible layouts in the `sna` package that are created by the `gplot.layout.*()` family of functions. The default layout is the Kamada-Kawai layout. This is a force-directed layout for undirected networks (Kamada and Kawai, 1989). There are, however, many other layouts possible.

All layouts that `geom_net` is currently capable of graphing are listed in table 1. There is not currently functionality to add layout parameters to the layout function within `geom_net`, though this will be included in the final version. Therefore, all functions in table 1 that are not marked with asterisks are using their respective default parameters according to the `sna` documentation for `gplot.layout`.

2. MANY MANY EXAMPLES

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

2.1 Blood Donation

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. 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 because loops exist on every vertex in this example, as blood between two people of matching ABO and RhD type can always be exchanged.

```r
ggplot(data = blood$edges, aes(from_id = from, to_id = to)) +
  geom_net(vertices = blood$vertices, vcolour = I('red'), layout = 'circle',
           vlabel = TRUE, vsize = I(3), directed = TRUE) +
  expand_limits(x = c(0,1), y = c(0,1)) + theme_net
```

This network is shown in figure 2. Here, we have used the aesthetics `vcolour` and `vsize` set to identity values to change the size and color of all vertices. We have also used the `layout` and `vlabel` arguments to change the default layout to a circle layout and to print the blood types, respectively. The circle layout places blood types of the same ABO type next to each other and spreads the vertices out far enough to distinguish between the various “in” and “out” types. You can tell clearly from this plot that the O- type is the universal donor: it has out degree of seven and in degree of zero. Additionally, you can see that the AB+ type is the universal recepient, with in degree of seven and out degree of zero. Anyone looking at this plot can quickly determine which type(s) of blood they can receive and which type(s) can receive their blood.

2.2 Email Network

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. In order to better visualize the structure of the communication network between employees, all emails that were sent out to all employees are removed in the subsequent examples.

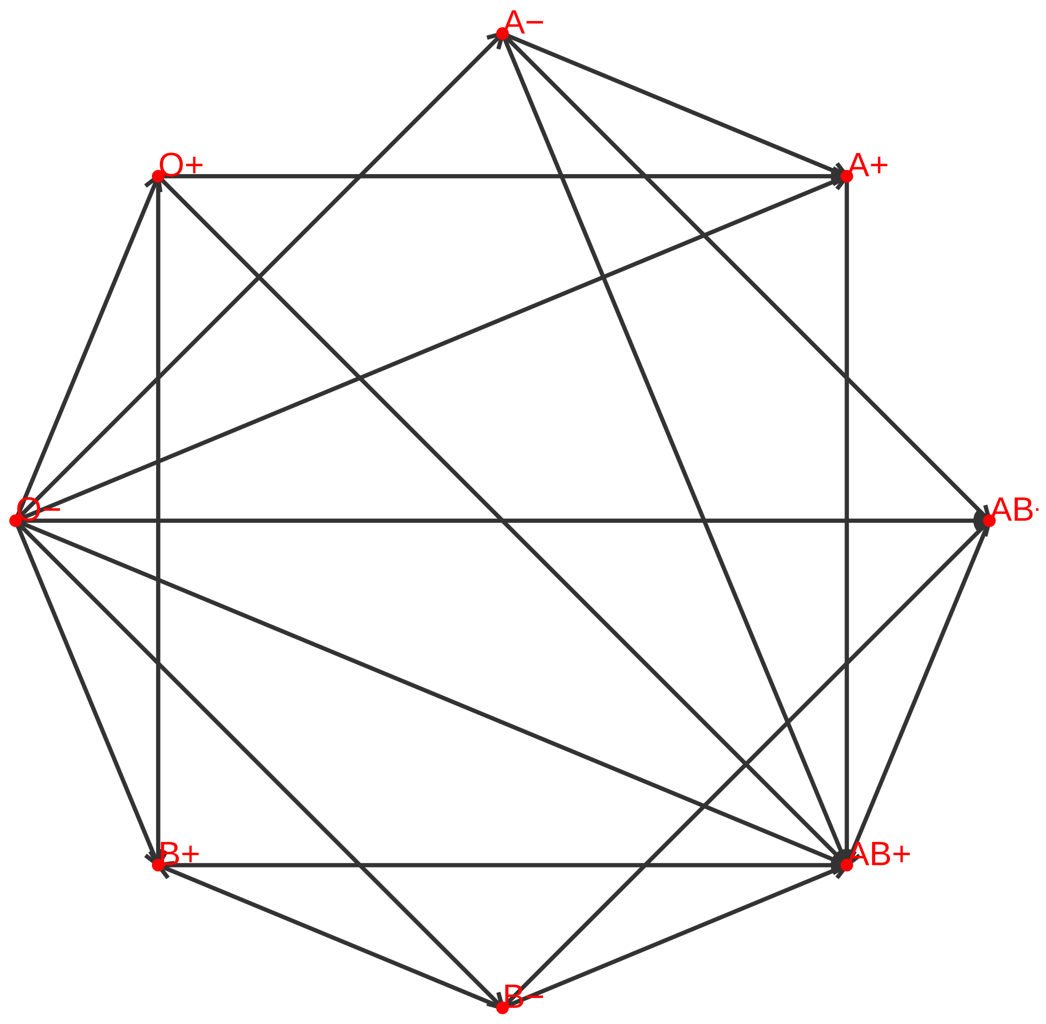

Figure 2. Network of blood donation possibilities in humans by ABO and RhD blood types.

```r
ggplot(data = subset(email.net, nrecipients < 54), aes(from_id = From, to_id = to)) +
  geom_net(vertices = employee,ealpha = 0.1, vsize = I(4), directed = TRUE,
           aes(vcolour=cols[as.numeric(CurrentEmploymentType)])) +
  expand_limits(x=c(0,1), y = c(0,1)) + theme_net
```

This network is plotted in figure 3. There are six distinct clusters in this network which almost perfectly correspond to the six different types of employee in this company: administration, engineering, executive, facilities, information technology, and security. Unfortunately, there is currently no legend to associate color with employment type. This will be remedied in the final version of our geometry. Additionally, the edges between employees in the same cluster are much darker than edges between employees in different clusters. This is due to the value of the `ealpha` aesthetic: more emails between two employees lead to darker edges. The value is set to 0.1 in this example, so that 10 or more emails between 2 employees results in a completely opaque edge. This pattern of heavy communication between employees of the same type is fairly unsurprising. To make this visualization more interesting and informative, we facet the network by day: each panel in 4 shows the different email networks associated with each day of the week.

```r
ggplot(data = subset(email.net, nrecipients < 54), aes(from_id = From, to_id = to)) +
  geom_net(vertices = employee, ealpha = 0.2, vsize = I(2), directed = TRUE,
           aes(vcolour=cols[as.numeric(CurrentEmploymentType)])) +
  expand_limits(x=c(0,1), y = c(0,1)) + facet_wrap(~day, nrow = 2) + theme_net
```

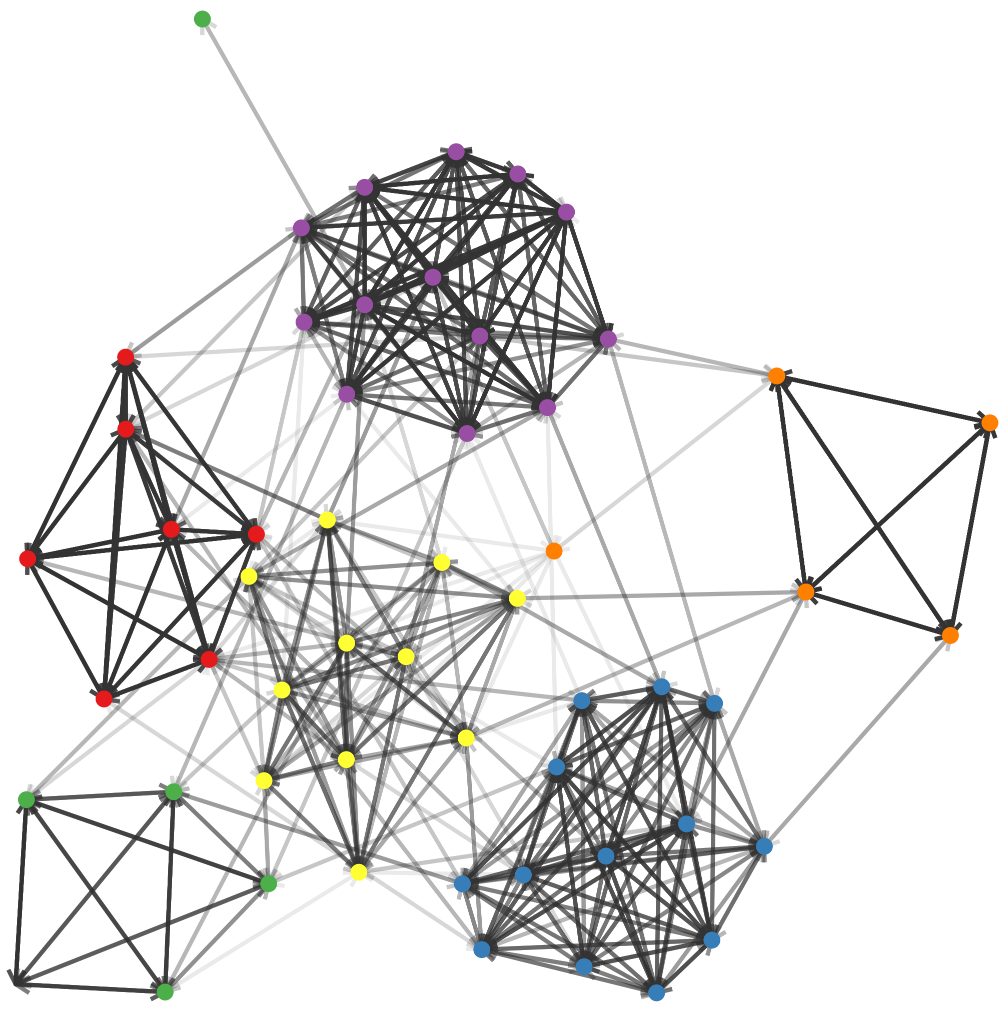

Figure 3. Email network within a company over a two week period.

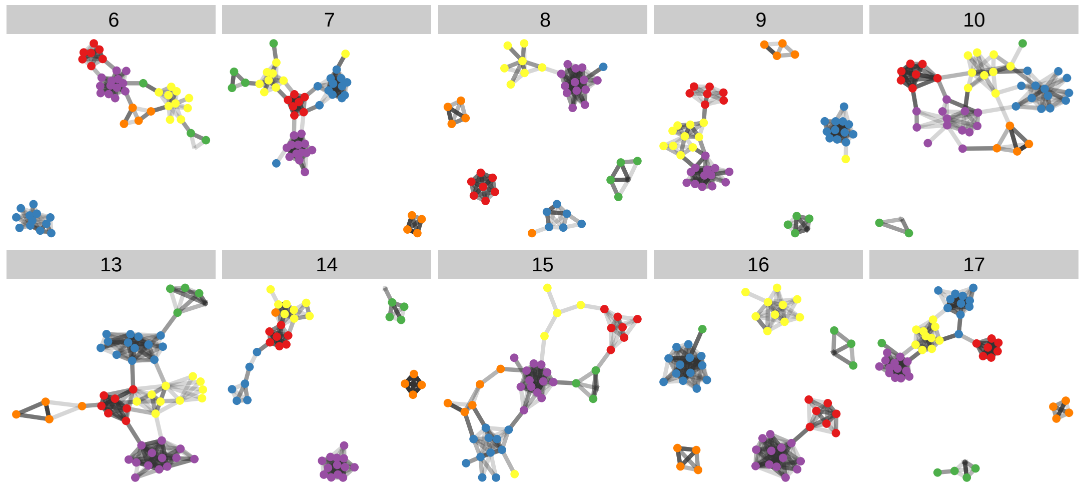

Figure 4. The same email network as in figure 3 facetted by day of the week.

This plot is shown in figure ??. With the facetting, we can see that there are several days where one or more departments do not communicate with any of the other departments. There are only two days, 13 and 15, without any isolated department communications. Facetting is one of the major benefits of creating a geometry for networks in `ggplot2`. Facetting quickly separates dense networks into separate subnetworks for easy visual comparison and analyses.

2.3 `ggplot2` Theme Elements

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. Each vertex represents one possible theme option. There is an arrow from one theme option 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 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`.

```r
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. In this plot, it is easy to quickly determine the parent and child vertices. Using this plot made creation of the `theme_net` object used throughout these examples very simple. We just made set each of the major parent elements, `text`, `rect`, and `line` to `element_blank()` and then set the aspect ratio equal to one.

2.4 Mad Men Networks

The following code creates the network example given in the introduction. We changed the vertex size and edge color for all vertices and edges, included vertex labels, and colored the vertices according to the character's gender.

```r
ggplot(data = madmen$edges, aes(from_id = Name1, to_id = Name2)) +
  geom_net(vertices = madmen$vertices, vsize=3, vlabel= TRUE, ecolour="grey30",
           aes(vcolour=c( "#FF69B4", "#0099ff")[as.numeric(Gender)])) +
  expand_limits(x = c(0,1.25), y = c(0,1)) + theme_net
```

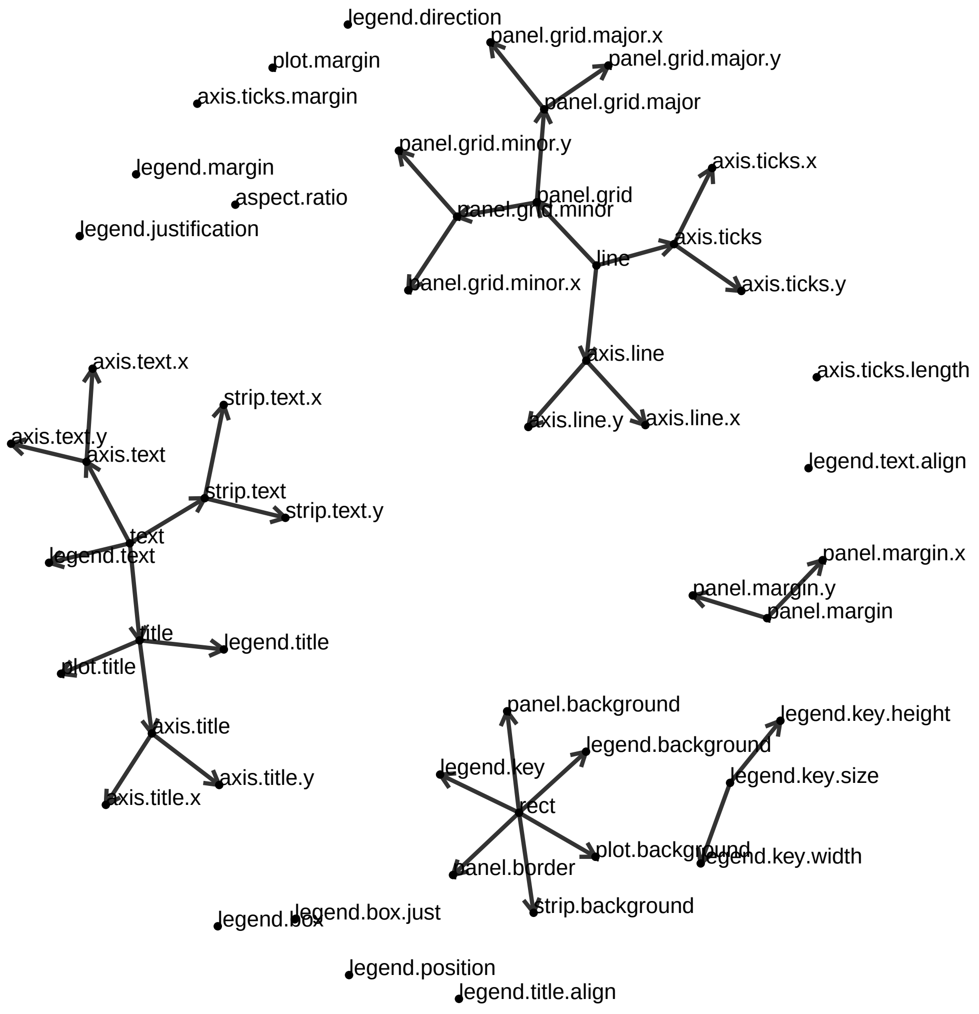

Figure 5. Inheritance structure of `ggplot2` *theme elements. This is a recreation of the graph found at* `http://docs.ggplot2.org/current/theme.html`.

There is another Mad Men network included in the `gcookbook` package (Chang, 2013). It is a directed network, also of romantic relationships between characters, but it also includes advances made by one character that were rejected by the other. For example, Roger Sterling made advances toward Betty Draper, but Betty refused him, and so there is a directed edge going from Roger to Betty, but not from Betty to Roger. If the advance is reciprocated, like between Sal Romano and the Bellhop, there are two directed edges between the two vertices.

```
ggplot(data = mm.directed$edges, aes(from_id = Name1, to_id = Name2)) +
  geom_net(vertices = mm.directed$vertices, directed = T, vlabel = T,
           vsize = I(2.5), layout = 'fruchtermanreingold',
           aes(vcolour=c( "#FF69B4", "#0099ff")[as.numeric(Gender)])) +
  expand_limits(x = c(0,1.1), y = c(0,1)) + theme_net
```

This network is shown in figure 6. This network is much more connected than the previous Mad Men example. It also allows us to see much more of the drama in the show. For instance, you can see that Roger made advances towards Betty, his business partner's wife, which is much more scandalous than what we can see in the first network.

2.5 College Football

This next example comes from M.E.J. Newman's network data web page (Girvan and Newman, 2002). It is an undirected 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. There is an additional variable in the vertex data frame corresponding to the conference each team belongs to, and there is an additional variable in the edge data frame that is equal to one if the game occured between teams in the same conference or zero if the game occured between teams in different conferences.

```
colors <- brewer.pal(12, name = 'Paired')
ggplot(data = football$edges, aes(from_id = from, to_id = to,
                                  elinetype = in.conf + 1)) +
  geom_net(vertices = football$vertices, ealpha = 0.3, vsize = 2, vlabel = TRUE,
           aes(vcolour=colors[as.numeric(factor(value))])) +
  expand_limits(x = c(0,1), y = c(0,1)) + theme_net
```

The network of football games is given in figure 7. Here, we have changed the `elinetype` aesthetic to correspond to games that occur between teams in the same conference or different conferences. These lines are dotted and solid, respectively. We have also assigned a different color to each conference, and the vertices and their labels are colored according

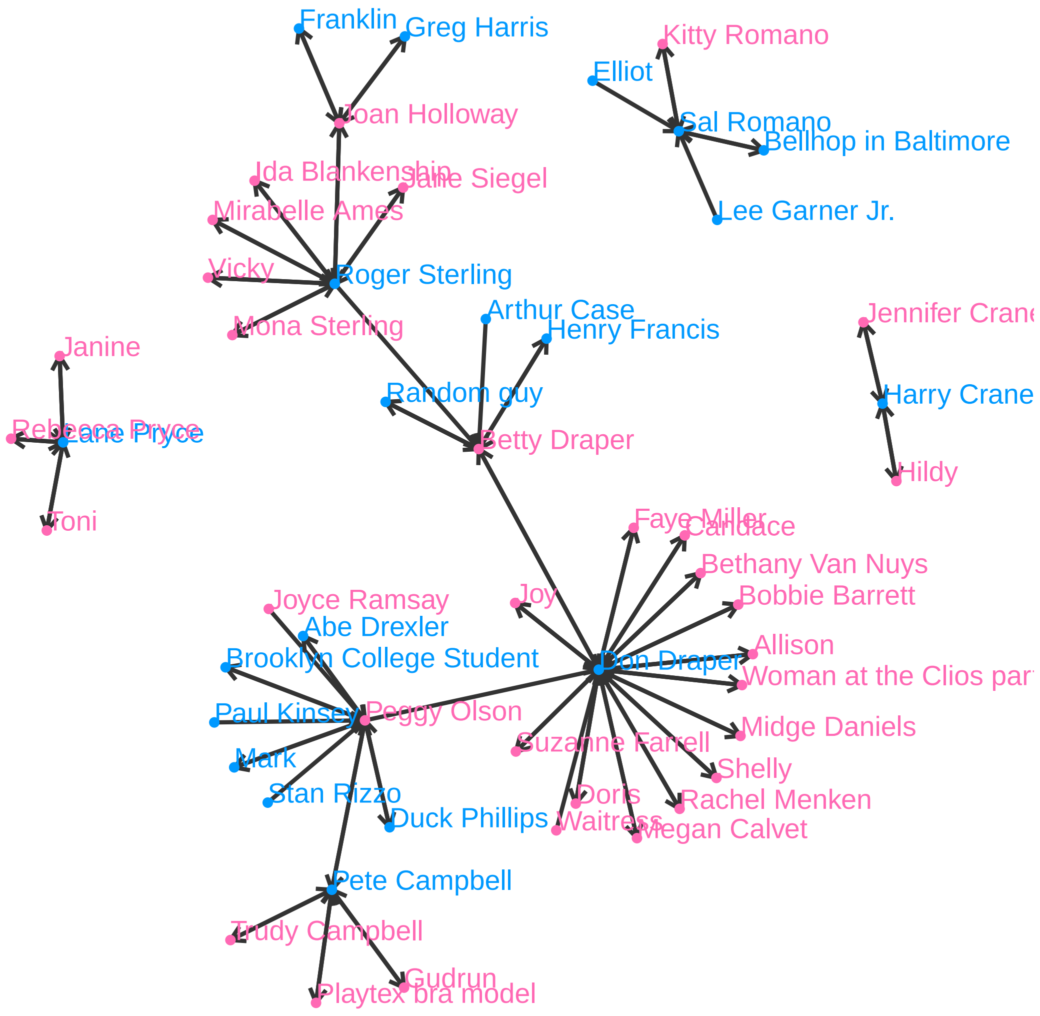

Figure 6. A directed network of relationships in Mad Men.

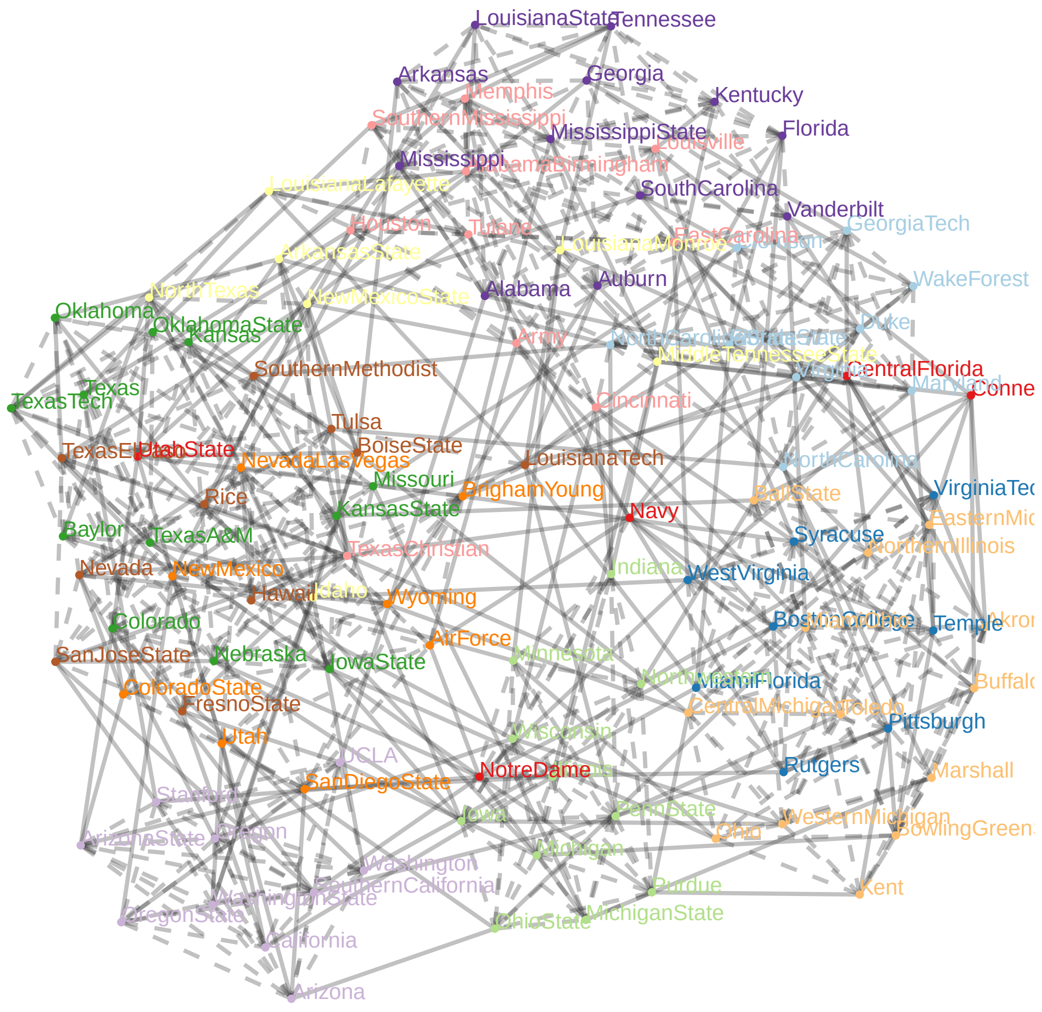

Figure 7. The network of regular season Division I college football games in the season of fall 2000. The vertices and their labels are colored by conference.

to their conference. This coloring and changing of the linetypes make the structure of the game network easier to view. There is one conference consisting of Navy, Notre Dame, Utah State, Central Florida, and Connecticut, which is spread out all over the network, whereas most other conferences' teams are all very close to each other because they play within conference much more than they play out of conference. At the time, these five schools were all independents and did not have a home conference. Without the coloring capability, we would not have been able to pick out that difference as easily.

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$edges, aes(from_id = from, to_id = to,
                                esize = value/mean(value))) +
  geom_net(vertices = lesmis$vertices, ealpha = 7/10,
           layout = 'fruchtermanreingold', vlabel=TRUE) +
  expand_limits(x = c(0,1.1), y = c(0,1)) + theme_net
```

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"
```

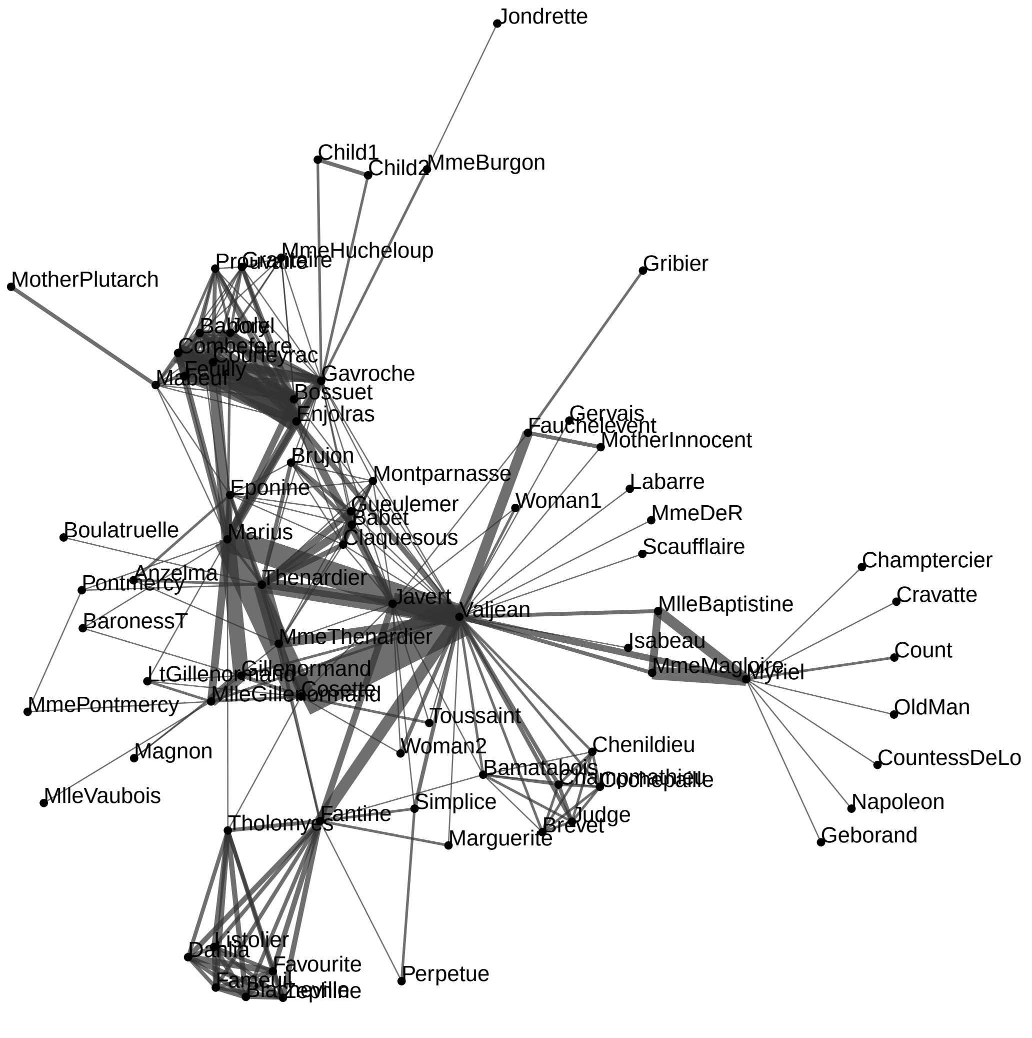

Figure 8. Co-appearance network of characters in Victor Hugo's Les Misérables.

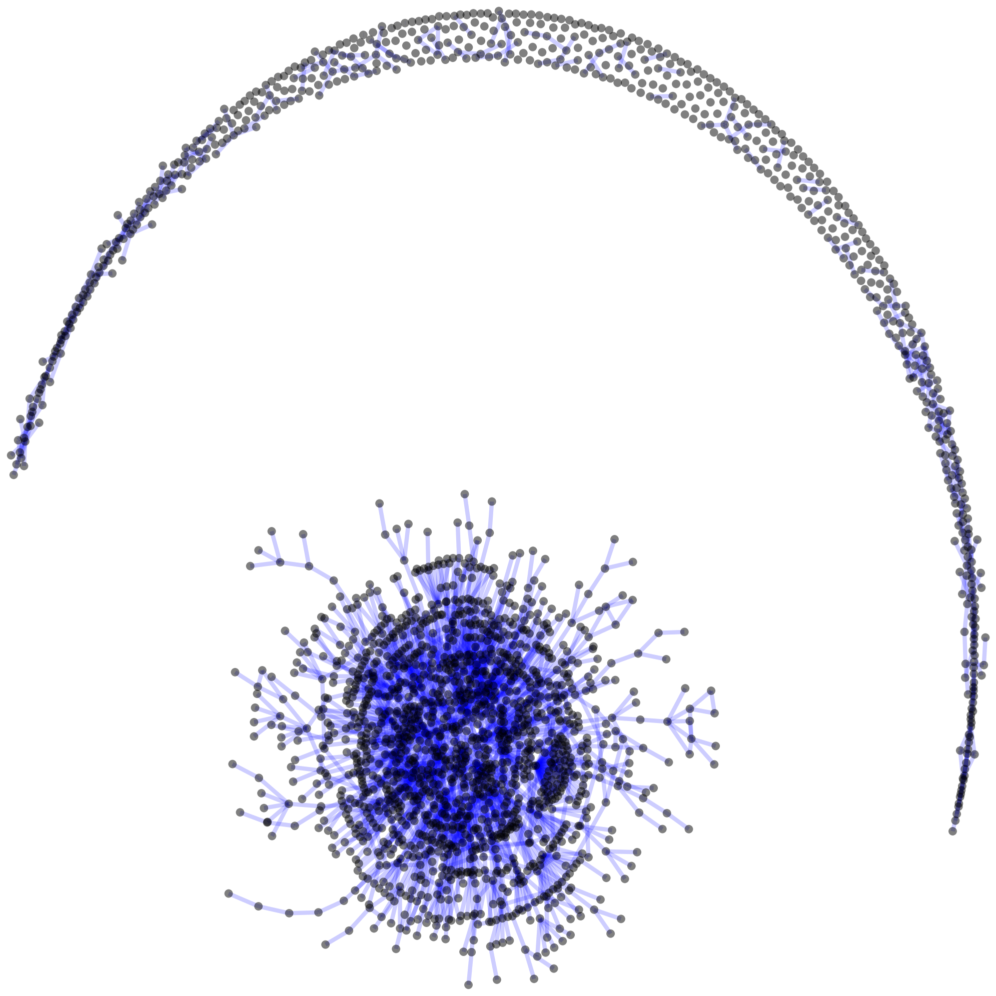

Figure 9. Complete protein-protein interaction network in S. cerevisiae.

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! You're doing just fine, keep going!

XXX have a look at the ggplot2 help online and include something along these lines for both stat_net and geom_net

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`

mention that a network is constructed from different layers - the node layer of points, an edge layer of line segments and an optional text layer of labels.

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.

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/.

Neuwirth, E. (2014), *RColorBrewer: ColorBrewer Palettes*, R package version 1.1-2.

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.